Comp2240 - Assignment 2

Cody Lewis c3283349@uon.edu.au

October 4, 2019

Sharing the Bridge

This program was very simple to implement, in that it only required a single semaphore indicating whether a farmer may cross the bridge. In order to ensure that the implementation is fair, I had to initialize the semaphore as a hard semaphore. The result runs very quickly with the output, however, I figure that may be checked by doing something like piping the output into `less`.

Ice-Cream Time

This program turned out a bit more difficult to implement, especially since I initially tried to implement it as a discrete time event simulation. The discrete time event approach turned out to be unsuccessful due to requiring concurrent access and creating messy interactions. This lead my to use real time involving sleeps and a timer, I timed in terms of a quantum parameter which I determined the value of by testing the program at the maximum capacity on the ES409 lab environment. I was able to be certain of maximum capacity as the program only needs to be able to handle 4 concurrent threads running at any given time, hence, I only needed to give sufficient time to handle that.

A major factor in making this implementation work correctly was the ordering of locks. I had to use semaphore to lock access to the seats, time, and the line of customers, these locks ensured mutual exclusion of the operations in the seat placement algorithm. In order to implement the extra rule for placement, where no customers will be served when all 4 seats were taken until all are emptied, I had to add an extra semaphore which locks away the ability to take a seat. For this to work I had to implement a method of keeping track of the amount of available seats and locking when there are none, then unlocking when they are all available.

Hot or Iced Coffee?

In order to implement the algorithm specified, I decided to take what I had learnt from the previous problem in using a timer, and sleeps. However, this time, I managed locking through the use of monitors this time as per the Assignment specifications. In order to give ordering of client arrival, I have them be placed in a queue, this would be the line for the coffee machine. Then, I also gave the coffee machine a priority queue of event times, this gets used by giving the coffee machine a reference where it can determine the amount of time to sleep before the next event occurs in the system. I found that a required a secondary parameter for time units in the case of this program, epsilon, which allows for the distinguishment of multiple simulataneous events without clashes, or incrementing to the time.

An important element of getting this program to work correctly was the ordering of waits and the locks. I found that to ensure the maintainance of the order of the serving of the clients, I had to lock the queue of clients, then while that is locked performing the check for notifying the client that the stop waiting. I decided to have an additional wait contained in the serving of clients in the case of a mismatch of hot and cold between what is being served currently and what the client wants.