Concurrent Programming Homework 3

Marcus Samuelsson

February 2023

1 Unisex Bathroom Problem

The code is built up of three functions **work()** and **bathroom()**. Firtly the three semaphores are initiated using **sem_init** with all having value 1 so each can be accessed by one thread at a time. Each thread is then started using **pthread_create** in the work function with a struct containing their id, their number according to sex and their sex as the parameter. The number of threads are set as the input **argv[1]** multiplied by two to create men and women. When the random time set using **rand()** has passed the thread enters the queue by calling the **sem_wait** for the bathroomInQueue sem_t. Then it checks what the gender of the current person(s) is and then either enters or waits in a while loop until the bathroom is free or the person can enter. Because they wait after calling **sem_wait** the queue will start to build up after them according to no matter if it is a man or a woman, so that everyone goes follows a fair order. When a person enters the bathroom they call **sem_post** for the bathroomInQueue and allowes the next person to check if they can go in.

In the bathroom the person increases the integer **inBathroom** which tells the people in queue how many are still left in the bathroom and waits for a random time to simulate the bathroom break. This is increased when they enter and decreased when they exit. To make sure that **inBathroom** functions correctly as the threads do write to it, another **sem_wait** and **sem_post** is called for the around the code modifying the variable. All of this is run in a while loop that runs forever, making it so the threads rotate working and bathroom breaks.

Lastly, to make sure that the text is printed correctly in the console the function **print()** was created. This function takes in a string and the uses a semaphore(writeLock) to make sure that each text is printed correctly.