Report FINAL
System Programming

Baran Hasan Bozduman
171044036

June 10, 2021

Problem Definition

The required in this project a setting up client-server communication also some functionalities of SQL database (like SELECT, SELECT DISTINCT, UPDATE). The tricky part is there will be a thread pool in the server so main thread will distribute the the queries to threads and each thread should be busy with the related client so there will be kind of producer consumer problem to manage available threads. Also since there will be other threads and we will use same database to read and write we should protect that and we can obtain that by using reader writer paradigm.

Design Decision

.

SERVER

Firstly I declare structs for queue and data set(its a struct and it has char* and size of char pointer of it) also i declare my threads as global in an array and they are also struct but they just only keep thread ids

Inside of main I initialized a named semaphore to prevent double instantiating since they are initialized by kernel and are managed by them so it does not let to run it second time it returns an error instead of that so until you send SIGINT signal you can not run that program second time

Then u get terminal values by using get opt and check them in case of user can make mistake

I read data and place in the memory and all the things are dynamic.

I create a socket and bind an address on it. I initialized listen value as 100 so is there will be more connection than that it can not accept.

I allocate space for threads according to pool size and create threads, i send their ids as parameter so each thread will know which is which.

I determined the available thread numbers as pool size at first and it enters the infinite while loop

In while loop i check the number of available threads by using conditional variable and if there is no one to start a job it will wait for a signal from pool threads. Then we can accept new connection request.

Then I add the accepted request in the queue so each thread can poll from queue. while we are doing that we should protect them with mutex.

After I added new request to queue i signal the threads so they can take it and do their jobs

In thread method it waits for signal from queue after it get from main thread it continues to run.

Takes a request from queue

Then it reads request from connection and sends it to parse function

In parse function it parses the requests

Select the function either read or write.

While we are doing that we implement reader writer paradigm as learnt in class.

In communication phase i send first row number so client knows how many time it will make read.

When client is done it will send a finish message which is "DONE" so server can close that socket and can accept new connections.

Or if gets SIGINT signal it exits from threads and frees the places allocated.

CLIENT

In client side things more easy it behaves according to mes

It sends DONE message when the queries done which belongs related user

Other than those it prints data according to incoming responses from server. You can see the outputs below

To show program doe not allow to run second time

```
anonxx@eXmachina final]$ ./a.out -p 8090 -o log -l 5 -d mac3
anonxx@eXmachina final]$ kill -SIGINT 78498
anonxx@eXmachina final]$ ./a.out -p 8090 -o log -l 5 -d mac3
anonxx@eXmachina final]$ ./a.out -p 8090 -o log -l 5 -d mac3
t's Already Created[anonxx@eXmachina final]$
```

Program runs in background as daemon process you can see ps -aux result

```
anonxx      78834  0.4  0.9 1141072 76220 ?       Sl   02:24   0:00 thumbnail.so [
root        78856  0.0  0.0      0     0 ?       I    02:24   0:00 [kworker/6:2-e
anonxx      78862  0.0  0.0  43500   104 ?       Sl   02:24   0:00 ./a.out -p 809
anonxx      78889  0.0  0.0  11608  4732 pts/1    R+   02:24   0:00 ps -aux
[anonxx@eXmachina final]$
```

