Operating Systems and Networks

Assignment 5 - Question 2

Contents

The Clasico Experience

1) Entities

1.1) Person

- Each person has a `struct`:

```
typedef struct Person
{
    pthread_t thread;               /* The thread of the person */
    char name[PERSON_NAME_LEN];
    char team;
    llint patience_time;
    llint group_number;
    llint arrival_time_delay;
    llint goal_threshold;
    Zone *zone;                     /* The zone in which the person gets a seat */
    bool finished_watching;         /* If the person has finished watching the match */
}
Person;
```

1.2) Group

- Each group has a `struct`:

```
typedef struct Group
{
    llint id;                   /* The group number */
```

```
    llint num_people;
    Person **people;
    llint num_not_at_gate;      /* The count people who have not reached the exit gate */
    pthread_mutex_t lock;       /* Lock to modify the num_not_at_gate variable */
}
Group;
```

1.3) Team

- Each team has a `struct`:

```
typedef struct Team
{
    char name;
    llint goals;                /* Number of goals the team has scored */
    pthread_mutex_t lock;       /* To modify the goals and for the cond var of goal */
    pthread_cond_t goal_cond;   /* The cond var related to the goals of the team */
}
Team;
```

1.4) Zone

- Each zone has a `struct`:

```
typedef struct Zone
{
    char name;
    llint capacity;
    sem_t seats_left;
}
Zone;
```

1.5) Goal Chance

- Each goal chance has a `struct`:

```
typedef struct GoalChance
{
    pthread_t thread;           /* The thread that will simulate it */
    char team;
    llint time;
    double probability;

    bool (*is_successful)(struct GoalChance*);  /* Function to see if the goal was success */
}
GoalChance;
```

2) Threads

2.1) Person

- Each person has its own thread to simulate it.
- First of all it sleeps for the time it takes the person to reach the stadium.

2.1.1) Finding a seat

- I have ran a while loop till the person does not get a seat in the zones they can go and their patience time has not run out.
- If its patience time runs out, it leaves the stadium and waits for the friends at the exit gate
- Otherwise they try to find seats in the zones that they can go by using the `sem_trywait` on the semaphore whose value is the number of seats available in the zone.
- If they do not get seat in any of the valid zones, then they go and wait on a global semaphore. This semaphore was initialized with a value of 0 and whenever a person leaves a zone, a seat gets free and we `sem_post` on this global semaphore. **This allows people to check for seats again and prevents any busy waiting.**
- For waiting on the above semaphore, we use the function `sem_timedwait` which allows us to come out of waiting when the patience time gets elapsed. So, when the time gets elapsed, we continue to the exit gate without watching the match.

2.1.2) Watching the Match

- If the person is neutral to any team, then the thread just sleeps for the spectating time and then moves on the exit gate after it.
- Otherwise, the person watches the match until the spectating time gets over or the opposition team goals more goal than a given threshold of the person.
- For this, we wait in a while loop.
- We break out of it when either the spectating time gets over (by checking the `person->finished_watching`) or when the goals of the opposition team exceeds the threshold.
- As the number of goals of opposite team is a critical variable, we acquire lock before the while loop and in the while loop, we do a `pthread_cond_timedwait` on the condition variable related to the goals of the opposition team (`team->goal_cond`) with the time being the spectating time.
- If we come out of the conditional waiting due to timeout, we set the `person->finished_watching` to true and thus the person moves on to the exit gate.
- Otherwise, it wakes when the opposition team has scored a goal (broadcast-ed from another thread) and it then checks if goals have passed the threshold. If they have, then it breaks out of the loop and frees the lock. Otherwise, it starts waiting on the conditional variable again.

2.1.3) At The Exit Gate

- At the exit gate, the person acquires the lock of the group to which it belongs and decreases its `group->num_not_at_gate` variable.
- If it reaches 0, it means that all the people from the group are at the gate and hence they all leave for the dinner.
- Otherwise, it just exits the thread function

2.2) Goal Chance

- Each goal chance has its own thread.
- It first sleeps till the time for the goal chance has reached.
- Then it uses the `goal_chance->is_successful()` function to see if the goal was successful.
- If its not successful, it just exits the function.
- Otherwise, it acquires the team's goal lock, increases the value by one and then broadcasts on the team's goal conditional variable `team->goal_cond` to tell the people threads watching that the opponent team has made a new goal.