Dalton Meny

CS 470

Thread Synchronization Project Discussion

My program consists of a main function with 4 auxiliary functions. The main function reads in the user input and creates the consumer and producer threads using the respective functions. Those functions use the insert_item and remove_item functions to change the globally defined buffer. There are two unnamed semaphores used in the producer and consumer threads. The first semaphore (sem1) has an initial value of the buffer size and will cause producer to wait if buffer is full. The consumer will post to sem1 after it consumes something to indicate there is one less item in the buffer. The second semaphore (sem2) has an initial value of zero and will cause the consumer to wait if there is nothing to consume. After a producer produces something, it will post to sem2 indicating that there is something to consume. There are also two pthread mutex locks used in this program. The first one, simply called mutex, is used to enclose the in and out variables in remove_item and insert_item to ensure that only one function is accessing and altering the buffer and the indexes at a time so that there is mutual access. The second pthread mutex lock (print_mutex) is used around all print statements to make sure that they are atomic.

Questions

1. I think the hardest aspect of thread manipulation was finding out which pthread functions to use and where to put them. I also found the circular queue a slightly hard to understand but easy to implement.
2. The easiest part to understand about thread manipulation was the basic function of the auxiliary functions. Disregarding the synchronization aspect, they were very simple to understand and create.
3. The hardest part of thread synchronization to understand was finding out which type of synchronization to use where. I especially found the semaphores confusing and how to use them properly.
4. The easiest part of thread synchronization was using the pthread mutex locks. They are very straight-forward and easy to use granted you know where to use them.
5. There are some things I would change in my project. I think it is a little dense and would like to make it more readable. Also, my locks and semaphores may not be in the optimal places, so that is something that I would like to make sure of and possibly change.
6. I found it surprising how simple and straightforward it was making the threads. I also didn’t fully understand the importance of sleep at the beginning of the project. I also found it interesting how much less code is needed to solve this problem using threads compared to the shared memory example in class.