

Program 3: Synchronization

Report

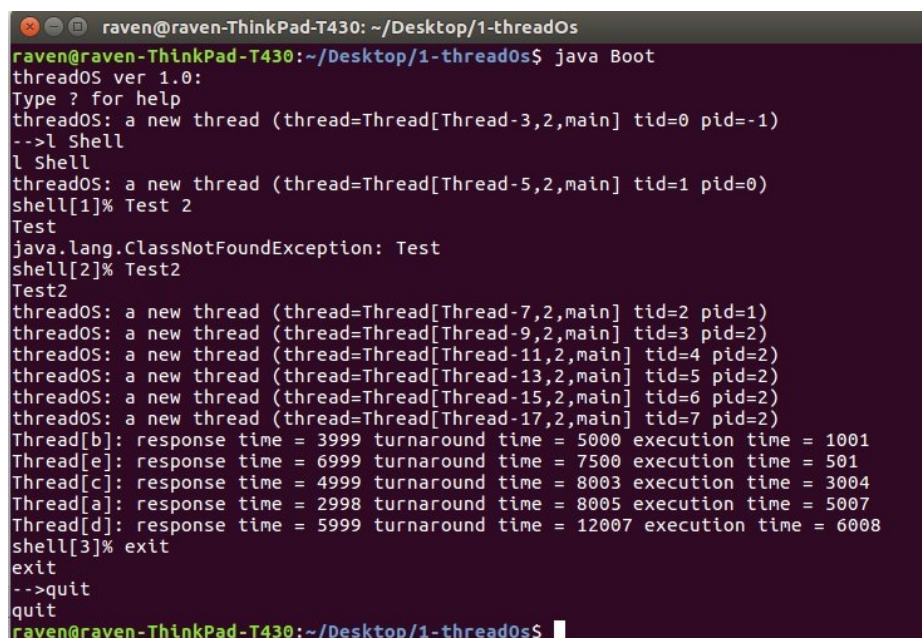
Part 1's Algorithm and design:

Parts inside the kernel were written following the instructions in the comments. First TCB was acquired then tid and pid based on the case and then enqueueAndSleep and DequeueAndWakeup were called.

SyncQueue class include four functions two constructors, enqueueAndSleep , and dequeueAndWakeup. In enqueueAndSleep the incoming thread is put to sleep and its tid will be returned for other threads to finish. In dequeueAndWakeup threads with the incoming pid will wakeup.

QueueNode file includes a Vector called queues. In sleep function first check is for it's the parent by checking the size of the queue then this thread will be put into wait. In wakeup function incoming thread will be put to the vectors and next thread will be notified.

The overall result of the first part is fast and results are shown in the figure below:



```
raven@raven-ThinkPad-T430: ~/Desktop/1-threadOs
raven@raven-ThinkPad-T430:~/Desktop/1-threadOs$ java Boot
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]% Test 2
Test
java.lang.ClassNotFoundException: Test
shell[2]% 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 = 3999 turnaround time = 5000 execution time = 1001
Thread[e]: response time = 6999 turnaround time = 7500 execution time = 501
Thread[c]: response time = 4999 turnaround time = 8003 execution time = 3004
Thread[a]: response time = 2998 turnaround time = 8005 execution time = 5007
Thread[d]: response time = 5999 turnaround time = 12007 execution time = 6008
shell[3]% exit
exit
-->quit
quit
raven@raven-ThinkPad-T430:~/Desktop/1-threadOs$
```

Part 2's Algorithm and design:

Parts inside the kernel were written following the instructions in the comments again. In each case has a while loop with conditions that inside each while ioQueue will call enqueueAndSleep.

