Yunus Delipınar 26617

PA2 REPORT

My locking algorithm is below as pseudo code:

```
pthread_con_t cond = PTHREAD_COND_INITIALIZER;   (global variable)
pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER; (global variable)
While (winning state is not reached and table is not full)
    Get row and column numbers
    Lock the thread using(pthread_mutex_lock(&lock)
    while (turn number is even)
        Allow player x to insert board
        Pthread_cond_wait(&cond, &lock)
    while(turn number is odd)
        Allow player 0 to insert board
        Pthread_cond_wait(&cond, &lock)
    If (game has not ended)
        While(selected row and column is not empty)
            Select new row and column
        Insert mark to the selected row and column of the board
        Print who inserted where
        Update turn
        Unlock the thread(pthread_mutex_unlock(&lock)
        Broadcast signal to break condition (pthread_cond_signal(&cond)
```

As you see in the code above I choose the global lock method because it is easy to use and since we are not required to do fine grading one global lock variable is more than enough for this assignment. The other reason why I choose this type of lock instead of using lock in a struct is with this method code became more readable elegant and less confusing. I placed mutexlock before selecting a row and column because if we placed just before the update part some issues may have risen as checking the location to insert I critical part. Finally, since I have used condition variables lock should be just before them.