Vinna Gu

Professor Khattab

CS1550: Introduction to Operating Systems

23 October 2020

Project 2

In order to assure that each tour guide had served at most 10 visitors, I created a variable called servedVisitors that would count each time a visitor was served or not. If the number of visitors was at least 10, the tour guide may leave after all the visitors left, or if there was another tour guide inside, it would wait for that tour guide to serve at most 10 visitors and leave together.

To ensure that my program would be deadlock free, I made sure that I had a lock that would down() around every shared variable to protect the critical section, and when I was complete, I would up() that lock to release it so another process may use the critical section. When I need to down() on a semaphore to make them wait such as my museumRoom, tourGuideRoom, and visitorRoom, I would first unlock my lock semaphore (which is essentially the mutex), allow them to wait and immediately following that, I would re-lock it using down() so that the process running before the wait may continue using the shared resources. This also prevents a process from simultaneously holding a resource and waiting for another one that’s being used by another.

To avoid starvation, I made sure the current process’ resources had the right resources before doing a wait to wait until another process to release them; otherwise, they might be waiting without the right resources. Also, to avoid starvation, I made sure not to have more than one lock/mutex.