Program 2 (Operating System Scheduler)

Reza Naeemi

Purpose

This assignment implements and compares two CPU scheduling algorithms, the round-robin scheduling and the multilevel feedback-queue scheduling. The step-by-step procedure to complete this assignment is:

1. Observe the behavior of the *ThreadOS Scheduler* which uses a Java-based round-robin scheduling algorithm. Consider why it may not be working strictly in a round-robin fashion.
2. Redesign the *ThreadOS Scheduler* using *Thread.suspend( )* and *Thread.resume( )* so that it will rigidly work in a round-robin fashion.
3. Implement a multi-level feedback-queue scheduler for *ThreadOS.*
4. Compare the rigid round-robin and multi-level feedback-queue schedulers using test thread programs.

Documentation

Java Priority Scheduling

The ThreadOS Scheduler (see Scheduler.java) implements a native round-robin scheduler. Given that ThreadOS is a java application and ThreadOS applications are Java threads, the ThreadOS scheduler is subject to the underlying Java Virtual Machine (JVM) Scheduler. Given this, there is no guarantee that a thread with a higher priority will immediately preempt the current thread. In its default implementation, Scheduler.java does not strictly enforce round-robin scheduling. We can however modify the ThreadOS scheduler to enforce a rigid round-robin algorithm. By using the Thread.suspend() and Thread.resume() methods, we can force threads to block or be ready for execution.

The suspend() method suspends a target thread, whereas the resume() method resumes (moves the thread to a ready state) a suspend thread. To implement a rigid round-robin CPU scheduling algorithm, I modified the ThreadOS Scheduler to dequeue the front user thread from the ready list and resume it by invoking the resume() method. When the quantum expired, I suspend the thread with the suspend() method.

The Scheduler

The Scheduler itself is started by ThreadOS Kernel. It creates a thread queue that maintains all user threads invoked by the SysLib.exec(String args[]) system call. Upon retrieving this system call, ThreadOS Kernel instantiates a user thread and calls the scheduler's addThread(Thread t) method. A new TCB is allocated to this thread and enqueued in the scheduler's thread list. The scheduler repeats an infinite while loop in its run method. It picks up a next available TCB from the list. If the thread in this TCB has not yet been activated (but has been instantiated), the scheduler

starts it first. It thereafter raises up the thread's priority to execute for a given time slice.

When a user thread calls SysLib.exit() to terminate itself, the Kernel calls the scheduler dequeues this TCB from the circular queue, and finds out that it has been marked as terminated, it deletes that TCB.

Private data members of Scheduler.java

In addition to three private data members from the lecture slide example, we have two more data such as a boolean array - *tids[]* and a constant - *DEFAULT_MAX_THREADS*, both related to TCB.

Data members	Descriptions
private Vector queue;	a list of all active threads, (to be specific, TCBs).
private int timeSlice;	a time slice allocated to each user thread execution
private static final int DEFAULT_TIME_SLICE = 1000;	the unit is millisecond. Thus 1000 means 1 second.
private boolean[] tids;	Each array entry indicates that the corresponding thread ID has been used if the entry value is true.
private static final int DEFAULT_MAX_THREADS = 10000;	tids[] has 10000 elements

Methods of Scheduler.java

The following shows all the methods of *Scheduler.java*.

