Compilation is done using the provided makefile, although only one file is used.

Discussion:

This program aims to solve the producer-consumer issue using pthreads, or POSIX threads. The main idea behind pthread usage is thread creation, resulting in parallel programming opportunities involving multiple computations being done simultaneously on different threads.

The program is comprised of 3 major functions: the Main, producer, and consumer functions. The main function drives the entire program with calls to the producer and consumer functions along the way, and will throw errors if necessary run time parameters (time to sleep, number of producers, and number of consumers) are not provided. The producer and consumer functions work as expected and rely on external “insert\_item” and “remove\_item” functions to handle the actual operations of insertion and deletion of buffer elements.

My code makes use of pthreads for simultaneous production and consumption, and handles synchronization issues via two mutex locks and two semaphores. The two mutex locks, “In” and “Out”, ensure the item being ‘produced’ or ‘consumed’, inserted into the buffer or removed from it respectively, is not tampered with until the operation has fully completed. The two semaphores, “full” and “empty”, protect against buffer overhead. This translates more specifically to preventing production with a full buffer, and preventing consumption with an empty buffer.

Questions:

1. What aspect of thread manipulation did you find most difficult to understand?

*The actual syntax of pthreads and casting of its attribute variables threw me for a considerable loop.*

1. What aspect of thread manipulation did you find least difficult to understand?

*The pthreads work synonymously to threads used for real time user interaction in Visual Studio (C#), so this was easy for me to grasp.*

1. What aspect of thread synchronization did you find most difficult to understand?

*It was difficult for me to understand why two distinct mutex locks were needed in this project.*

1. What aspect of thread synchronization did you find least difficult to understand?

*At this point it is second nature to protect creation and destruction of elements in real time, so that much comes naturally.*

1. What, if anything, would you change in your current design?

*I would reorganize my code as to utilize the suggested header file.*

1. What, if anything, did you find interesting or surprising about thread manipulation or thread synchronization that you did not know before doing this project?

*Just how easily data can be corrupted and the overwhelming importance of proper implementation of synchronization measures.*