Project 2: Application for Multithreaded Synchronization

CECS 326: Operating Systems

Fall 2020

October 15, 2020

Sergio Vasquez 018663448
Kuldeep Gohil 015499534
sergio.vasquez01@student.csulb.edu
Kuldeep.Gohil@student.csulb.edu

Output for Project 2 (assuming 3 students and room capacity of 2)

```
kuldeep@kuldeep:~/Desktop/CECS 326/lab2_cecs326$ ./project2.o 3 2
Student 1 shows up in the office
Student 2 shows up in the office
Student 2 asks a question
Professor starts to answer question for student 2
Professor is done with answer for student 2
Student 2 is satisfied
Student 1 asks a question
Professor starts to answer question for student 1
Professor is done with answer for student 1
Student 1 is satisfied
Student 1 asks a question
Professor starts to answer question for student 1
Professor is done with answer for student 1
Student 1 is satisfied
Student 1 leaves the office.
Student 0 shows up in the office
Student 2 asks a question
Professor starts to answer question for student 2
Professor is done with answer for student 2
Student 2 is satisfied
Student 0 asks a question
Professor starts to answer question for student 0
Professor is done with answer for student 0
Student 0 is satisfied
Student 0 leaves the office.
Student 2 asks a question
Professor starts to answer question for student 2
Professor is done with answer for student 2
Student 2 is satisfied
Student 2 leaves the office.
```

Overview:

In this project, we used the knowledge we gained from the previous project about multithreaded synchronization and applied it to a real-life scenario. Specifically, we created a code to synchronize a professor and his/her students during office hours. Unlike the previous project with two parts, this project was more difficult to separate between two people. So, Sergio created the code and Kuldeep helped troubleshoot it.

Design:

We implemented many different functions such as QuestionStart(), QuestionDone(), AnswerStart(), AnswerDone(), EnterOffice(), LeaveOffice(), Student(), and Professor().

- QuestionStart() function is used to take the student ID and waits until it's the student's turn to ask the question. When ready, it prints, “student x asks a question.”

- QuestionDone() function waits until the student is satisfied with the answer to the most recent question and prints “student x is satisfied.”

- AnswerStart() function allows the Professor to start answering a question for the student. When ready, it prints “Professor starts to answer question for student.”

- AnswerDone() function waits until the Professor is done answering a question and then prints, “Professor is done with answer for student x.”

- EnterOffice() function allows students to enter the office of the professor. When possible, it prints, “student x shows up in the office.”

- LeaveOffice() function allows students who no longer have any questions to leave. When ready to leave, it prints, “student x leaves the office.”

- Student() function creates a thread that represents a new student with a unique ID. By using a unique ID for each student, it allows other functions to work properly.

- Professor() function creates a thread that loops calling AnswerStart() and AnswerDone() functions to answer questions students ask.

Using these functions, we were able to create a real-life synchronized application of a professor and students during office hours. This project allowed us to improve our understanding of multithreaded synchronization.