Homework 3 - Unisex bathroom

Ludvig Larsson

February 17, 2022

1 Implementation

The implementation was quite straight forward. It has its roots in readers writers problem shown during the lecture on semaphores. I took inspiration from there and started the implementation. Men and women have different functions to execute in the program. The are almost the same except for some variables used that have different names. This could have been generalized but I thought it was easier to read the code than to use conditional statements throughout the function to differentiate between the genders.

Execution for an arbitrary gender. Check if anyone of the other gender is currently in the bathroom, if not, check if there are any people of same gender waiting to enter the bathroom and let one in. Use the bathroom and afterwards check if there are any people of the same gender still in the bathroom. In case there are none, i.e. they're "at work", signal a women to enter the bathroom. This is of course synchronized by semaphores to achieve correct behavior. If a man cannot enter the bathroom due to women already being inside the man stands in line with a semaphore that counting to 1. To check how many of opposite gender waiting for, and being inside, the bathroom a semaphore is used as a mutex.

This implementation described does not ensure fairness. Based on delay for using the bathroom and working, one gender could easily occupy the bathroom until all are done with their visits (command line argument). To ensure fairness I implemented a time keeping mechanism. If the person entering is the first of their gender the time starts, when subsequent people of same gender is checking the a valid bathroom state to allow entering, the time keeping could be overdue and therefore put them to wait in line instead. This prevents one gender from occupying the bathroom and therefore fairness is ensured.

2 Discussion

Execution of the program is good. There are no dead locks discovered over the approximate 100 times I have ran the program. This doesn't assure that the program is free of dead locks but the logic in the program seems reasonable to me so I would say that it cannot happen, fingers crossed.

One thing I got stuck on after writing the program and ensured that it was fair was the possibility of it being unfair. Namely the execution trace (printouts) was identical with and without the additional fairness implementation. The execution trace always shows that all men threads enters the toilet once, then all women does the same and the genders take turns until they're done with their total visits.

This is most likely because my original code follows the flow: `work --> enter bathroom --> allow women if bathroom empty --> loop`. Let's say there are 2 men. First man uses bathroom for a short amount of time then works a short amount of time while the second man takes long to use the bathroom. This leads to the first man entering the bathroom again since second man is still in the bathroom (condition if bathroom empty is false). This is impossible by my calculations though (fairness ensured already). The rules of the program says that delay for work should be longer than time to use bathroom, for example: work 2-3 seconds and bathroom visit 1-2 seconds. Therefore, `delay(bathroom minimal = 1) + delay(work minimal = 2) < delay(bathroom maximal = 2)` is always false and the program cannot be unfair with original implementation.