CSE344 System Programming Final Project Report

June 10, 2021

Student: Furkan Sergen Dayioglu
Student No:121044015
Instructor: Erchan APTOULA

1 Overview

In this assignment, we were asked to implement rudimentary sql database with server client. Server will take the port , logfile path, thread pool size and dataset path. In this homework, dataset will be csv file.

Csv file will be read and parsed as an sql file. Will be saved into data structure. I used char*** table for this topic.

2 Synchronization Problem Between Threads

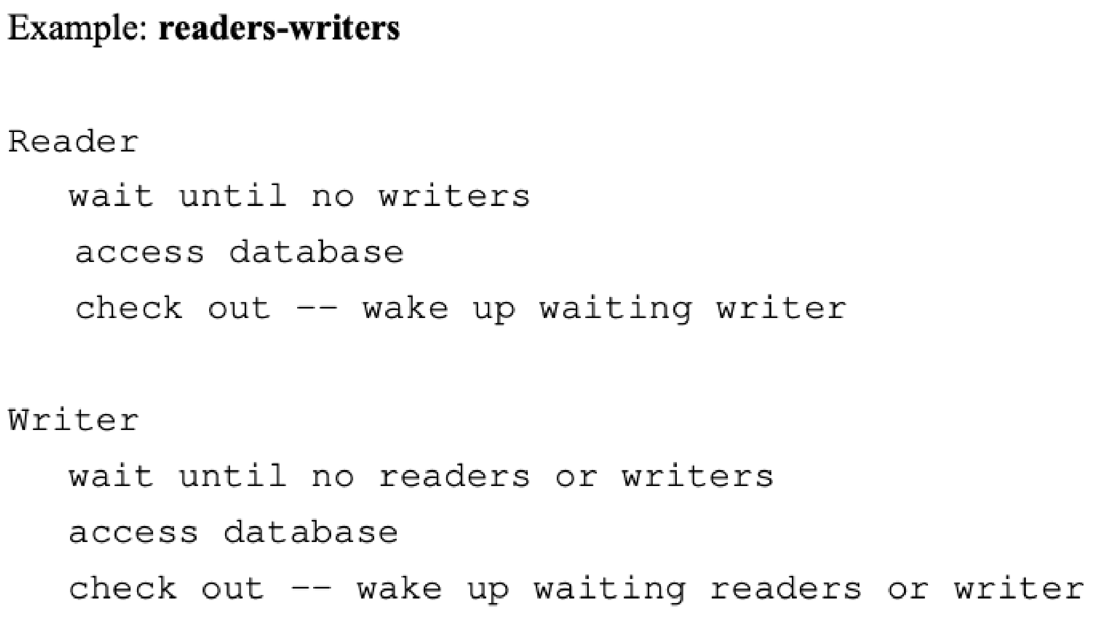
In this part of homework, in case of synchronization issue, we had to deal with reader-writer problem. In order to solve this issue, we had to use mutexes and condition variables.

If we talk about readers and writers problem, here we have a special case. That means our working threads are both readers and writers. Which also means while a thread evaluates SELECT query, it will be reader, because select query does not change any entry in table, But while evaluating UPDATE query, it will be writer.

2.1 Critical section

I used mutexes to create critical section. One of the critical section checks if thread is free and available, second one is, created to provide synchronious between readers and writes.