As below you can see output for just one client

```
[anonxx@eXmachina final]$ ./client -i 1 -a 127.0.0.1 -p 8090 -o queries
Client-1 connecting to 127.0.0.1:8090
Client-1 connected and sending query 'SELECT * FROM TABLE;'
Server's response to Client-1 is 10 records, and arrived in 0.0 seconds.
(1623281741) Series_reference Period Data_value Suppressed STATUS UNITS Magnitude Subject Group Series_title_1 Series_title_2 Series_title_3 Series_title_4 Series_title_5
(1623281741) BDCQ.SEA1BB 2011.06 80078  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2019.03 102031  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2019.06 90836  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2019.09 88032  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2019.12 100531  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2020.03 100947  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2020.06 93239  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2020.09 92891  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2020.12 102338  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
Client-1 connected and sending query 'SELECT * FROM TABLE;'
Server's response to Client-1 is 10 records, and arrived in 0.0 seconds.
(1623281741) Series_reference Period Data_value Suppressed STATUS UNITS Magnitude Subject Group Series_title_1 Series_title_2 Series_title_3 Series_title_4 Series_title_5
(1623281741) BDCQ.SEA1BB 2011.06 80078  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2019.03 102031  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2019.06 90836  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2019.09 88032  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2019.12 100531  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2020.03 100947  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2020.06 93239  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1AA 2020.09 92891  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623281741) BDCQ.SEA1BB 2020.12 102338  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
Client-1 connected and sending query 'SELECT Series_reference, Period, Series_title_5 FROM TABLE;'
Server's response to Client-1 is 10 records, and arrived in 0.0 seconds.
(1623281741) Series_reference Period Series_title_5
(1623281741) BDCQ.SEA1BB 2011.06
(1623281741) BDCQ.SEA1AA 2019.03
(1623281741) BDCQ.SEA1AA 2019.06
(1623281741) BDCQ.SEA1BB 2019.09
(1623281741) BDCQ.SEA1BB 2019.12
(1623281741) BDCQ.SEA1AA 2020.03
(1623281741) BDCQ.SEA1BB 2020.06
(1623281741) BDCQ.SEA1AA 2020.09
(1623281741) BDCQ.SEA1BB 2020.12
A total of 3 queries were executed, client is terminating.
```

Below you can see for the update operation

```
Client-1 connected and sending query 'UPDATE TABLE SET Period='ZZZ' , Data_value='XXX', Suppressed='OOO' WHERE Series_reference='BDCQ.SEA1BB';'
(1623282061) Series_reference Period Data_value Suppressed STATUS UNITS Magnitude Subject Group Series_title_1 Series_title_2 Series_title_3 Series_title_4 Series_title_5
(1623282061) BDCQ.SEA1BB ZZZ XXX OOO F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1AA 2019.03 102031  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1AA 2019.06 90836  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1BB ZZZ XXX OOO F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1BB ZZZ XXX OOO F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1AA 2020.03 100947  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1BB ZZZ XXX OOO F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1AA 2020.09 92891  F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
(1623282061) BDCQ.SEA1BB ZZZ XXX OOO F Number 0 Business Data Collection - BDC Industry by employment variable Filled jobs "Agriculture Forestry and Fishing" Actual
Server's response to Client-1 is 5 records updated, and arrived in 0.0 seconds.
```

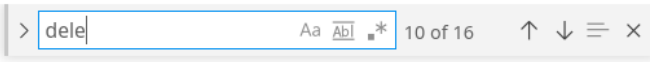
There are some queries which is created by me

```
1 SELECT * FROM TABLE;
2 SELECT * FROM TABLE;
3 SELECT * FROM TABLE;
1 SELECT * FROM TABLE;
4 SELECT Series_reference, Period, Series_title_5 FROM TABLE;
10 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
11 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 UPDATE TABLE SET Period='ZZZ' , Data_value='XXX', Suppressed='OOO' WHERE Series_refere
11 SELECT * FROM TABLE;
12 SELECT * FROM TABLE;
10 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 UPDATE TABLE SET Period='ZZZ' , Data_value='XXX', Suppressed='OOO' WHERE Series_refere
11 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
11 SELECT * FROM TABLE;
12 SELECT * FROM TABLE;
10 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
11 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
4 SELECT * FROM TABLE;
11 UPDATE TABLE SET Period='ZZZ' , Data_value='XXX', Suppressed='OOO' WHERE Series_refer
```

