PA-3 REPORT

Buse Gündoğar
27931

In this programming assignment, I tried to implement the solution method given in the assignment description pdf, which was formed of:

- one semaphore for entering & leaving conditions.
- two counters for participants that are waiting for their demo groups and a mutex to protect their atomicity.
- two semaphores for blocking/waiting & waking up for demo sessions.
- one barrier for making sure 2 students and assistant participated in their demo's session.

The flow of my code is:

- In the beggining, necessary imports are written for threads, semaphores etc. Then, I initialized some global variables which will be used from both assistant and student threads. Since they will be updated and used from both of these functions, I implemented them globally. They are consisting of:
    - 2 threads list (for students and assistants), 1 mutex for protecting atomicity, 1 semaphore for controlling the entrance and leaving from the classroom, 2 semaphores for controlling demo sessions for students and assistants, 1 barrier for checking if 2 students and 1 assistant is in a demo session and 2 counters for counting the number of students/assistants waiting to attend demo sessions.
- In my **main thread**, first I declared my program's aim. Then, I stored the number of students and assistants which were entered from the user with the "./demosim num1 num2" command. I checked if these numbers are positive or not. If any of them are negative, I terminated the program. Similarly, if student number is not 2 times the number of assistances, the program again terminated. After these checks, I wrote the necessary initializations for semaphores and threads. Then, I wrote 2 for loops to create student threads and assistant threads. Similarly, I added another 2 for loops for students and assistants to join their threads to wait executing their operations. After all threads finished their operations, main prints its final statements and end the program by returning 0.

- In my **student thread**, students firstly prints that they want to enter the classroom and this request is updated by the "availability" semaphore. (In the assistant thread, whenever a new assistant thread comes to classroom it opens up space for 3 student threads to come into classroom by using "sem_post" command. Whenever this commands are called, waiting students can enter the classroom.)
- Then, again in student thread, whenever student can enter the classroom, it firstly prints it statement which says that the student entered the classroom. Then I locked

my atomicity thread before entering to if/else conditions which updates the counters. Since these numbers are reachable from also other threads, I needed to use mutex before the if/else statements.
    - In the if statement, I checked if there are already 1 student and 1 assistant. If so, we can wake them up and start a demo session. If not, we should update the student counter and also let the current student sleep until the demo group can be formed. If the student enters to the if statement, I decremented the number of student and assistant by 1, since they will not be waiting anymore and will be attending to the demo session. To let them get into demo session, I used semaphore post statement for both student and assistant, meaning that they are available to attend the demo.
    - At the end of both if and else, I unlocked the mutex so that program can continue execution.
- If the student wakes up, it continues to the part where it prints that it is attending to the demo. Then I put the barrier-wait statement there. It is put there since I needed to check if 3 people (2 students and 1 assistant) arrived in demo before ending the demo session. If 3 people condition is satisfied, the barrier is passed and the student can print that he/she is leaving the classroom.
- Before leaving the student thread, I also put a "sem-post" for the availability in the classroom, since 1 more student can get into classroom after the current student.

- In my **assistant thread**, it firstly prints that the assistant is entering the classroom. In the homework document, it says that the number of students can be maximum 3 times the number of assistants. To satisfy this, whenever a new assistant thread starts, it lets 3 students enter the classroom by calling sem-post for the availability of the classroom.
- Then, just like the student thread, before the if-else statement, I locked the mutex to ensure atomicity. This time, if statement checks if there are already 2 students waiting for a demo session. This time assistant number can be 0 or more, since we will be already in an assistant thread and yet not changed the waiting assistant number. This means that there is already at least 1 assistant in the room, so we can start the demo session. In the if part, student counter is decreased by 2 and 2 sem-post call for student is written. In this case, they will be available to attend the demo session. If we get into else part, it means that there is not enough students to start the demo session. So, the assistant will sleep and because of that the counter of assistant is increased by 1. At the end of both if and else, the mutex is unlocked.
- If the student passes this if/else, it means it is awake and can attend to demo. So, it prints that it is participating to the demo. Similar to the student thread, I put the barrier here to wait for 3 people condition for the demo to be satisfied. If so, it again continues and prints that demo is over. Since the assistant will be leaving after that point, I put 3 sem-wait for student threads, since we had the condition that the student number must be maximum 3 times the assistant number. After that, assistant prints that it is leaving.