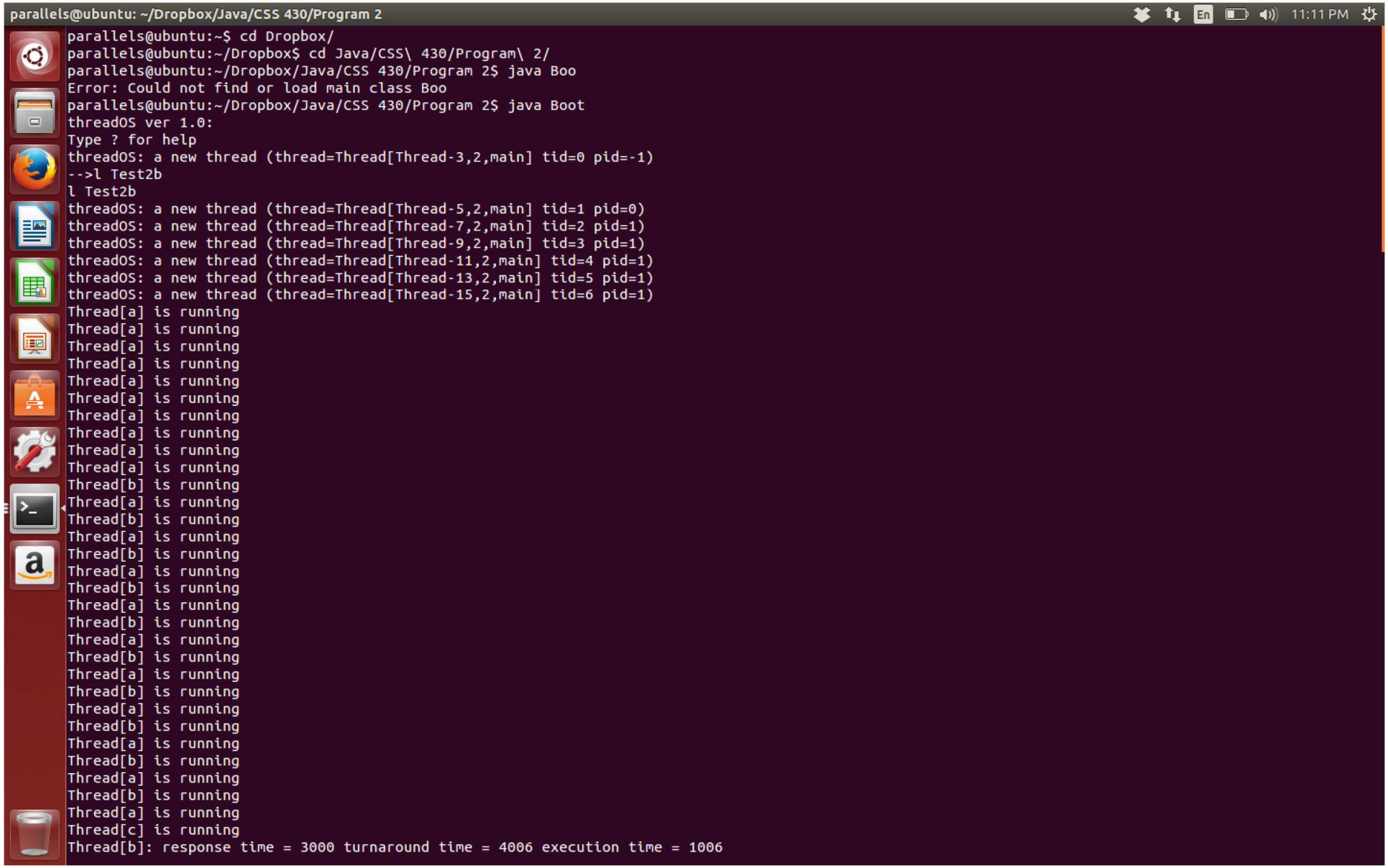
Methods	Descriptions
private void initTid(int maxThreads)	allocates the *tid[]* array with a *maxThreads* number of elements
private int getNewTid( )	finds an *tid[]* array element whose value is false, and returns its index as a new thread ID.
private boolean returnTid(int tid)	sets the corresponding *tid[]* element, (i.e., *tid[tid]*) false. The return value is false if *tid[tid]* is already false, (i.e., if this *tid* has not been used), otherwise true.

