Name: Katerina Baousi
Candidate no: 32438

CW2 COMSM2001- Socket programming

Project report

In this assignment we were asked to implement a server that will be responsible to manipulate the 2-way communications between the database and a pool of threads. Sockets created and used in this implementation follow the socket flow diagram and actions were performed in order to handle multithreading. Furthermore, two mutexes and one condition variable were created in order to manage the alterations performed by the threads in sensitive values.

When the server starts up, a fixed sized pool of 4 worker threads is created. These worker threads are delegated to the worker function and wait there until a connection is made on the data port. Two sockets were created (one for the data port and one for the control port). Then once created the two sockets bind to each port and wait until they listen to something happening on the data port or the control port. At this point polling needs to take place and the polling structure needs to be initialised in order to test if there is data available to any of the two sockets. An infinite loop begins inside which it is decided whether the event happened on the controller socket or on the data socket.

In the first case the connection is instantly accepted and the information written is read and parsed. If the user types COUNT the number of items in the database will return.If he types SHUTDOWN the mutex (named mut) will be locked, the value of the global variable shut_down will be changed and broadcasted.Then the mutex will be unlocked again. In that way the working threads will finish their work and the waiting thread will break the loop. All of them will meet in the end of main thread join and the mutexes and condition variables responsible for their synchronisation will be destroyed.

If the event happens on the data port the connection gets accepted , the mutex is locked and the special file descriptor produced is stored inside an array in a characteristic position indicating the increasing number of connections. This array is a global variable used by the workers as well. Then , if the number of accepted connections(queue_counter) is less than the number of threads used by the program, the connection is established by changing the value of the connectionEstabl global variable and signaling. Furthermore, the value of the accepted connections (queue_counter) is increased and the mutex is unlocked. If the program has reached the point when it has accepted the 4th connection (i.e the queue_counter is equal to the number of threads) then a special flag is used in order to prevent the connections from being established.

The threads once created they are delegated to the worker function and wait in an infinite loop until they get the signal that a connection has been made in the data port. It is worth mentioning that signal() function is the most adequate in this case because it can wake up one thread at a time and since there are multiple threads waiting it will be really inappropriate to wake them all at the same time in this implementation. In order for the threads to wait a mutex will be first locked and then the threads will wait until there has been a connection to the data port or a shut_down message has been broadcasted. In the second case they will break the loop and return to the main thread to join. In the first case , the flag showing the global variable connectionEstabl will be set to zero and the file descriptor of the connection will be read by the aforementioned global array. The mutex will be unlocked and the thread will enter another infinite loop responsible for reading and writing in the database. Once it has entered the loop the data given in the terminal are read and parsed and a mutex (dataMut) is locked in order to perform the appropriate changes in the database. The user can enter a series of commands and the procedure will repeat as it is (read,parse,lock,execute,unlock). However, when the user enters a blank line the connection closes, the dataMut unlocks and the loop breaks.

To sum up, this assignment focused on manipulating multithreading and socket programing. Four threads were created.In order to achieve synchronisation and avoid busy waiting two mutexes and one condition variable were used. The error handling part of the program was implemented by aborting the program and closing the connections when errors occurred in its vital components (e.g thread creation, socket creation, binding and listening) and by closing the connection on of the client when errors occurred during the reading and parsing procedures.