Writers-Readers Figure



Example: **readers-writers**

```
Reader
   wait until no writers
   access database
   check out -- wake up waiting writer

Writer
   wait until no readers or writers
   access database
   check out -- wake up waiting readers or writer
```

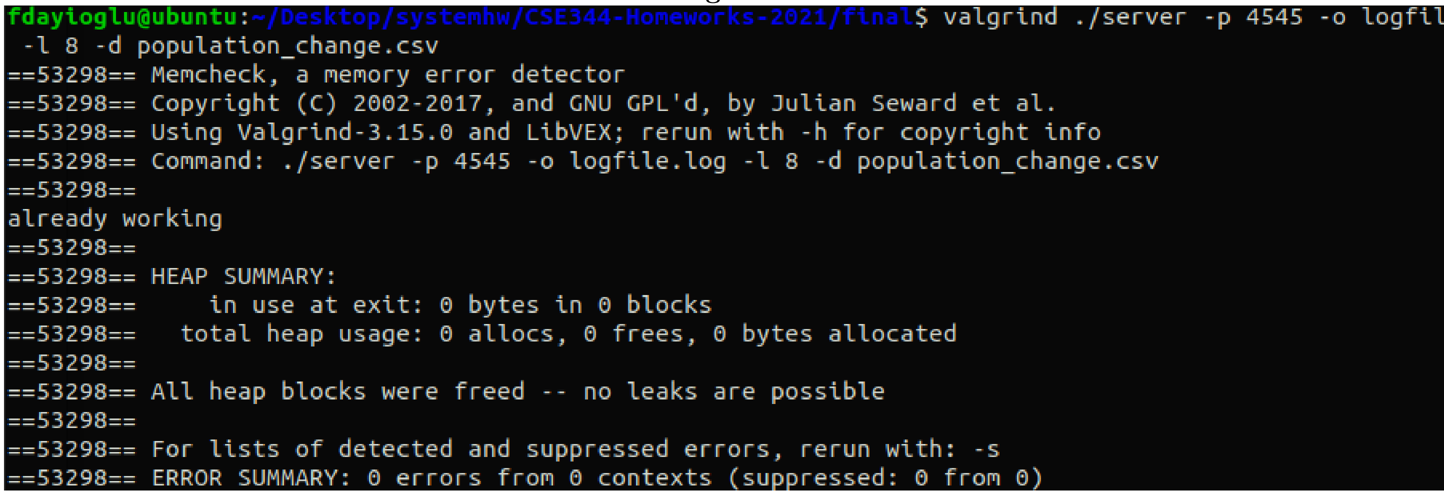
3 SIGINT situation

When termination signal come, it has been asked to running threads will finish their jobs gracefully, and free their resources and then join. Signal handler frees resources

4 Avoiding Double Instantiation

In this challenge, according to my researches on github, i inspired from some old student, and i used shared memory, because it was not allowed to use flocks to create any temporary process file.

One Instance Figure



```
fdayioglu@ubuntu:~/Desktop/systemhw/CSE344-Homeworks-2021/final$ valgrind ./server -p 4545 -o logfil
 -l 8 -d population_change.csv
==53298== Memcheck, a memory error detector
==53298== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==53298== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==53298== Command: ./server -p 4545 -o logfile.log -l 8 -d population_change.csv
==53298==
already working
==53298==
==53298== HEAP SUMMARY:
==53298==     in use at exit: 0 bytes in 0 blocks
==53298==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==53298==
==53298== All heap blocks were freed -- no leaks are possible
==53298==
==53298== For lists of detected and suppressed errors, rerun with: -s
==53298== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

5 Thread pool

PTHREAD_T* type of thread pool allocated regardless the command line argument which determines the size of thread pool. As we know threads get memory adresses (void*) as parameters. I created a threadP_t struct in order to pass the essencial parameters to thread.

6 SQL Evaluation

In this part, i used a wrapper sql_parser funstion. But it only checks first token. If it is select, then calls select_parser else calls update_parser.

6.1 Select Query

Tokenize rest of the query string into tokens to find necessary column names and check if disctinction asked. Then select_Q workes as READER of the program and get the table, send it to client

6.2 Update Query

Tokenize rest of the query string into tokens to find necessary column names and check the where column asked. Then update_Q workes as WRITER of the program and update the table, send effected row count to client

7 Notes

Because of some technical issues, i could not be able to test my program very well. But ıf we examine it part by part, it leaves no wasted memory.