Name: Thuan Tran
CSS 430 Program 3
Date: May 6th,2017

REPORT

Table of Contents

Part 1

For this part, I implemented 2 new classes SyncQueue.java and QueueNode.java . I also made some modification in the Kernel.java as well

SyncQueue.java

This class is used to hold a list of QueueNode that each represent different condition (functionality), This class also provided methods that allow ThreadOS to sleep a certain thread. Or to wake up a certain thread as well. Note that this provided additional benefits toward Java ways of only providing wait() and notify(). In this case, we can sleep and wake up threads that satisfy a certain condition

enqueueAndSleep()

This class takes in a condition and let it sleep by invoking the sleep() method of QueueNode. Note that it will return the tid of the thread that woke it up

dequeueAndWakeup()

This class takes in a condition and wake up a thread that belong to that condition. Note that there are 2 versions of this method. The first version only take in the condition. And the second version take in both the condition and the tid that need to be wake up. Without specifying the tid, we assume the tid will be equal to 0 for the second version

QueueNode.java

This class is utilized in SyncQueue.java to represent a certain condition. The class QueueNode.java have a vector of Integer to represent threads that also have the same condition. There are 2 methods in this class: wake() and sleep that is utilized in the SyncQueue class

sleep()

This method sleep a certain thread until a notification. It will return the id of the first thread that wake up the thread that is calling the sleep method.

wake()

This method enqueue a thread with its tid into the Vector of Integer and notify the sleep method

Kernel.java

In the kernel.java, I made a few modification in the wait and exit case

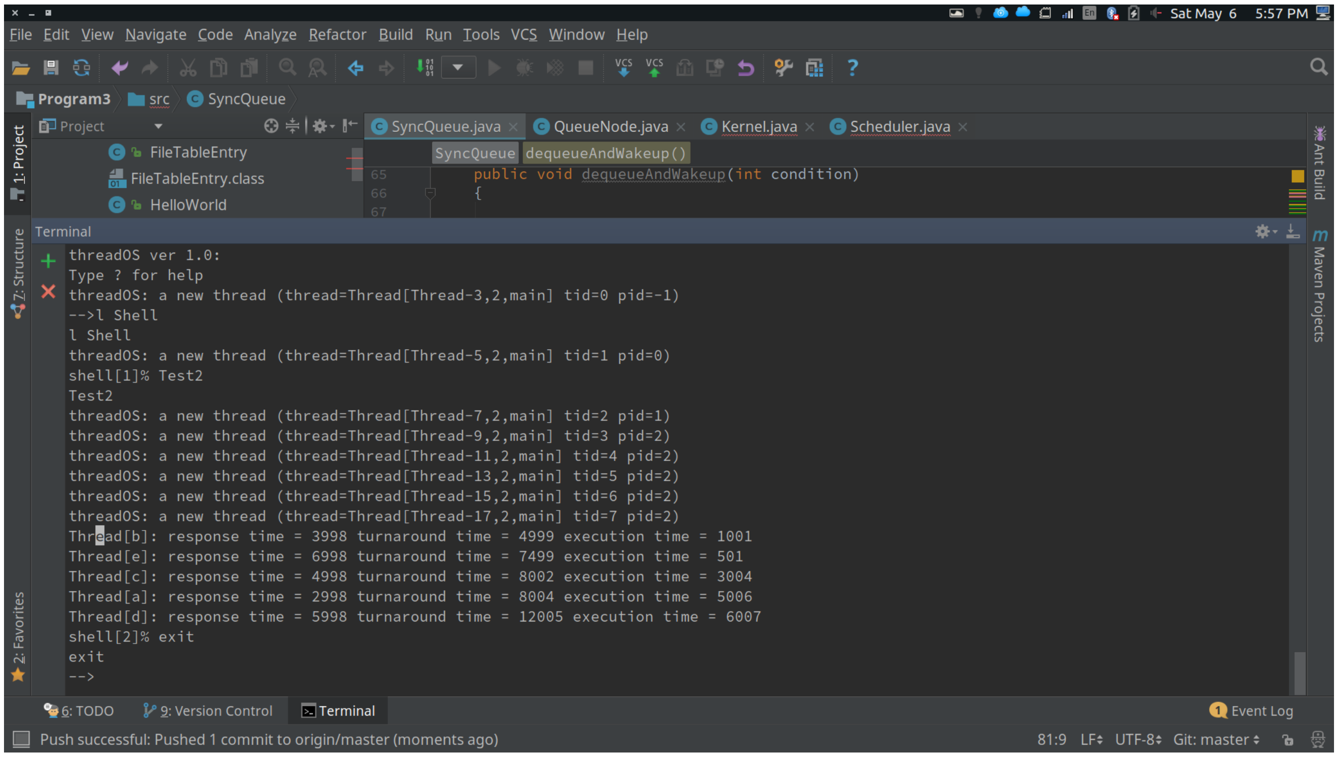
Wait

The wait case first get the Tcb that is currently running and use that Tcb to get the tid of the thread in order to enqueuAndSleep that thread

