CS232 Operating Systems
Assignment 03: Concurrency and Synchronization
Design document

Name: Muhammad Munawwar Anwar
ID: ma04289

November 22, 2020

Task 1 - TA's office has a maximum of 3 seats

Total_Student is a semaphore which is initialised with the value MAX_SEATS. Whenever class_pfun_enter() or class_os_enter() is run, sem_wait(&Total_Students) is called which decrements the value of semaphore Total_Student by 1. Whenever class_pfun_leave() or class_os_leave() is run sem_post (&Total_Students) is called which increments the value of Total_Student by 1. When the value of Total_Student is 0, any student who wants to enter the TA's office will have to wait until the value of Total_Student is > 0 and the student thread is enqued on a queue. Thereby, ensuring that no more than 3 students can enter the TA's office at the same time. Whenever the the value of Total_Student is > 0, a student thread is dequed and the student enters the TA's office.

chair_Lock is a semaphore which is initialised with a value of 1 . The semaphore chair_lock acts as a central lock which must be acquired by a thread before it updates the values of students_inoffice, class_pfun_inoffice, class_os_inoffice and students_since_break. After updating these values, the thread releases the lock. chair_lock prevents a race condition from occuring by ensuring that only one thread enters the critical section at one time.

Task 2 - There are no PFUN and OS students in the TA's office at the same time

os_cv and pfun_cv are semaphores which are initialised with the value 0. Both os_cv and pfun_cv act as conditional variables. When OS students are in the TA's office pfun_cv causes the PFUN threads to sleep and when PFUN students are in the TA's office os_cv causes the OS threads to sleep. Therefore ensuring mutual exclusion between the OS and PFUN students threads. os_flag is True when OS students are in the TA's office and False when PFUN students are in the TA's office. pfun_flag is True when PFUN students are in the TA's office and False when OS students are in the TA's office. Both os_flag and pfun_flag are initialised as False. pfun_num_students_waiting keeps track the number of PFUN students that are waiting when the OS students are in the TA's office and os_num_students_waiting keeps the track number of OS students waiting when PFUN students are in the TA's office.

When the class_pfun_enters() runs it acquires chair_lock and then checks if os_flag is set to True. If os_flag is False, then it sets pfun_flag to True and updates values accordingly. However, if the os_flag is True, class_pfun_enters() calls sem_wait(&os_cv) which causes PFUN student thread to sleep.But before calling sem_wait(&os_cv), class_pfun_enters() releases the chair_lock and increments the value of pfun_num_students_waiting by 1. By releasing the chair_lock,class_pfun_enters() ensures that there is no deadlock. When the class_os_leave() runs, it checks that if the class_os_in_office is equal to 0 and os_flag is set to True. If this is the case, then class_os_leave() calls `sem_post(&os_cv)` `pfun_num_students_waiting` times to wake all the sleeping PFUN student threads. When `class_pfun_enters()` runs, it re-acquires the `chair_lock` first and then it checks that students since break is less than 10 and the number of students in office is less than 3. If this is the case the PFUN thread will update values or else it will wait.

When the `class_os_enters()` runs it then acquires `chair_lock` and checks if `pfun_flag` is set to True. If `pfun_flag` is False, then it sets `os_flag` to True and updates values accordingly. However, if the `pfun_flag` is True, `class_os_enters()` calls `sem_wait(&pfun_cv)` which causes OS student thread to sleep.But before calling `sem_wait(&pfun_cv)`, `class_os_enters()` releases the `chair_lock` and increments the value of `os_num_students_waiting` by 1. By releasing the `chair_lock`,`class_os_enters()` ensures that there is no deadlock. When the `class_pfun_leave()` runs, it checks that if the `class_pfun_in_office` is equal to 0 and `pfun_flag` is set to True. If this is the case, then `class_pfun_leave()` calls `sem_post(&pfun_cv)` `os_num_students_waiting` times to wake all the sleeping OS student threads. When `class_OS_enters()` runs, it re-acquires the `chair_lock` first and then it checks that students since break is less than 10 and the number of students in office is less than 3. If this is the case the PFUN thread will update values or else it will wait

Task 3 - TA takes a break after helping 10 students

`ta_break` is a semaphore which is initialised with the value `TA_LIMIT`. Whenever `class_pfun_enter()` or `class_os_enter()` is run, `sem_wait(&ta_break)` is called which decrements the value of semaphore `ta_break` by 1. When the `ta_thread()` runs, it acquires `chair_lock` . Then it checks if `students_since_break` is equal to `TA_LIMIT` and `students_in_office` is 0. If this is the case, then it calls `sem_post(&ta_break)` `TA_LIMIT` times to wake all sleeping threads who were waiting while the TA was taking a break. When the value of `ta_break` is 0, any student who wants to enter the TA's office will have to wait until the value of `ta_break` is > 0 and the student thread is enqued on a queue. Thereby, ensuring that the TA takes a break after 10 students. Whenever the the value of `ta_break` is > 0, a student thread is dequed and the student enters the TA's office.