Programming Assignment (PA) -3 (Riding to a Soccer Game) Report

CS307 – Operating Systems

20 November 2021
DEADLINE: 06 December 2021, 23:55

Rebah Özkoç
00029207

Pseudo code of the main function:

```
team_a_count = 0
team_b_count = 0

For team A create n number of threads and give team name as an argument.
For team B create k number of threads and give team name as an argument.

Initialize semaphore “A” with 0
Initialize semaphore “B” with 0
Initialize semaphore “general” with 5
Create a barrier with thread value 4
For every team A supporter thread call “create_supporter” function
For every team B supporter thread call “create_supporter” function
```

Pseudo code of the flow of threads in create_supporter function:

```
Set is_captain to false
Create temporary variables team_count, other_count, temp_sem, other_sem

Call wait semaphore for “general”
// This semaphore has value 5 it won’t allow more than 5 threads to synchronously // run after this line
Print “The string Thread ID: threadId, Team: team, I am looking for a car”

Lock the mutex “mutex”
Increase the current thread team count by one
If (with the last thread current team thread number==2 and the other team >= 2)
    Set is_captain to true
    Set this team’s count to 0
    Decrease other teams count by 2
    Post semaphore of this team one time
    Post semaphore of other team two times
    Unlock mutex “mutex”
If (with the last thread current team thread number == 4)
    Set this team’s count to 0
    Post semaphore of this team three times
Else: // There is not proper number of team supporters to fill a car
    Unlock mutex “mutex”
    Call wait semaphore for current team semaphore

Print I have found a car message
Create a barrier and wait for 4 threads to come
Only for the captain thread:
    Print captain message
Post semaphore of “general” semaphore
```

Synchronization Mechanisms:

In this assignment, I have used three semaphores, one barrier, and one mutex. One of the semaphores was a general semaphore and it was preventing to reach the car searching and founding code more than 5 threads. 5 random threads are ensuring a correct passenger distribution can happen. Because with 5 threads there can be

5 A – 0 B passengers, 4 A – 1 B passengers, 3 A – 2 B passengers, 2 A – 3 B passengers, 1 A – 4 B passengers, 0 A – 5 B passengers and so on.
In every situation there is exactly one true car combination.

The other two semaphores are for team A and team B. These semaphores have value zero and they behave like condition variables. When a correct combination happens, the last coming thread opens the correct number of semaphores and decides which threads should continue to form a car. This ensures the correctness because by these semaphores we decide how many threads and which type of threads will continue to car forming code.

The barrier is for waiting for the captain thread to print its message. It has value 4. It holds the captain thread and other threads until every four threads print their "I found a car" message.

The mutex is for preventing data race and deadlock. It protects the count variables and condition checks from the other 4 thread that is inside the car searching codes.