there is a script file which includes 16 client also there is 5 or 6 threads

```
#!/bin/sh
./client -i 1 -a 127.0.0.1 -p 8090 -o queries &
./client -i 2 -a 127.0.0.1 -p 8090 -o queries &
./client -i 3 -a 127.0.0.1 -p 8090 -o queries &
./client -i 4 -a 127.0.0.1 -p 8090 -o queries &
./client -i 5 -a 127.0.0.1 -p 8090 -o queries &
./client -i 6 -a 127.0.0.1 -p 8090 -o queries &
./client -i 7 -a 127.0.0.1 -p 8090 -o queries &
./client -i 8 -a 127.0.0.1 -p 8090 -o queries &
./client -i 9 -a 127.0.0.1 -p 8090 -o queries &
./client -i 10 -a 127.0.0.1 -p 8090 -o queries &
./client -i 11 -a 127.0.0.1 -p 8090 -o queries &
./client -i 12 -a 127.0.0.1 -p 8090 -o queries &
./client -i 13 -a 127.0.0.1 -p 8090 -o queries &
./client -i 14 -a 127.0.0.1 -p 8090 -o queries &
./client -i 15 -a 127.0.0.1 -p 8090 -o queries &
./client -i 16 -a 127.0.0.1 -p 8090 -o queries &
```

we can see by log file just 16 times we get delegated message which means for each client just one thread was busy




```
1   Executing with parameters:
2   -p 8090
3   -o log
4   -l 5
5   -d mac3
6   Loading dataset...
7   Dataset loaded in 0.001 seconds with 41 records.
8   Thread #0: waiting for connection
9   Thread #1: waiting for connection
0   Thread #2: waiting for connection
1   Thread #4: waiting for connection
2   Thread #3: waiting for connection
3   A connection has been delegated to thread id #0
4   A connection has been delegated to thread id #1
5   Thread #0: received query 'SELECT * FROM TABLE;'
```

the number of query which is 42 as in queries file

```
73  Thread #1: received query 'SELECT * FROM
74  Thread #1: query completed, 41 records have been returned
75  Thread #3: waiting for connection
76  No thread is available! Waiting...
77  A connection has been delegated to thread id #3
78  Thread #3: received query 'SELECT * FROM TABLE;'
79  Thread #3: query completed, 41 records have been returned
80  Thread #2: waiting for connection
81  No thread is available! Waiting...
82  A connection has been delegated to thread id #2
83  Thread #2: received query 'SELECT * FROM TABLE;'
```

there is 42 result returned that means each queries has done

```
1   Executing with parameters:
2   -p 8090
3   -o log
4   -l 5
5   -d mac3
6   Loading dataset...
7   Dataset loaded in 0.001 seconds with 41 records.
8   Thread #0: waiting for connection
9   Thread #1: waiting for connection
10  Thread #2: waiting for connection
11  Thread #4: waiting for connection
12  Thread #3: waiting for connection
13  A connection has been delegated to thread id #0
14  A connection has been delegated to thread id #1
15  Thread #0: received query 'SELECT * FROM TABLE;'
Thread #0: query completed, 41 records have been returned
Thread #4: waiting for connection
Thread #0: waiting for connection
Termination signal received, waiting for ongoing threads to complete.
All threads have terminated, server shutting down.
```

when the Server gets SIGINT signal it prints that to the log file

The Requirements I have achieved

It provides all requirements but one I tried the rules below and they worked properly on my computer

1. No compilation error
2. No compilation warning
3. $Make\check{f}ile$
4. $ReportinLa\check{t}exformat$
5. Printed usage information in case of missing or invalid argument
6. Program did not crashed
7. No zombie process
8. No deadlock
9. No poor synchronization

10. Submitted on time
11. Informed user in case of system errors
12. ...

The Requirements I have failed

Since i could not manage the time i can not implement SELECT DISTINCT so don't try the query files which includes DISTINCT operations, Algorithm was ready bu I could not do that because of insufficient time(The idea was after get the value I will create a column and mark them if it occurs second time so when i send it I would just send the unmarked rows of that column)