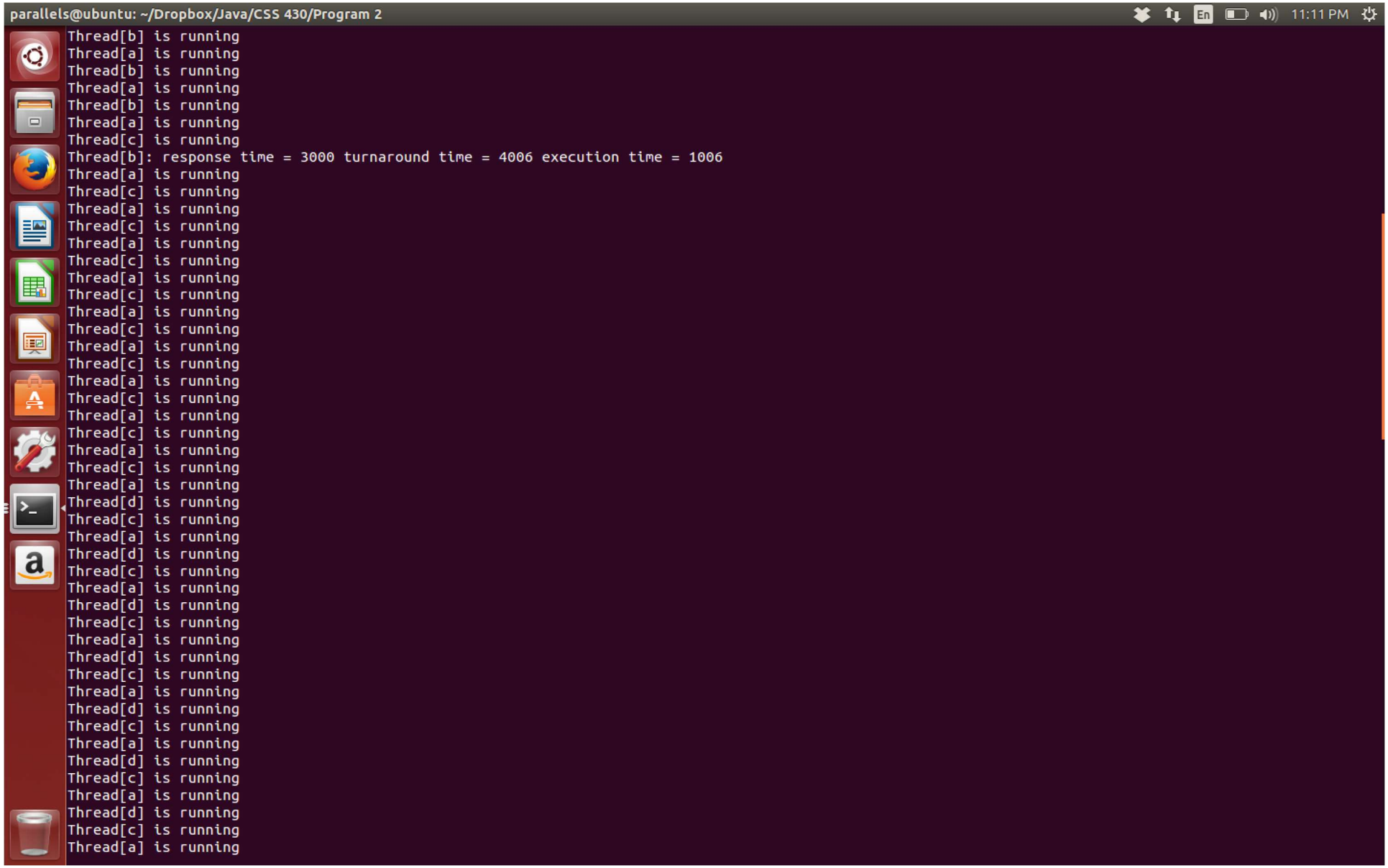
public int getMaxThreads( )	returns the length of the *tid[]* array, (i.e., the available number of threads).
public TCB getMyTcb( )	finds the current thread's TCB from the active thread queue and returns it
public Scheduler(int quantum, int maxThreads)	receives two arguments: (1) the time slice allocated to each thread execution and (2) the maximal number of threads to be spawned, (namely the length of tid[]). It creates an active thread queue and initializes the *tid[]* array
private void schedulerSleep( )	puts the Scheduler to sleep for a given time quantum
public TCB addThread(Thread t)	allocates a new TCB to this thread *t* and adds the TCB to the active thread queue. This new TCB receives the calling thread's id as its parent id.
public boolean deleteThread( )	finds the current thread's TCB from the active thread queue and marks its TCB as terminated. The actual deletion of a terminated TCB is performed inside the run( ) method, (in order to prevent race conditions).
public void sleepThread(int milliseconds)	puts the calling thread to sleep for a given time quantum.
public void run( )	This is the heart of Scheduler. The difference from the lecture slide includes: (1) retrieving a next available TCB rather than a thread from the active thread list, (2) deleting it if it has been marked as "terminated", and (3) starting the thread if it has not yet been started. Other than this difference, the Scheduler repeats retrieving a next available TCB from the list, raising up the corresponding thread's priority, yielding CPU to this thread with *sleep( )*, and lowering the thread's priority.

