Operating Systems Lab #2

David Tyler

10/23/14

Objective

Learn how to do multithreading and how to synchronize threads using semaphores and mutexes.

Background

The class used the pthreads library for this laboratory. Mutexes can be used to protect critical sections of the code when multithreading. This allows the programmer to not have to worry about other threads modifying the same section of the code at the same time. Semaphores are very similar to mutexes but one can use counting semaphores to allow x numbers of accesses to a resource instead of just one like with a mutex. When doing multithreading, one must be careful to not let the parent thread die until all the children die. pthread_join() can be used to avoid this situation.

Algorithm

The algorithm for this lab was very simple. Once the user specified the number of threads to be created, the program looped that many times and created the requisite number of threads as Consumers. One additional thread was created for the Producer. The producer thread simple continuously fed random numbers into a circular queue. If the queue got full, it would simply wait until more numbers were consumed. The consumer threads would each pop one number out of the queue, print it, and exit. This way, every thread was able to consume a number. It would have been easy to have this program run forever, but that was not required and would have needed priorities to be set high on threads when they are created and low once they have consumed to ensure every thread would get to eat a number. After all the threads were created, the main function joins every consumer thread and waits for them to finish. After the consumer thread finishes, shared memory is destroyed and the producer keeps going as per spec while the main function joins it.

Results

```
#./lab2.exe 3
PRODUCER: INSERT 7590 into BUFFER
PRODUCER: INSERT 1575 into BUFFER
PRODUCER: INSERT 4084 into BUFFER
THREAD 0: REMOVE 7590 from BUFFER
THREAD 1: REMOVE 1575 from BUFFER
PRODUCER: INSERT 899 into BUFFER
PRODUCER: INSERT 9952 into BUFFER
THREAD 2: REMOVE 4084 from BUFFER
PRODUCER: INSERT 7886 into BUFFER
```

```
All threads have eaten!
```

Conclusions

There were no problems with this lab. I hope exiting the threads after only eating one number is ok!

Readme

see README

Source

see lab2.c