CMSC 621

Project 2: Clocks, Multicast, and COMMIT: Assignment 2

Distributed mutual exclusion

Anurag Pawar (LA22171)

- **Requirements specifications**

The goal of 'n' node distributed system is to achieve mutual exclusion with 'Centralized mutual exclusion algorithm'. A node will work as the centralized leader which will connect with all the other nodes and handle the mutual exclusion. Locking mechanism is used to avoid the race condition.

- **Program Design**

The project consists of two files.

1) *server.cpp*
2) *client.cpp.*

Server will accept the connection from number of clients and perform the mutual exclusion and client will send the request or response to server. The server's code will be executed once but number of clients will be spawned to achieve distributed environment. Server will wait for all clients to connect and then only it will start the centralized mutual algorithm. A shared file will be handled by all clients, they all will update the number in the shared file.

- **Code Overview**

The connection between clients and the server is achieved with the help of *UNIX-based TCP sockets*. Following socket programming functions are used for communication purposes.

socket(): To create a communication endpoint and will return a file descriptor.

accept(): To accept new connections.

bind(): To assign the address to the socket.

listen(): To make server to listen (look for) connection requests.

connect(): Used by client to connect to a server.

To handle multiple clients, *POSIX threads* are used. Each client is handled by a thread. Following commands are used for threads.

pthread_t : Declaration of thread.

pthread_create : Creation of thread function.

pthread_join : Command to wait for a thread to finish.

For avoiding race condition locking mechanism is used with *lock* variable, commands as follows:

pthread_mutex_lock(&lock): to apply the lock

pthread_mutex_unlock(&lock): to release the clock

Flow of the code

1) Once the connection is established between leader and all the nodes, it will start the implementation of centralized mutual algorithm.
2) Each client will send 'REQUEST' to leader to gain the access of the file.
3) The leader will give access to first client with 'OK' message and put rest in a queue.
4) Once the client in use will finish the modifying of the number in the file, it will send 'RELEASED' to the leader.
5) The leader will give access to the next client in the queue, once the access is granted then it will remove that client with *pop()*.
6) In this way all the clients will be given access to the shared file.

- **Compiling instructions**

1. Run the make file.
2. Server will start first. Run '*./server*' to start server.
3. Run *python3 spawn.py* in terminal with same directory as code folder, it will spawn clients.

(Note: If you get 'Binding Failed' error please change the port number in both files.)

- **Tradeoffs and possible improvements**

The time daemon can handle 'n' number of clients but in order to achieve this one need to update the definition of 'number_of_clients' from the header file to that particular number.