Exit

The exit case also get the Tcb that is currently running. This case, it dequeAndWakeup and thread by using its pid and its tid. After doing that, it deletes the thread in the scheduler

Result of Part 1



```java
public void dequeueAndWakeup(int condition)
{
```

```
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
shell[1]% Test2
Test2
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=2)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=2)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=2)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=2)
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=2)
Thread[b]: response time = 3998 turnaround time = 4999 execution time = 1001
Thread[e]: response time = 6998 turnaround time = 7499 execution time = 501
Thread[c]: response time = 4998 turnaround time = 8002 execution time = 3004
Thread[a]: response time = 2998 turnaround time = 8004 execution time = 5006
Thread[d]: response time = 5998 turnaround time = 12005 execution time = 6007
shell[2]% exit
exit
-->
```

Push successful: Pushed 1 commit to origin/master (moments ago)

Illustration 1: Output when running Test2.java

Part 2

In this part, I created 3 new classes called Test3, TestThread3a and TestThread3. I also made modification to the RawRead, RawWrite and Sync so that there will be no spin loop anymore.

Modification of RawRead, RawWrite and Sync

Instead of doing the spin loop, I now enqueue and sleep the thread based on certain condition.

Test3.java

This class is a user level thread that will create X paris of thread. In each pair, it will perform a computation and IO

Algorithm

I decided to create a constructor with an String[] array argument so that I can pass in the number of X pairs to be created from the command line

Within the run method, I used the SysLib.toStringarray method to parse in the name of the class that is going to be run. At this point, I also created a variable to hold the current start time. After that, I do a for loop that execute X number of times. Each time execute TestThread3a and TestThread3b

After the for loop for execution, I did another for loop to join the threads. Note that this time I did a for loop of X*2 because X is a pair so there will be 2X threads created

After that, I just get the current time when all the for loop is done and subtracted with the time when the method start and print out the elapsed time

