 To synchronize the processes using semaphores I used 7 semaphores with 4 of them being flags for expressing the current execution of the agent and smokers. Then I used another semaphore, called lock, to act as a lock so that no two processes run at the same time. These semaphores were controlled via the P and V functions provided in the sem.h file. Another semaphore was used so that I could keep track of how many times the smokers were smoking and then not perform the processes if they were done smoking and the agent did not place anything on the table. This semaphore might not have been necessary, and I probably could have completed this without it. The other semaphore was used as a flag, I called it Latch, this flag would not print the critical section if it was 0 which only happens after the agent stops.

Image 1: Shows the synchronization for semaphores Image 2: Shows synchronization for pthreads

To synchronize the processes using pthreads, I created 4 pthread ID’s, and 5 mutex locks. 4 of the mutex locks were used for the agent and smoker processes, the other mutex lock was acting like a lock to not let multiple processes run at the same time. This was achieved by locking when entering a process and then unlocking it when we left it. At the same time, I also locked the process we entered so that it will not run the second time. I also unlocked the agent process upon leaving so that the agent process can run again. This was done inside each smoker process. To run the processes, I used the pthread\_create function to begin running the functions. Finally, to wait for the processes to finish, I used the pthread\_join function that will terminate the processes.

Some similarities were unlocking and locking processes when they were or weren’t being used. Also, I used a semaphore flag that acted as a mutex lock (in the case of pthreads). Pthreads used functions instead of forking children, but each one was still ran using the lock flag or mutex that I created. I also had a critical section if statement for both semaphores and pthreads. Overall the two approaches were very similar.