#define number_of_clients 5

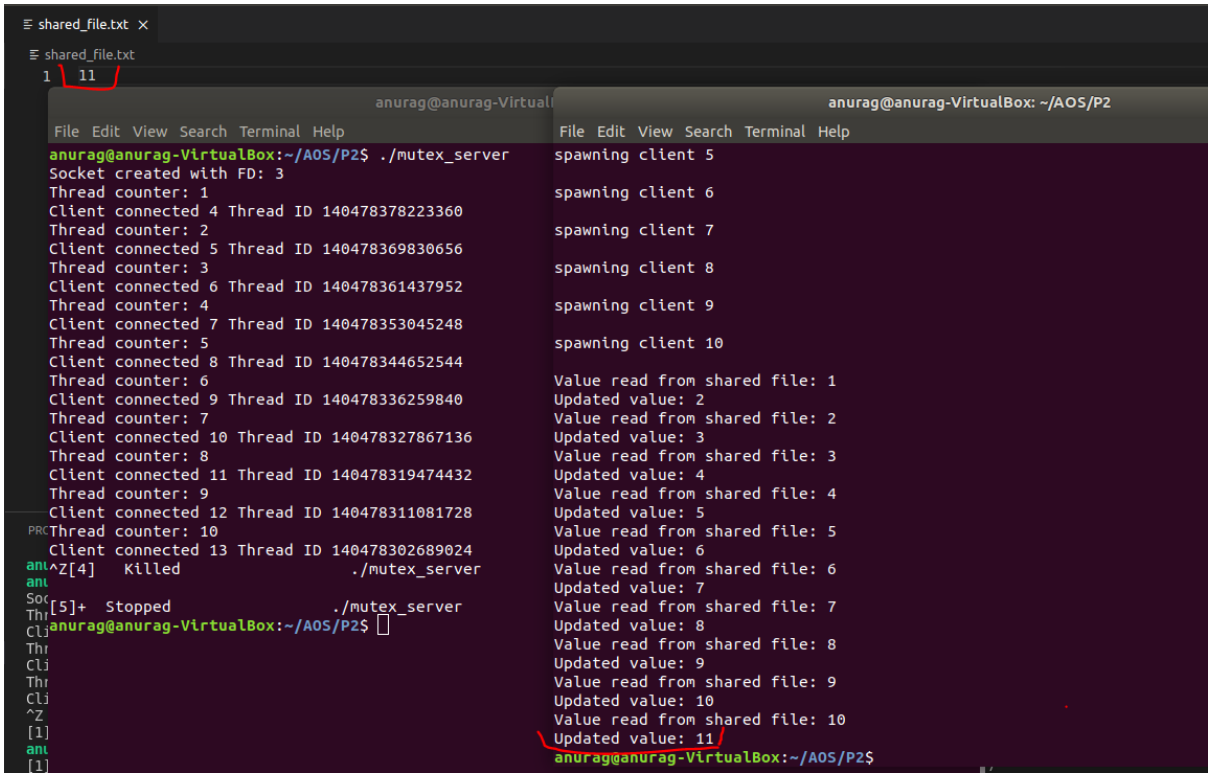
Same change is required in '*mutex_spawn.py*' file. The range of the for loop in the file needs to be adjusted according to number of clients.

- **Thoroughness of evaluation**

Code has been tested with different number of clients, screenshot as below:

Execution with 3 clients:

Execution with 10 clients:


shared_file.txt
shared_file.txt
1 11
anurag@anurag-Virtual
File Edit View Search Terminal Help
anurag@anurag-VirtualBox:~/AOS/P2$ ./mutex_server
Socket created with FD: 3
Thread counter: 1
Client connected 4 Thread ID 140478378223360
Thread counter: 2
Client connected 5 Thread ID 140478369830656
Thread counter: 3
Client connected 6 Thread ID 140478361437952
Thread counter: 4
Client connected 7 Thread ID 140478353045248
Thread counter: 5
Client connected 8 Thread ID 140478344652544
Thread counter: 6
Client connected 9 Thread ID 140478336259840
Thread counter: 7
Client connected 10 Thread ID 140478327867136
Thread counter: 8
Client connected 11 Thread ID 140478319474432
Thread counter: 9
Client connected 12 Thread ID 140478311081728
Thread counter: 10
Client connected 13 Thread ID 140478302689024
^Z[4] Killed ./mutex_server
[5]+ Stopped ./mutex_server
anurag@anurag-VirtualBox:~/AOS/P2$
anurag@anurag-VirtualBox: ~/AOS/P2
File Edit View Search Terminal Help
spawning client 5
spawning client 6
spawning client 7
spawning client 8
spawning client 9
spawning client 10
Value read from shared file: 1
Updated value: 2
Value read from shared file: 2
Updated value: 3
Value read from shared file: 3
Updated value: 4
Value read from shared file: 4
Updated value: 5
Value read from shared file: 5
Updated value: 6
Value read from shared file: 6
Updated value: 7
Value read from shared file: 7
Updated value: 8
Value read from shared file: 8
Updated value: 9
Value read from shared file: 9
Updated value: 10
Value read from shared file: 10
Updated value: 11
anurag@anurag-VirtualBox:~/AOS/P2$