Anıl Şen – 29556

CS307 PA3 REPORT

Discussion

The program,

allows each thread to look for a spot if one of the desired combinations has not been formed yet. Once the desired combination is satisfied inside the car, the program does not let other threads look for a spot until the car is driven by the assigned captain.

initially allows 4 threads to enter and it checks if there is 3A 1B case or vice versa. If so, the program calls one more thread to come so that we are sure of getting one of the desired combinations. At the end of each turn, the program checks whether we called one more thread in the turn. If so, we call just 3 threads more to come to check whether the desired combination can be formed by having 4 threads at total again. If not, that means we are facing the "3 to 1 case" again, so we need to call 1 more thread to pass. In case there is no need for calling 1 more thread, the program continues to allow 4 more threads to enter again.

resets the flags used for some operations at the end of each turn and releases the locks obtained right after a specific job finishes to avoid any deadlock problems and to maintain the sustainability of the subsequent turns if there is any.

contains,

- ➔ 1 semaphore with value 4 to allow for just 4 threads to enter initially and through "sem_post" method; we can signal the threads sleeping and can perform the mechanism of the program mentioned above, which is allowing 1, 3 or 4 more threads to come.
- ➔ 3 barriers with value 4, 1 barrier with value 3, 2 barriers with value 2, and 4 different mutexes for synchronization goals.

The logic behind the program

Line 1- Let 4 threads pass.

Line 2- If you have 3 1 Case (i.e. you have either 3A 1B or 3B 1A):

Line 3 - Allow 1 more thread to come.

Line 4 - Put them in the car.

Line 5 – If you still have threads, allow 3 more threads to come and go to line 2.

Line 6 - Else:

Line 7 - Put them in the car.

Line 8 – If you still have threads, go to line 1.

By using this approach, it is guaranteed that there exists exactly 4 *mid (spotting message mentioned in the document)* messages before the captain of the car declares its captainship, because once the proper combination

is achieved, we spot them in the car and have their print statements printed. Also, there are only 3 possibilities which the program allows; 2A-2B, 4A, 4B, thanks to the provided necessary “if” checks regarding the problematic case which is 3A-1B or 3B-1A.

Flow of the threads - Pseudocode

Try to get the semaphore with value 4, otherwise wait.

Lock “mutex_obj” and print the hello message, and then unlock mutex_obj.

IF a thread is from group A:

    Lock mutex_objA & Increment the Acount by 1 & Unlock mutex_objA

ELSE:

    Lock mutex_objB & Increment the Bcount by 1 & Unlock mutex_objB

IF TimeFor3Fans and OnlyOneShouldCallAFan is 0:

    Wait at a barrier with value 4.

ELSE IF TimeFor3Fans is 1 and OnlyOneShouldCallAFan is 0:

    Wait at a barrier with value 3.

IF a thread is from group A:

    Go through the barrier with value 2 that is associated with group A. Otherwise wait at the barrier.

ELSE:

    Go through the barrier with value 2 that is associated with group B. Otherwise wait at the barrier.

IF Acount is equal to 3 and B count is equal to 1 or vice versa:

    Lock the mutex_objSecond.

    IF OnlyOneShouldCallAFan is equal to 0:

        Set OnlyOneShouldCallAFan as 1.

        Increment the value of the semaphore by 1 through sem_post.

    Unlock the mutex_objSecond.

Wait at a barrier with value 4.

IF a thread is from group A:

    Lock mutex_objA. & Decrement the Acount by 1. & Unlock mutex_objA.

ELSE:

    Lock mutex_objB. & Decrement the Bcount by 1. & Unlock mutex_objB.

Lock “mutex_obj” and print the “finding spot” message, and then unlock mutex_obj.

Wait at a barrier with value 4.

Lock the mutex_obj.

IF OnlyOneShouldBeCaptain is equal to 0:

Set OnlyOneShouldBeCaptain as 1.

Print the captain message with carID.

Increment carID by 1.

Unlock the mutex_obj.

Lock the mutex_objSecond.

Increment threadNumberBeforeReset by 1.

IF threadNumberBeforeReset is equal to 4:

Set threadNumberBeforeReset as 0 & Set OnlyOneShouldBeCaptain as 0.

IF OnlyOneShouldCallAFan is 1:

Set OnlyOneShouldCallAFan as 0 & Set TimeFor3Fans as 1.

Increment the semaphore value by 1, 3 times by using sem_post.

ELSE:

Set TimeFor3Fans as 0.

Increment the semaphore value by 1, 4 times by using sem_post.

Unlock the mutex_objSecond.