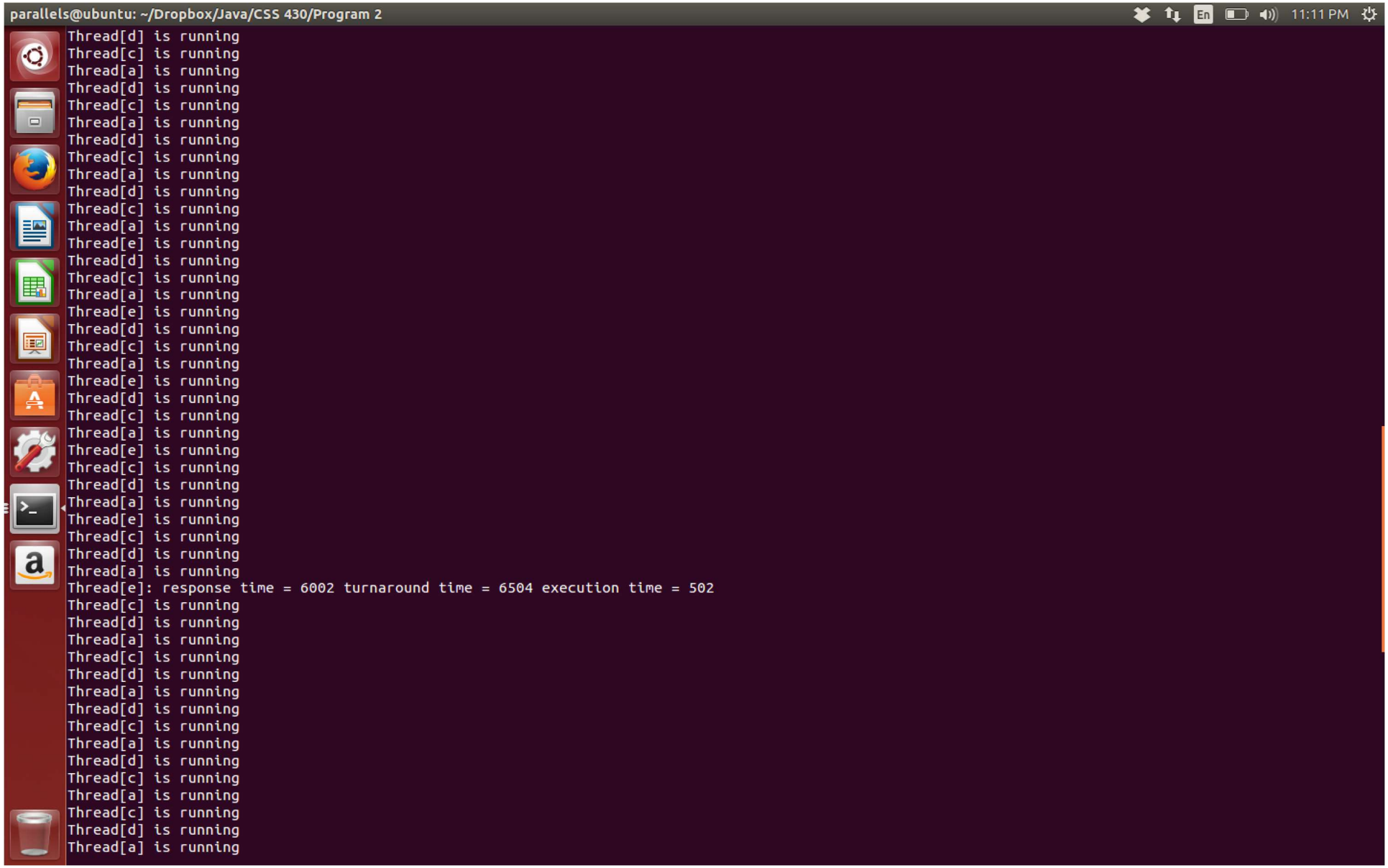
Execution Console Output

Part 1: Modifying Scheduler.java with suspend() and resume() - Test2b

Test2b spawns five child threads from TestThread2b each named Thread[a], Thread[b], Thread[c], Thread[d], and Thread[e]. They print out "Thread[*name*] is running" every 0.1 second. If the round-robin schedule is rigidly enforced to give a 1-second time quantum to each thread, you should see each thread printing out the same message about 10 times consecutively.

