Report Documentation for Program 3

Building files:

For Part 1:
Rename the current *Kernel.java* to *Kernel.new*, and rename the *Kernel.old* to *Kernel.java*. To build *SyncQueue.java, QueueNode.java* simply use the following command:
`javac *.java` or specified java classnames.

To run Test2 for the files, follow the listed steps

1. Import all ThreadOS necessary files into current directory
2. Use `java Boot` to load ThreadOS
3. Use `l Test2` to load Test2 in order to test the output from the modified Kernel.

For Part 2:
Build the following files: *Test3.java, TestThread3a.java, TestThread3b.java*, using:
`javac *.java` or specified java classnames.

Then check if the *Kernel* has modifications to RAWWRITE, RAWREAD, and SYNC. If so, it's *Kernel.new* otherwise it's *Kernel.old*. Build the respective Kernel.

To run Test3 for the files, follow the listed steps

1. Import all ThreadOS necessary files into current directory
2. Use `java Boot` to load ThreadOS
3. Use `l Test3 X` to load Test3 in order to compare performances between Kernel versions. Where X indicates a variable amount of threads. If no integer is placed, a random amount of threads will be utilized.

Program Purpose

For Part 1:
Modify the Kernel from the original WAIT and EXIT cases to utilize a Synchronized Queue (SyncQueue.java) with queue implementations for waking and sleeping threads.

For Part 2:
Compare the versions of busy waiting / spinloops from the *Kernel.old* and the newly modified Kernel with proper sleeping of threads on particular conditions for RAWREAD, RAWWRITE, and SYNC.

Algorithm Discussion for Part 1

SyncQueue.java
Purpose is to hold multiple QueueNodes which have different conditions or functionality. The following methods of: *enqueueAndSleep()* and *dequeueAndWakeup()*, are utilized to sleep and wake threads based on a condition.

enqueueAndSleep() takes in an *int* value as a condition and then invoked *sleep()* from *QueueNode.java* which then returns the tid of the thread that is woken up.

dequeueAndWakeup() takes in an *int* value as a condition and then wakes up the thread that has the condition value. There is another version which has an additional *int* input of a TID (thread ID) to specify which thread to wake-up specifically. By default the DEFAULT_TID value is set to 0.

QueueNode.java
This is a class utilized in *SyncQueue.java* where it has a vector of Integers to represent threads with the same condition. The methods of *wake()* and *sleep()* are used in this class.

wake() Enqueues a thread with the TID into the vector of Integers and notifies the sleep method.

sleep() Sleeps a thread until a notification, returns the TID of the first thread that wakes up the thread calling this method.

Kernel.java
Made modifications to the EXIT and WAIT cases to utilize *SyncQueue.java* and *QueueNode.java*.

WAIT now gets the current running TCB (Thread Control Block) to get the TID to use *enqueueAndSleep()* on that thread.

EXIT also gets the current running TCB and used *dequeueAndWakeup()* instead on the PID and the TID. The thread is then deleted in the scheduler.

Part 1 Result of Test 2 in Shell.class

```
[22:41:19] daniely@uw1-320-07: ~/prog3/ThreadOS $ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Shell
l Shell
threadOS: a new thread (thread=Thread[Thread-5,5,main] tid=1 pid=0)
shell[1]% Test2
Test2
threadOS: a new thread (thread=Thread[Thread-7,5,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,5,main] tid=3 pid=2)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=2)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=2)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=2)
threadOS: a new thread (thread=Thread[Thread-17,5,main] tid=7 pid=2)
Thread[e]: response time = 6999 turnaround time = 7499 execution time = 500
Thread[b]: response time = 3999 turnaround time = 12000 execution time = 8001
Thread[c]: response time = 4999 turnaround time = 25002 execution time = 20003
Thread[a]: response time = 2999 turnaround time = 35003 execution time = 32004
Thread[d]: response time = 5999 turnaround time = 40002 execution time = 34003
shell[2]%
```

Part 2:

3 New classes were created as part of to utilize the modifications to *Kernel.java* in RAWREAD, RAWWRITE, and SYNC which are termed as *Test3.java*, *TestThread3a.java,* and *TestThread3b.java*. This is to compare the spinlock /busy wait performance to the new modifications through tests.

