Socratis Katehis

11/28/2017

CSC 332 LAB – Task 6

For my semaphore approach, I decided that I would you five locks to control the four processes. One lock for each process, and another main lock for the three smokers. This had to be done because I did not have function calls for any of the processes and wrote everything in main(). Once I created the semaphores, I created a child for each process and I put locks into the appropriate places. Each child process had a while loop that was always true to simulate an infinite scenario. This approach did not require me to use an array to maintain values.

For my pthread approach, I used one mutex that was locked and unlocked in each of my processes. I only had to write one agent process and one smoker process. I then created three pthreads for the smokers with the implemented smoker process, and one thread for the agent. An always-true while loop was put into each process to simulate an infinite scenario just like the semaphore approach.

My “table” was implemented with an array, and the strings with a character array.

Both approaches printed what the agent put onto the table, and then printed the appropriate smoker that would pick the items up from the table and smoke.

Looking at my semaphore implementation, you can see that the code is longer than the pthread implementation. Although length of code isn’t an important distinction between programs, the pure ease of implementation with pthreads makes it easier to solve a problem like cigarette-smokers. Also, the semaphore implementation required more lock mechanisms. Specifically, I used five locks. One lock was generally used throughout the code to maintain processes from leaking into each other. One lock was used for the agent. The remaining three locks were used for each smoker. My pthreads only required one mutex for both processes (smokers and the agent), so as previously mentioned it was easier to implement.

Besides that, I appreciate the semaphore approach to the problem more than the pthreads approach. With semaphores there is more control in the code (specifying locks with more than just conditions) and is a better representation, in my opinion, of what we learned as far as concurrency in the recitation.