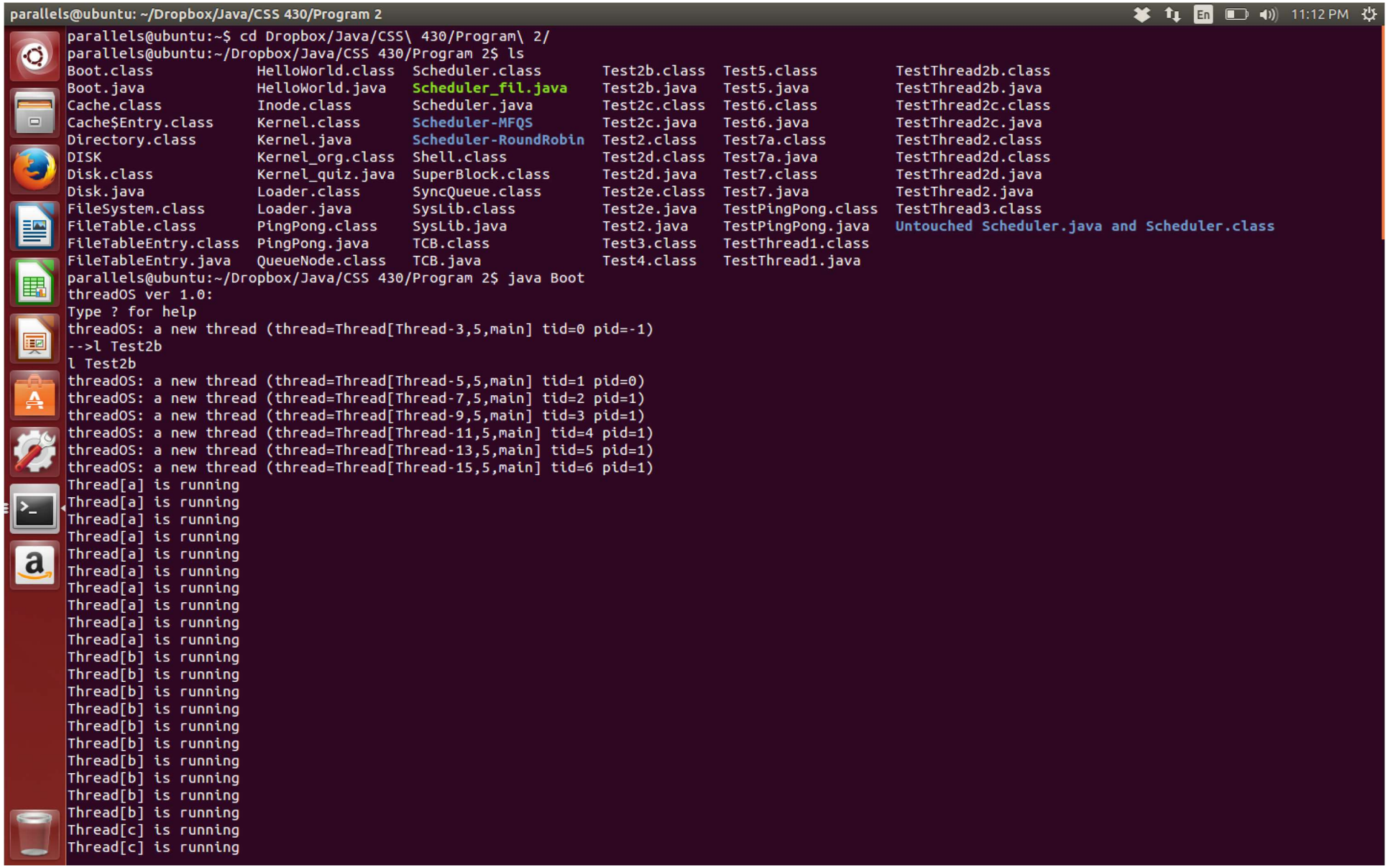

parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:11 PM
parallels@ubuntu:~$ cd Dropbox/
parallels@ubuntu:~/Dropbox$ cd Java/CSS\ 430/Program\ 2/
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ java Boo
Error: Could not find or load main class Boo
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test2b
l Test2b
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
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=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[c] is running
Thread[b]: response time = 3000 turnaround time = 4006 execution time = 1006

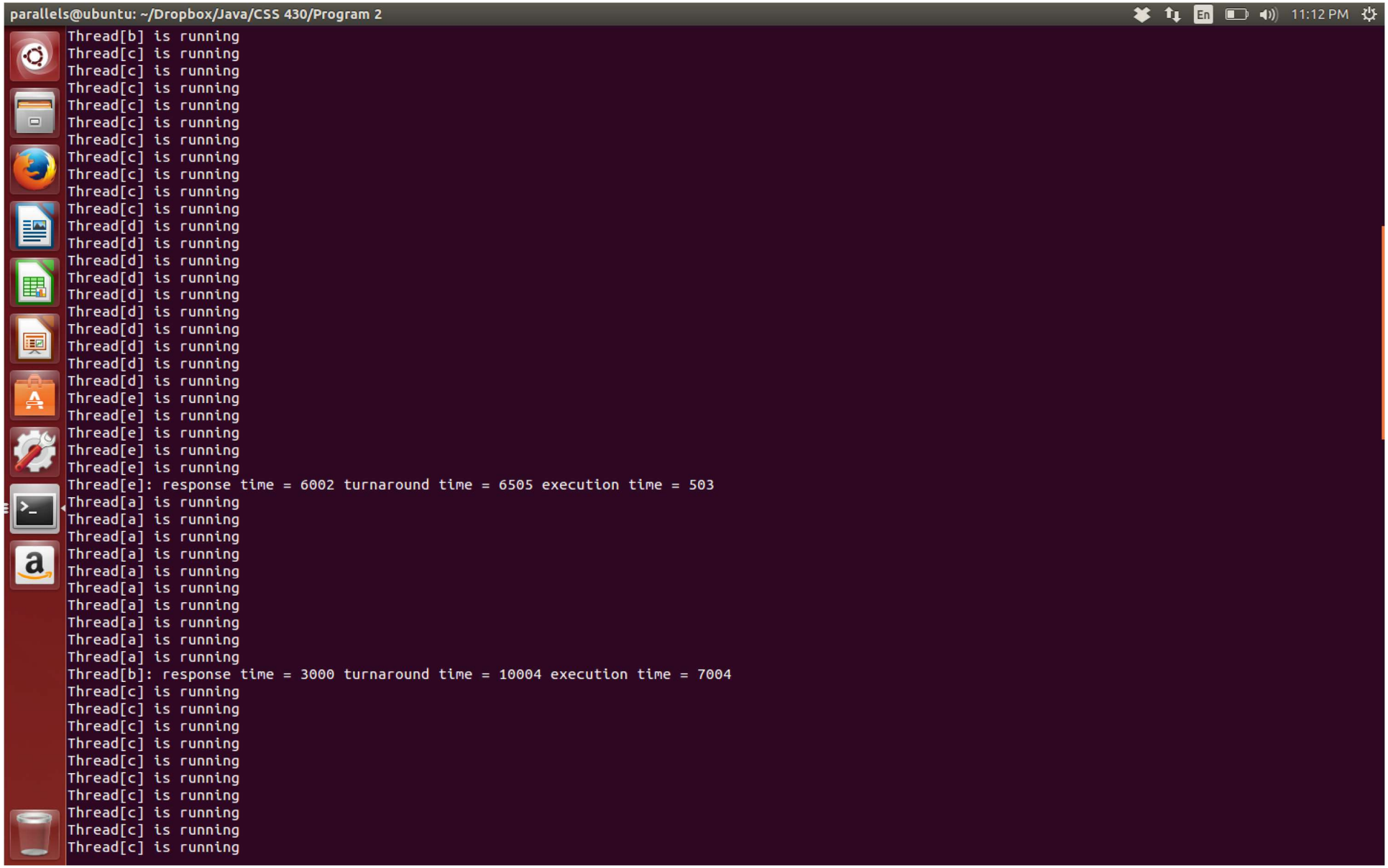

parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:11 PM
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[b] is running
Thread[a] is running
Thread[c] is running
Thread[b]: response time = 3000 turnaround time = 4006 execution time = 1006
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running

