Programming Assignment (PA) -2
(Tic-Tac-Toe with Threads)
Report

CS307 – Operating Systems

14 November 2021
DEADLINE: 15 November 2021, 23:55

Rebah Özkoç
00029207

In my homework, I used the mutex mechanism of the Pthread (POSIX thread) library of C to solve multithread problems. The pseudo code for my locking algorithm:

```
thread play (symbol, n):
    while(true):
       lock(&mutex)
        // Check if the thread has turn
        if it's not symbol's turn:
           unlock(&mutex)
            continue
        change turn to the other symbol
        if game did not end:
            play once for the symbol
            if not symbol won:
                if is tie:
                    change game end status to end
                   unlock(&mutex)
                    return
            else: // symbol won the game
                change game end status to end
               unlock(&mutex)
                return
        else:
           unlock(&mutex)
           return
       unlock(&mutex)

run thread play function for two threads with symbols x and o.
```

I used a coarse-grained mutex mechanism to implement a tic-tac-toe game. By this way I prevented accessing to the game board by the two players (threads) at the same time and violating the turn rule. If I had used a fine-grained mutex and locked individual cells of the table instead of the whole table, I would also have achieved to prevent data racing during the writing to the game board array, but threads might have skipped their turn and incorrectness might have occurred.

In this implementation, firstly a thread locks mutex then checks if it has the turn, if not it unlocks the mutex and prevents any deadlock. Secondly thread checks if the game ended. By this way program prevents to play a false move if the game has already been ended by the other thread. Lastly if the did not end thread makes its move and checks if any winning or tie situation occurred then it unlocks the mutex to prevent any deadlock and returns or continues to loop according to the current situation. The thread locks mutex at the start of the while loop and unlocks mutex just before every return statement or end of while loop. This prevents every possible deadlock and provides atomicity for every critical section.