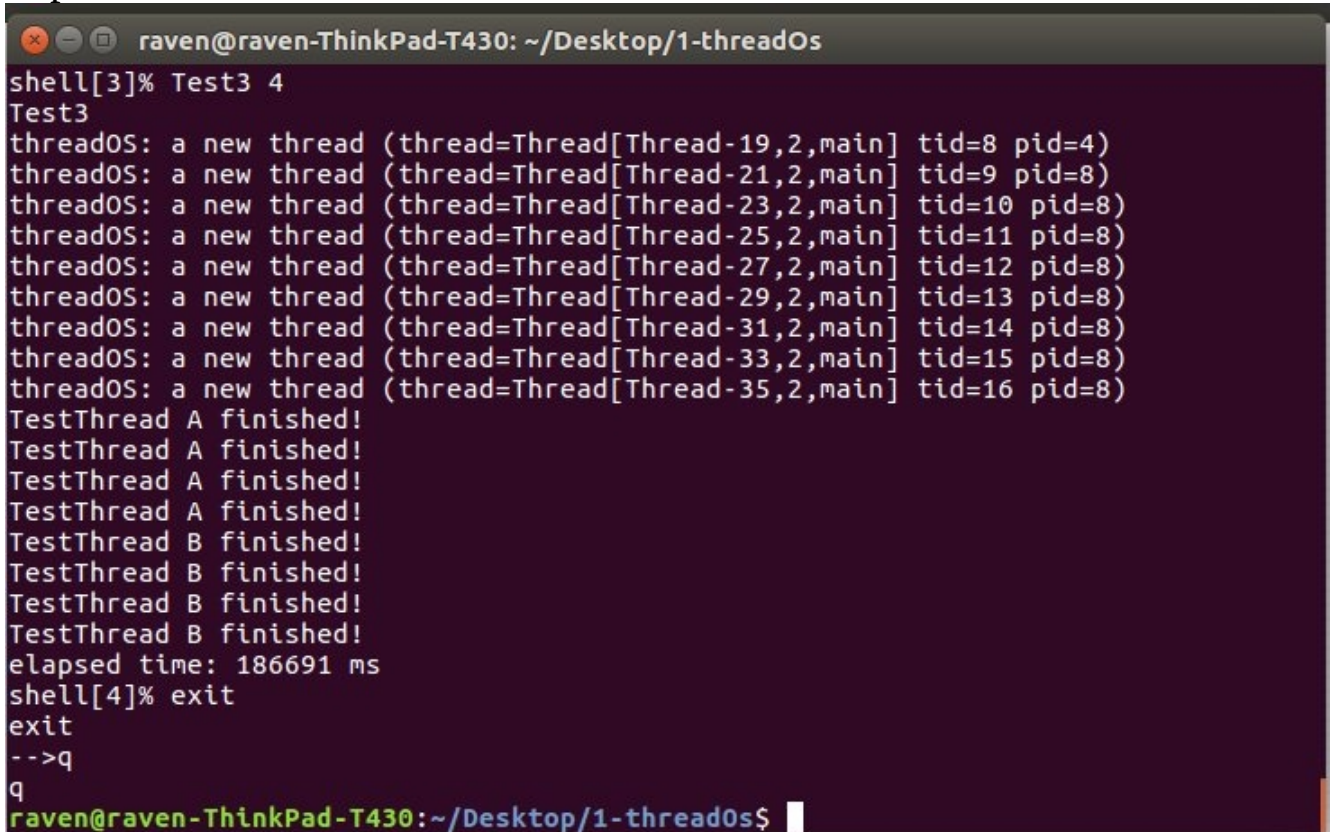
In order to test this algorithm I created Test3 alongside TestThread3. In Test3 two arguments will be called for TestThread3. One will be TestThread3 A and the other one TestThread3 B. Then threads will be executed equal to the argument that will be written in front of Test3.

Inside TestThread3 if the calling thread is A it will calculate some random math equation. If it is B it will write some random bytes into a byte buffer for 1000 blocks.

Following is the results for Test3 in part 1 algorithm implementation:

```
raven@raven-ThinkPad-T430: ~/Desktop/1-thread0s
raven@raven-ThinkPad-T430:~/Desktop/1-thread0s$ java Boot
thread0s ver 1.0:
Type ? for help
thread0s: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Shell
l Shell
thread0s: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
shell[1]% Test3 4
Test3
thread0s: a new thread (thread=Thread[Thread-7,2,main] tid=2 pid=1)
thread0s: a new thread (thread=Thread[Thread-9,2,main] tid=3 pid=2)
thread0s: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=2)
thread0s: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=2)
thread0s: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=2)
thread0s: a new thread (thread=Thread[Thread-17,2,main] tid=7 pid=2)
thread0s: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=2)
thread0s: a new thread (thread=Thread[Thread-21,2,main] tid=9 pid=2)
thread0s: a new thread (thread=Thread[Thread-23,2,main] tid=10 pid=2)
TestThread A finished!
TestThread A finished!
TestThread A finished!
TestThread A finished!
TestThread B finished!
TestThread B finished!
TestThread B finished!
TestThread B finished!
elapsed time: 191954 ms
shell[2]% exit
exit
-->q
q
raven@raven-ThinkPad-T430:~/Desktop/1-thread0s$
```

and next photo is the results for Test3 in part 2's algorithm implementation:

A terminal window titled 'raven@raven-ThinkPad-T430: ~/Desktop/1-threadOs' showing the execution of a program. The user enters 'Test3 4' at the shell prompt. The program outputs 'Test3' followed by eight lines of thread creation messages: 'threadOs: a new thread (thread=Thread[Thread-19,2,main] tid=8 pid=4)' through 'tid=16 pid=8'. Then, it outputs eight 'finished!' messages, alternating between 'TestThread A' and 'TestThread B'. Finally, it outputs 'elapsed time: 186691 ms'. The user enters 'exit' at the shell prompt, and the terminal shows 'exit', '-->q', and 'q' before returning to the shell prompt 'raven@raven-ThinkPad-T430:~/Desktop/1-threadOs\$'.

As it is visible in both figures second implementation will complete the task faster than the first part. Doing second algorithm helps the thread to wait less while they are reading compare to the first algorithm that happens one by one for each thread.