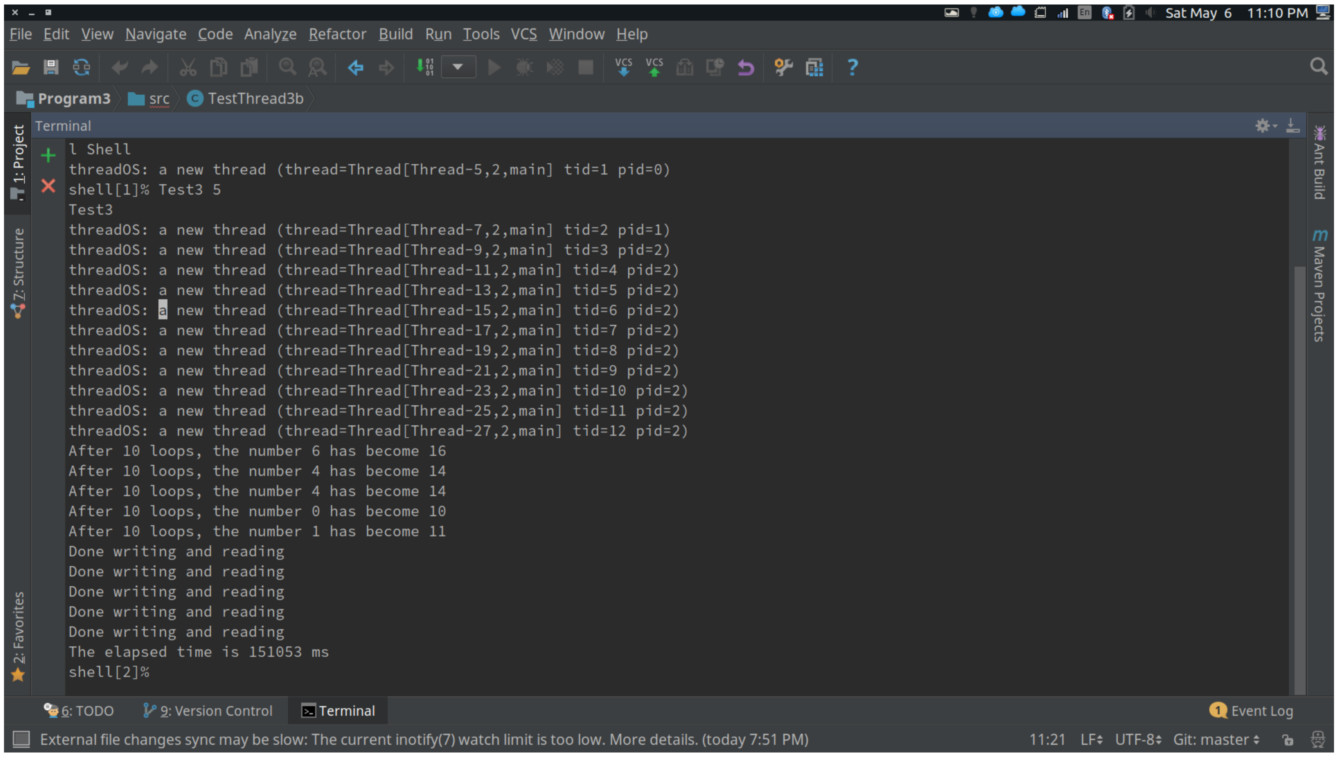
TestThread3a.java

This thread perform a computation. I decided to make it simple by generating a random number frrom 0 to 9 and perform +1 to the number in a loop 10 times. Then I printed out the result after the loop to show that the thread is finished. And exit

TestThread3b.java

This Thread perform a read operation from the disk. It first created a byte array of 512 to hold the data. Then within a loop of 1000 times (because there are 1000 blocks), it read the data from the disk at the index into the byte array. Then it print out a statement that it is done with IO and exit

Result of Part 2



```
l Shell
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
shell[1]% Test3 5
Test3
threadOS: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
threadOS: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=2)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=2)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=2)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=2)
threadOS: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=2)
threadOS: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=2)
threadOS: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=2)
threadOS: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=2)
threadOS: a new thread (thread=Thread[Thread-25,2,main] tid=11 pid=2)
threadOS: a new thread (thread=Thread[Thread-27,2,main] tid=12 pid=2)
After 10 loops, the number 6 has become 16
After 10 loops, the number 4 has become 14
After 10 loops, the number 4 has become 14
After 10 loops, the number 0 has become 10
After 10 loops, the number 1 has become 11
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
The elapsed time is 151053 ms
shell[2]%
```

Sat May 6 11:10 PM
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Program3 src TestThread3b
Terminal
1: Project
7: Structure
2: Favorites
Ant Build
Maven Projects
6: TODO
9: Version Control
Terminal
Event Log
11:21 LF UTF-8 Git: master

External file changes sync may be slow: The current inotify(7) watch limit is too low. More details. (today 7:51 PM)