Kernel.java

Simple modifications to the RAWWRITE, RAWREAD, and SYNC cases to use the ioQueue and INTERRUPT_DISK so that CPU may relinquish threads to another and to have the disk be interruptible.

Test3.java

This class is a user level thread maker that creates a specified X number of thread pairs. For each pair it will then run *TestThread3a.java* and *TestThread3b.java* to mimic computation and disk reads and writes. Both classes are run in pairs so they are intertwined. Total time for total execution is timed here. A variable amount of threads can be inputted to run the command, or none for a random selection.

TestThread3a.java

This class is to mimic computations for an OS, as such it uses a recursive Factorial addition through for loops to mimic heavy duty addition computation. Computation time per thread for execution time is calculated here.

TestThread3b.java
This class is to mimic the disk reads and writes for an OS, as such it creates a block for the disk and then reads and writes to it. Writing time per thread for execution time is calculated here.ss

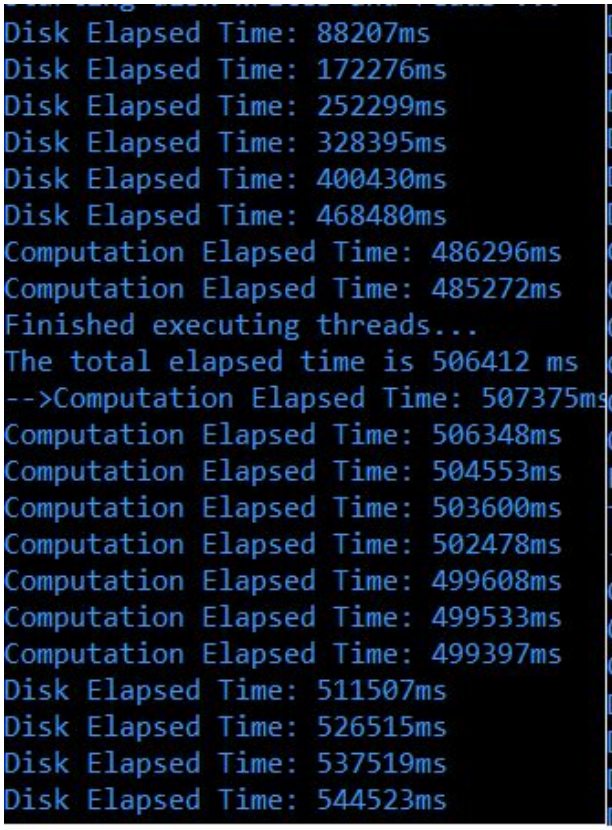
Light Performance Test for *Test3.java* using the new *Kernel.java*

```
-->l Test3 4
l Test3 4
threadOS: a new thread (thread=Thread[Thread-19,5,main] tid=8 pid=0)
threadOS: a new thread (thread=Thread[Thread-21,5,main] tid=9 pid=8)
threadOS: a new thread (thread=Thread[Thread-23,5,main] tid=10 pid=8)
threadOS: a new thread (thread=Thread[Thread-25,5,main] tid=11 pid=8)
threadOS: a new thread (thread=Thread[Thread-27,5,main] tid=12 pid=8)
threadOS: a new thread (thread=Thread[Thread-29,5,main] tid=13 pid=8)
threadOS: a new thread (thread=Thread[Thread-31,5,main] tid=14 pid=8)
threadOS: a new thread (thread=Thread[Thread-33,5,main] tid=15 pid=8)
threadOS: a new thread (thread=Thread[Thread-35,5,main] tid=16 pid=8)
Running Executions...please wait
Starting computations ...
Starting disk writes and reads ...
Starting computations ...
Starting disk writes and reads ...
Starting computations ...
Starting disk writes and reads ...
Starting computations ...
Starting disk writes and reads ...
Disk Elapsed Time: 40178ms
Disk Elapsed Time: 76159ms
Disk Elapsed Time: 108206ms
Disk Elapsed Time: 136203ms
Computation Elapsed Time: 206975ms
Computation Elapsed Time: 205162ms
Computation Elapsed Time: 204571ms
Computation Elapsed Time: 203403ms
Finished executing threads...
The total elapsed time is 212163 ms
```

