Mert Kılıçaslan

Tic-Tac-Toe with Threads (PA – 2) Report

CS307 - Operating Systems

Pseudo Code of The Locking Algorithm:

```
threadMutex
threadCond

int globalCounter = 0
bool gameEnd = false
winState: checkHorizontal, checkVertical, checkDiagonal

threadMainFunction(void structArgs){
        char array [ ][ ] = structArgs.arr
        int size = structArgs.size
        char letter = structArgs.letter

        if(letter == o)
                sleep(letter)

        int counter = 0
        while(globalCounter < size*size){
                mutexLock

                if(globalCounter is even){
                        globalCounter ++ , counter++
                        threadWait }

                        if(gameEnd)
                                break

                        int ri , rj = randomNumber() % size
                        array[ri][rj] = letter

                        if(winState){
                                gameEnd = true
                                threadSignal
                                threadUnlock
                                break}
                else{
                        globalCounter ++, counter++
                        threadSignal}
                mutexUnlock}}
```

Description of the Algorithm:

Two threads enter the routine function with different structs given that carries a letter, dynamically allocated array, and its size, if the letter 'o' thread enters the function first, it sleeps long enough to let thread ‘x’ take the lead. After that, the main loop iterates over until the loop counter reaches matrix size*size. Later, we have mutex lock to ensure that two concurrent threads do not simultaneously execute indexing and counting in the critical section. Then later two threads enter the if(globalCounter is even) condition while the second thread waits for the first thread to finish and give a signal. The first thread checks if the game is finished or not (for the second thread), then gets the random index numbers until the numbers are unique enough for the array. Then it enters the letter character to those indexes. And it checks if the array is on the winning state (for the first thread). Since we incremented the globalCounter for one, it is now an odd number. It enters the else part to signal other thread to wake up and go on, now the second thread becomes the first thread and unlocks the mutex and locks again. Until one of the threads enters the wining state condition, loops iterate over. If one of the threads enters the winning state, it signals others and breaks the loop, the other one gets the signal and also breaks and finishes the loop. If both threads do not enter the winning state, it means loop ended by local counter reach which means a tie, and otherwise, it means the last thread that enters its character to the array wins.