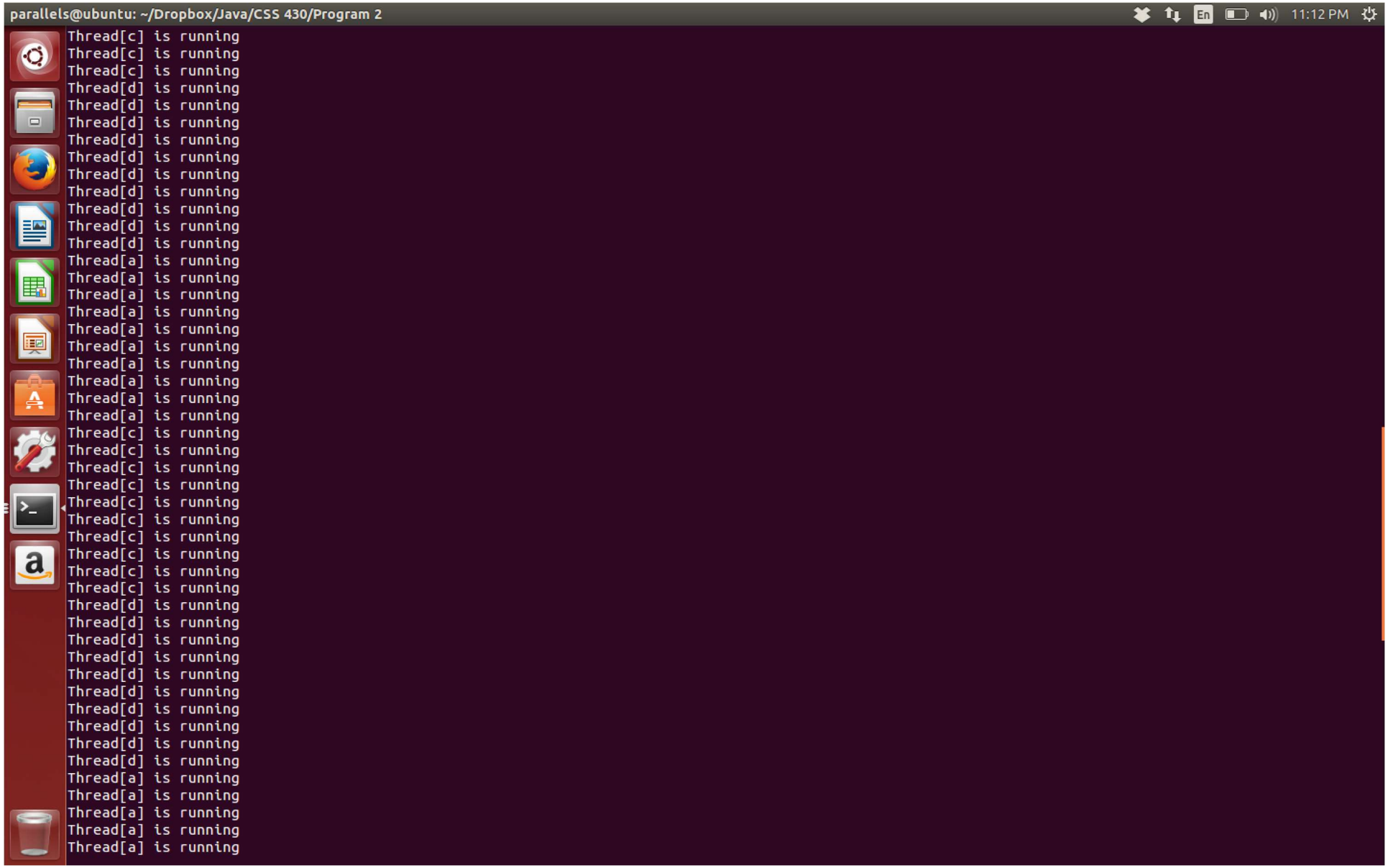

parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:11 PM
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[e] is running
Thread[c] is running
Thread[d] is running
Thread[a] is running
Thread[e] is running
Thread[c] is running
Thread[d] is running
Thread[a] is running
Thread[e]: response time = 6002 turnaround time = 6504 execution time = 502
Thread[c] is running
Thread[d] is running
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[d] is running
Thread[c] is running
Thread[a] is running
Thread[c] is running
Thread[d] is running
Thread[a] is running