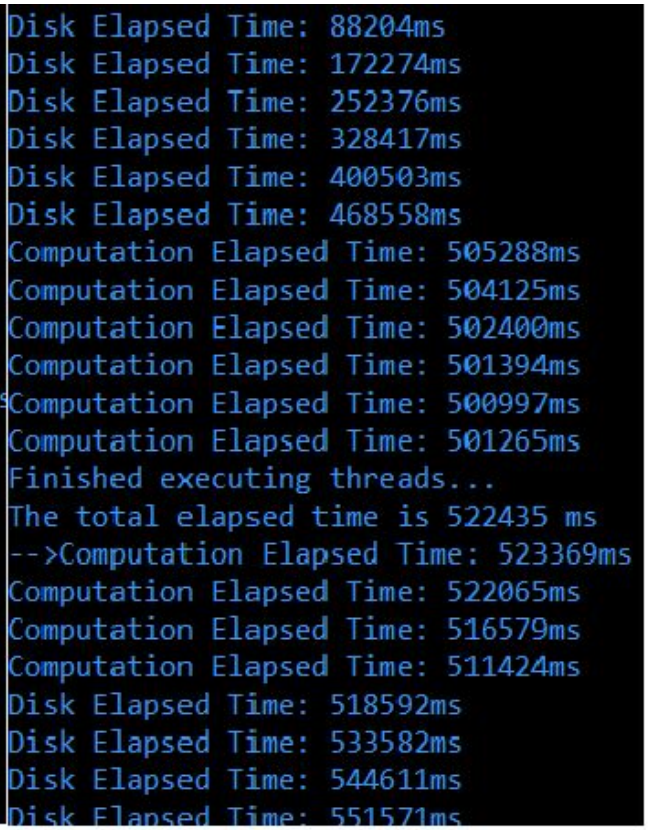
Light Performance Test for *Test3.java* using the old *Kernel.java*

```
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Test3 4
l Test3 4
threadOS: a new thread (thread=Thread[Thread-5,5,main] tid=1 pid=0)
threadOS: a new thread (thread=Thread[Thread-7,5,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,5,main] tid=3 pid=1)
threadOS: a new thread (thread=Thread[Thread-11,5,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,5,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,5,main] tid=6 pid=1)
threadOS: a new thread (thread=Thread[Thread-17,5,main] tid=7 pid=1)
threadOS: a new thread (thread=Thread[Thread-19,5,main] tid=8 pid=1)
threadOS: a new thread (thread=Thread[Thread-21,5,main] tid=9 pid=1)
Running Executions...please wait
Starting computations ...
Starting disk writes and reads ...
Starting computations ...
Starting disk writes and reads ...
Starting computations ...
Starting disk writes and reads ...
Starting computations ...
Starting disk writes and reads ...
Disk Elapsed Time: 40179ms
Disk Elapsed Time: 76250ms
Disk Elapsed Time: 108229ms
Disk Elapsed Time: 136251ms
Computation Elapsed Time: 198561ms
Computation Elapsed Time: 198012ms
Finished executing threads...
The total elapsed time is 206157 ms
-->Computation Elapsed Time: 205737ms
Computation Elapsed Time: 209179ms
```

Performance Test Directly Comparing Kernel.old and Kernel.new



Kernel.new



Kernel.old

Discussion on Performance

It's clear from the direct comparison of performance test, that the *Kernel.new* suffices to run faster than the *Kernel.old*. Where *Kernel.old* has busy wait implementation while *Kernel.new* has the proper modifications to RAWWRITE, RAWREAD, and SYNC.

This is likely due to the *Kernel.new* ensuring that the CPU relinquishes its thread waiting on the I/O so there's always work being done, rather than having idle waits which busy wait performs. As such there is a marginal performance increase due to the additional time not spent just waiting around.