CSS 430 Operating Systems
Programming Assignment Report #3
Author: Jay Hennen
11/12/2013

-----> Specification

1. SyncQueue.java
    1. SyncQueue.java maintains an array of QueueNode objects, which act as conditions for threads to sleep by calling enqueueAndSleep(int condition). Child threads which the parent are waiting on can notify and wakeup the parent by calling dequeueAndWakeup(int condition, int tid). The notifying child's tid is stored in each QueueNode and is read by the parent when it wakes up.
2. Kernel.old has implementations for WAIT and EXIT, by using the SyncQueue to put threads to sleep and wakeup.
3. Kernel.java has implementations for RAWREAD, RAWWRITE, and SYNC, eliminating busy-waits by using the SyncQueue instead.
4. TestThread3.java creates either a computation thread or a disk access thread depending on the parameter it is passed
    1. DiskThread() writes or reads a block to a random location on disk with every repetition
    2. CompThread() creates an initial random byte array, then performs modulus operations on the elements sequentially to create a new byte array, which is then re-used in the next iteration.
5. Test3.java creates X pairs of TestThread3s depending on the parameter passed. It will create the threads in pairs and store their TIDs and type “comp” or “disk” in a map, which is also used to ensure all child threads are completed before the program ends. It also provides timing information for each thread as well as the total elapsed time.

-----> Report: Discussions about performance results you got for part 2

1. There appeared to be little difference in performance between the Kernel with or without the SyncQueue synchronizing thread waits on my home Windows machine, the cause of which is uncertain. I assume it must be in the differences between how the Windows version of the JVM operates compared to the Linux version.
2. However, the results on Linux showed a dramatic increase in speed between the two versions of Kernel.java. The total elapsed time was much quicker when thread synchronization was introduced, as well as the turnaround time for each individual thread, because each could be completed more quickly. This is because the busy-waits were removed in the updated version, allowing more thread time to be spent on computation or disk access.