Programming Assignment (PA) - 3 Riding to a Soccer Game
Yiğit Kaan Tonkaz
29154

First of all, the problem indicates that there are 2 teams and their fans which should be multiple of four in total and each of them even. We need to put them into car in four groups by 2-2 or 4-0 groups and select 1 captain for ride.

<u>struct thread_</u>

In my code, I implement a struct called thread_ which has these properties,

```
struct thread_{
   char team;
   sem_t * sem_a;
   sem_t * sem_b;
   int * num_a;
   int * num_b;
};
```

I implement this struct because I need to reach the values when I am passing arguments in function. I use team as character A or B. I use 2 pointers that indicates the sem_t for each team to get in the condition thread and wait. I use num_a and num_b pointers that shows the num_A and num_B in the main as representing number of team fans waiting to get in the car. I use int pointers because it needs to change in each iteration(I do not use global variable instead I use pointers). Also, I do not use the value of semaphores instead when the valid condition enters, I increment the num_a or num_b like a decrement in the sem_wait.

<u>int main(int argc, const char * argv[])</u>

Firstly, in main I get inputs for console and checks them. If they wrong, main terminates.
Then I use srand() to get random sequence for fan to get in car.
And initialize my mutex and barrier by

```
pthread_mutex_init(&lock_t, NULL);
pthread_barrier_init(&barrier_t, NULL, 4);
```

I used barrier to limit 4 fans because if 4 fans are valid (checks in the function pthread_), I need to print spot statement after looking car statement and then print captain statement. In that case output might be like 4 looking, 4 spot,1 captain or 5 looking, 4 spot, 1 captain (1 fan need to wait for next).

Then I create my thread array and t_args array, and initilaze sem_t for each team.
Implement a while loop with rand to make random sequence for my t_args array

Then creating pthread by using pthread_ func and join pthreads after.

And Finally, I destroy barrier and sems and terminate my main.

<u>void * pthread_(void * args)</u>
In my function I take arguments and they casted to arg by

struct thread_ * arg = (struct thread_*) args;

And I create Boolean to check whether car is full.
After that I lock my mutex and print the looking statement because if I does not lock mutex, other threads might be overlap or interfere.
Then I check the team a or team b and increment the num of fans waiting. And then checks the conditions.
First I check the team again because I going to use sem_t and num according to the team.
Secondly, I check if there is valid combination of fans such as 4-0 or 0-4 or 2-2. If yes, then I sem_post the current team which provides increment the sem value(I did not use) and signal the sem to make thread sem to continue which is stuck in the pthread_condition. Then make the bool = 1 to select captain and confirms the car is full.
Then I need to reset the numbers of fans that are waiting so I decrement the fans num according to the team. Else, in no case, thread goes into else part and unlock the mutex to other thread to continue and sem_wait the current thread sem which provides current thread sem to wait(get in pthread_condition). For instance,
A B A B comes and t1 lock and print looking, then goes into sem_wait
T2 and T3 also do the same because there is no valid configuration. Then t4 (B) comes and get 2-2 case, make sem_post statement and make car = 1, then print spot. After that other 3 waiting threads are going to released because t4 signal in post statement. Remaining threads going to print spot and due to car = 1 last person(thread) prints the captain statement and destroy barrier and init it to be ready for next 4 combination. Finally unlock mutex and return to main.