Illustration 2: New Kernel Result

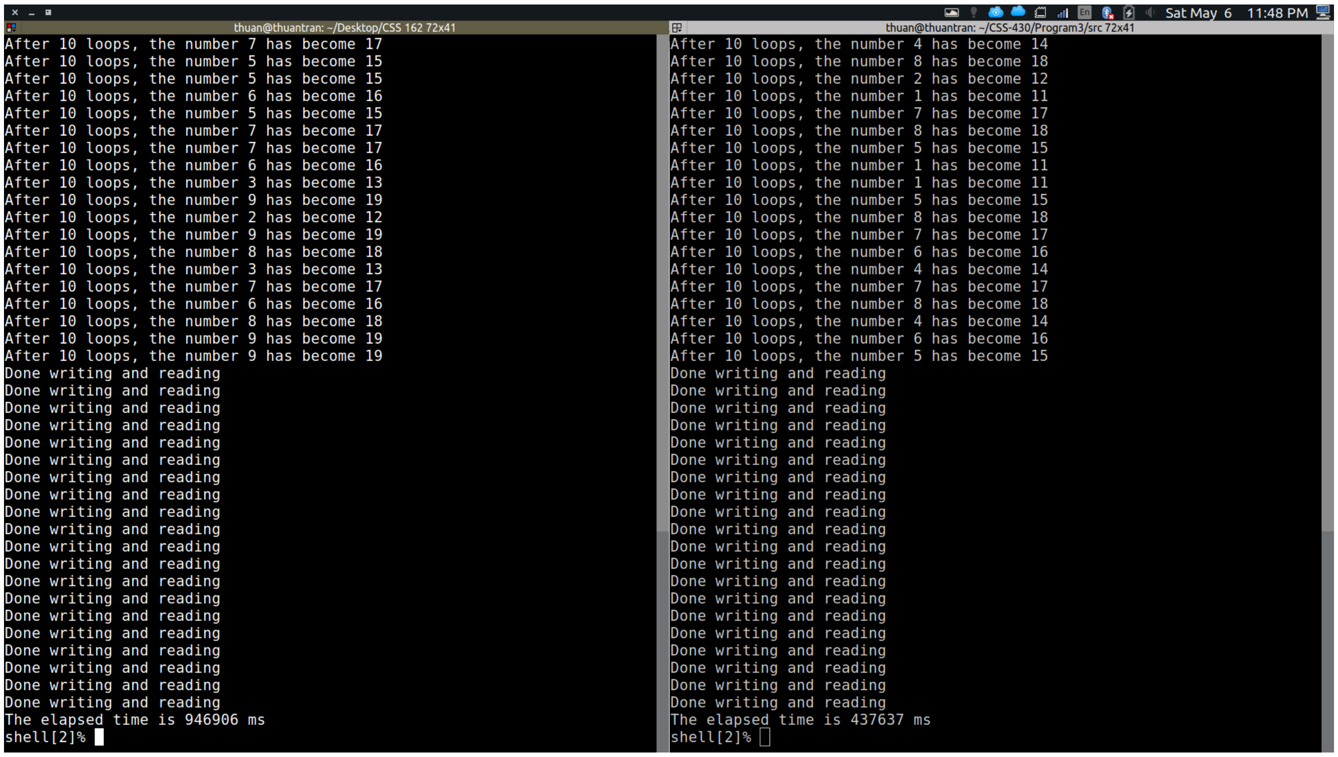
Discussion

Below is the output of running Test3.java with the creations of 20 pairs of threads. The output on the right belongs to the old kernel (busy waiting) where the output on the left is the new kernel (interrupt)

Looking at the output, I can see that the old kernel run twice as fast as the new kernel. I also notice that the old kernel produce the output sequentially. While the new kernel produce the output "Done writing and reading" all at the end (That is because they all end at that time) while the old kernel end one at a time.

One reason why this happen is because the new kernel perform context switch too much. Every time TestThread want to read from the Disk, the kernel interrupt it. Giving the permission to other thread.

Where as the old kernel perform busy waiting, that allow the thread to execute sequentially. Now, I am not saying that the old kernel is better than the new kernel. It just that given this Test3, there is nothing much in it to interrupt. However, depending on different purposes, the new kernel might be better because it allows more important operation to operate first


Sat May 6 11:48 PM
thuan@thuantran: ~/Desktop/CSS 162 72x41
thuan@thuantran: ~/CSS-430/Program3/src 72x41

```
After 10 loops, the number 7 has become 17
After 10 loops, the number 5 has become 15
After 10 loops, the number 5 has become 15
After 10 loops, the number 6 has become 16
After 10 loops, the number 5 has become 15
After 10 loops, the number 7 has become 17
After 10 loops, the number 7 has become 17
After 10 loops, the number 6 has become 16
After 10 loops, the number 3 has become 13
After 10 loops, the number 9 has become 19
After 10 loops, the number 2 has become 12
After 10 loops, the number 9 has become 19
After 10 loops, the number 8 has become 18
After 10 loops, the number 3 has become 13
After 10 loops, the number 7 has become 17
After 10 loops, the number 6 has become 16
After 10 loops, the number 8 has become 18
After 10 loops, the number 9 has become 19
After 10 loops, the number 9 has become 19
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
The elapsed time is 946906 ms
shell[2]%
```

```
After 10 loops, the number 4 has become 14
After 10 loops, the number 8 has become 18
After 10 loops, the number 2 has become 12
After 10 loops, the number 1 has become 11
After 10 loops, the number 7 has become 17
After 10 loops, the number 8 has become 18
After 10 loops, the number 5 has become 15
After 10 loops, the number 1 has become 11
After 10 loops, the number 1 has become 11
After 10 loops, the number 5 has become 15
After 10 loops, the number 8 has become 18
After 10 loops, the number 7 has become 17
After 10 loops, the number 6 has become 16
After 10 loops, the number 4 has become 14
After 10 loops, the number 7 has become 17
After 10 loops, the number 8 has become 18
After 10 loops, the number 4 has become 14
After 10 loops, the number 6 has become 16
After 10 loops, the number 5 has become 15
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
Done writing and reading
The elapsed time is 437637 ms
shell[2]%
```

Illustration 3: Output when running Test3. Left is new Kernel. Right is old Kernel