Homework 3

Jacob Hedén Malm
9804051499
jacmalm@kth.se
Concurrent Programming

For this homework assignment I chose to tackle task 4, the unisex bathroom problem. I modeled my solution after the solution to the readers writers problem presented during the course lecture. In the main method, I read user input to determine how many male and female workers (threads) I spawn, in the form of argv[1] and argv[2]. All of the male threads are then sent to mWork and the female threads are sent to wWork. These two functions act in the same way, but as mirror images of each other, with respect to global counters/semaphores.

The program state is kept track of using 4 global counters, and signalling is done using 2 semaphores, with a 3rd semaphore acting as a mutex lock. The 4 global counters keep track of how many women/men are using the bathroom currently, and how many women/men are in line to use the bathroom currently. Both the 2 signalling semaphores are initialized to 0, and they are used to delay processes in line to use the bathroom in a FIFO queue.

As a process enters the work method, a random time in the range of 0 - 59 seconds is generated for the process to sleep. This signifies “work” time. Then, when the process wakes up, it wishes to go to the bathroom. The first check that is done is whether or not there is someone of the opposite gender currently using the bathroom, or if there is someone of the opposite gender waiting to use the bathroom. If either of these two cases are true, the process joins its queue to use the bathroom. The first clause is self explanatory, men cannot use the bathroom at the same time as women. The second clause is to ensure fairness, we do not want a continuous stream of men into the bathroom to starve out the women in line to use the bathroom, or vice versa.

When a process joins the queue to use the bathroom, it releases the mutex which was grabbed to ensure that only one process changes the global state of the program at the same time, and then waits for the signalling semaphore of its gender to be incremented.

If the process does not join the queue, we know that it can use the bathroom. Thus, the global counter is incremented. We also check if there are processes of the same gender in line to use the bathroom, if this is true, it means that the current process has exited the queue due to a signal from the opposite gender stating that they have all evacuated the bathroom, or due to a process of the same gender that was further ahead in the queue, and thus we know that the current processes gender should be allowed to use the bathroom, and so we allow all processes to leave the queue and enter the bathroom. This is done by incrementing the signalling semaphore, which is a lock for the queue to the bathroom. As such, if this is incremented by one process, the first process in the queue is “free”, and will immediately enter

the same check if there are other processes in queue, and this process will in turn increment the semaphore, and so on until the queue is empty. For each process that exits the bathroom, we also decrement the queue counter.

If it is the case that there are no processes of the same gender in the queue, this means that the current process asked to use the bathroom when there was nobody in the bathroom, and nobody in line to use the bathroom. This also means that the mutex has not yet been released, as the only time this is done is right before a process joins its genders queue. Thus, the mutex is released now. The process then uses the bathroom, this is a separate function.

The bathroom function generates a random time in the range of 0-4 seconds, and sleeps the process for that time. It then returns.

As a process returns from using the bathroom, it decrements its counter keeping track of people of its gender currently using the bathroom. It then checks if there are workers of the same gender in the bathroom, and if there are workers of the opposite gender in line to use the bathroom. If there are no workers of the same gender in the bathroom currently, this means that the current process was the last of its kind to exit the bathroom. If there are processes of the opposite gender in line to use the bathroom, this means that a gender switch should occur. This is done by incrementing the opposite genders signalling semaphore, which will “free” one process of the opposite gender, which will in turn empty the line by incrementing its semaphore and so forth until the queue is empty. As such, fairness is guaranteed. All of this is done in an infinite while-loop, so that the simulation keeps going until it is manually cancelled by the system.