Project 2 Explanation
Joshua Spisak
03/06/2020

While I do feel I could have made a more elegant solution. I still feel that my solution is fair as it does no busy waiting, does not have any deadlocks and fulfills the requirements in terms of tour guides present in the museum and visitors per tour guide. It also does not get the value of semaphores which is nice.

A semaphore is used to track tour guides present in the museum (tour_guides) as well as the number of slots available for visitors (visitor_slots). This guarantees that no more than the designated amount of visitors can be in the museum at a given time as well as makes it easy for visitor and tour guide processes to be woken up when there are more slots available.

There are variables used to track the remaining visitors and tour guides (remaining_tour_guides and remaining_visitors). This is so that when there are no visitors the tour guide processes do not wait for visitors to arrive and exit instead, and when there are no more remaining tour guides the remaining visitor processes can just exit. This is one example of deadlock prevention.

The tour guides opening region is protected by a semaphore so only one process is opening at once, if there are no visitors present in the museum then they signal that they are pending an open and wait for the first visitor to trigger the semaphore. Once the visitor has trigger the tour guide process ups visitor slots allows visitors to proceed.

The tour guide leaving region is also protected such that the last visitor in the museum triggers the tour guide process so that they can leave and the next tour guide process can also leave since there are no more visitors active in the museum. Something interesting about this is that the end of the functions ups the semaphore protecting the leave region (closeing_sem) before it ups the semaphore that counts the number of tour guides active (tour_guides) this seems to allow the other tour guide waiting to leave to run before the next tour guide that wants to enter. If the next tour guide was able to enter before the other exiting tour guide, then that might allow more visitors to enter then blocking the exiting tour guide from exiting. I have some code in the openMuseum() function to prevent I think would work, but was unable to produce errant behavior in the first place so was unable to test. My intuition tells me that the ordering of the semaphores shouldn't really give me any guarantees, but eh, I can't produce errant behavior to test so.. I don't think I particularly care. But if I should I have some code written for that case.

Another region that I put protection on is the visitorArrival, I made it such that only one visitor process at a time could run arrival logic and take a slot. This is because I didn't always want to down on visitor_slots while holding the blanket semaphore (general_state_sem) (hopefully for obvious reasons to prevent deadlock). But this in itself allowed the visitor_slots_available variable to loose sync with the semaphore since it could be downed without the same protection visitor_slots_available had. Only allowed one visitor to run entry logic at a time prevents this deadlock.

Another particularly cool thing I've written is the ability to use pthreads semaphores instead of the cs1550_sem implementation. This was only for testing on thoth, I can compile with the flags `-D USE_PTHREAD -lpthread` and test on Thoth directly. In all the issues I needed to debug and problems I had I got the same behavior on qemu and Thoth which was pretty cool.