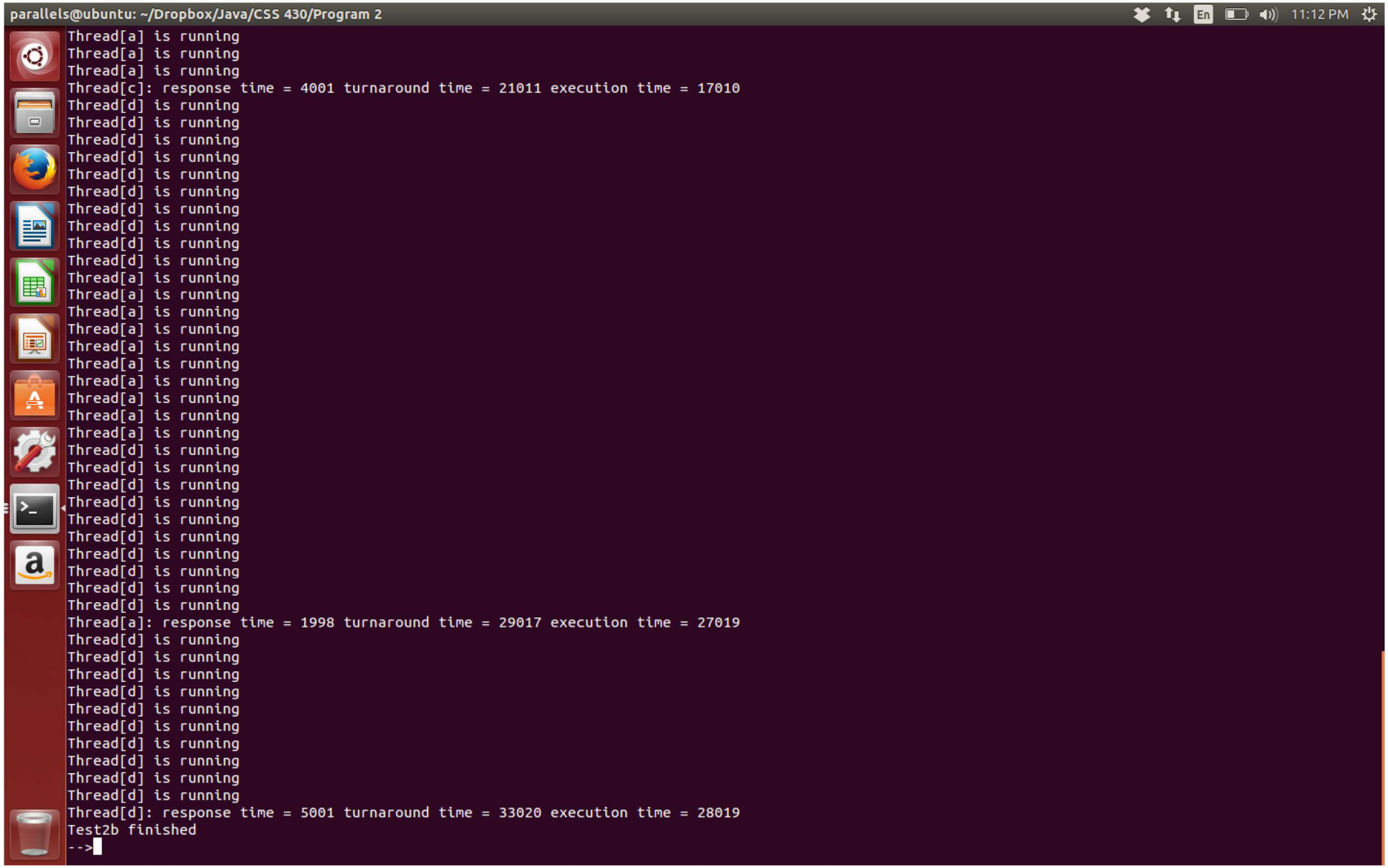
parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2 11:11 PM

```
Thread[c] is running
Thread[d] is running
Thread[a] is running
Thread[c]: response time = 4001 turnaround time = 7013 execution time = 3012
Thread[d] is running
Thread[a]: response time = 2000 turnaround time = 7028 execution time = 5028
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 5001 turnaround time = 11027 execution time = 6026
Test2b finished
-->
```


parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:12 PM
parallels@ubuntu:~$ cd Dropbox/Java/CSS\ 430/Program\ 2/
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ ls
Boot.class HelloWorld.class Scheduler.class Test2b.class Test5.class TestThread2b.class
Boot.java HelloWorld.java Scheduler_fil.java Test2b.java Test5.java TestThread2b.java
Cache.class Inode.class Scheduler.java Test2c.class Test6.class TestThread2c.class
Cache$Entry.class Kernel.class Scheduler-MFQS Test2c.java Test6.java TestThread2c.java
Directory.class Kernel.java Scheduler-RoundRobin Test2.class Test7a.class TestThread2.class
DISK Kernel_org.class Shell.class Test2d.class Test7a.java TestThread2d.class
Disk.class Kernel_quiz.java SuperBlock.class Test2d.java Test7.class TestThread2d.java
Disk.java Loader.class SyncQueue.class Test2e.class Test7.java TestThread2.java
FileSystem.class Loader.java SysLib.class Test2e.java TestPingPong.class TestThread3.class
FileTable.class PingPong.class SysLib.java Test2.java TestPingPong.java Untouched Scheduler.java and Scheduler.class
FileTableEntry.class PingPong.java TCB.class Test3.class TestThread1.class
FileTableEntry.java QueueNode.class TCB.java Test4.class TestThread1.java
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Test2b
l Test2b
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)
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[b] is running
Thread[c] is running
Thread[c] is running


parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:12 PM
Thread[b] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e] is running
Thread[e]: response time = 6002 turnaround time = 6505 execution time = 503
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[b]: response time = 3000 turnaround time = 10004 execution time = 7004
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running

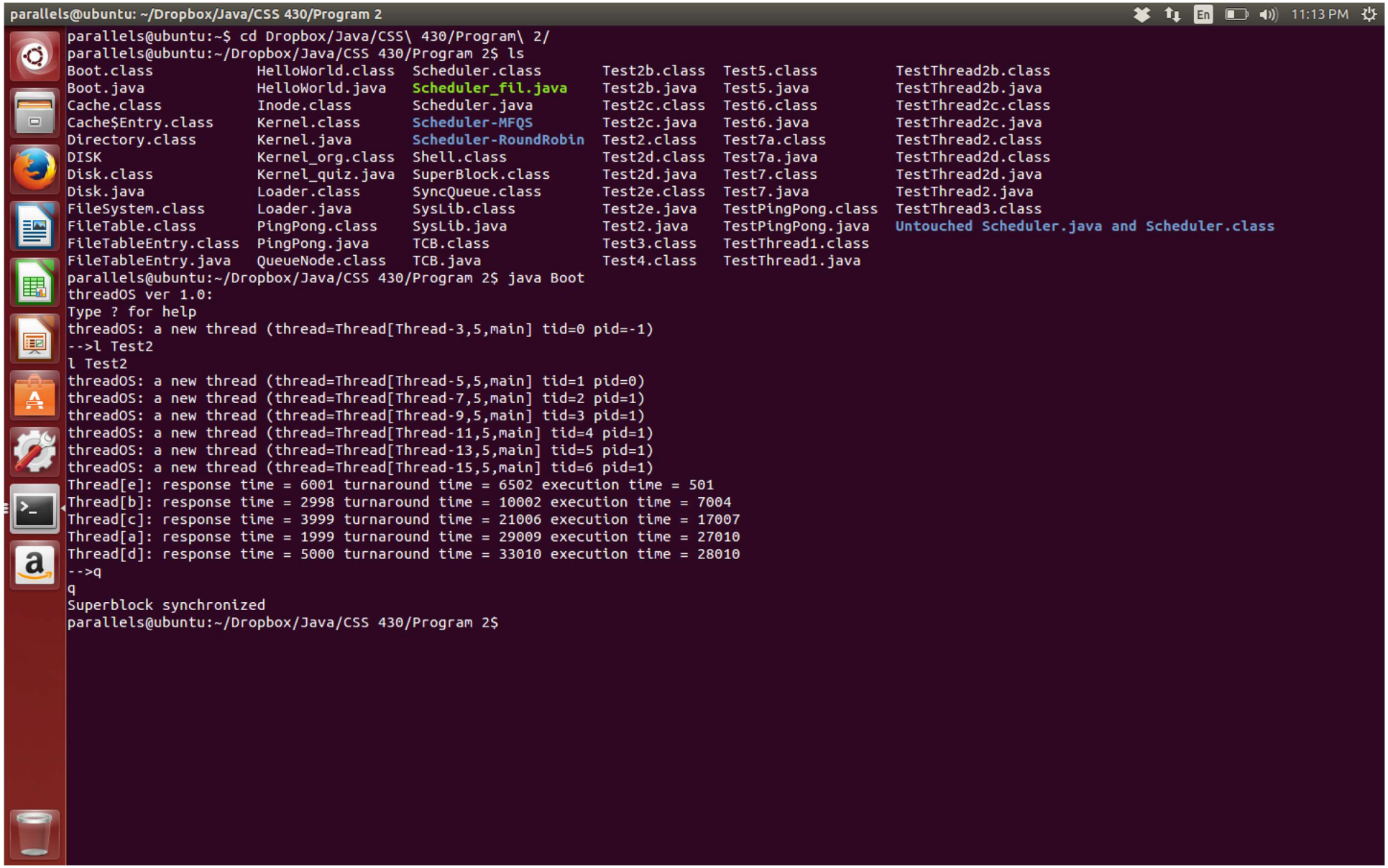

parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:12 PM
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[c] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running

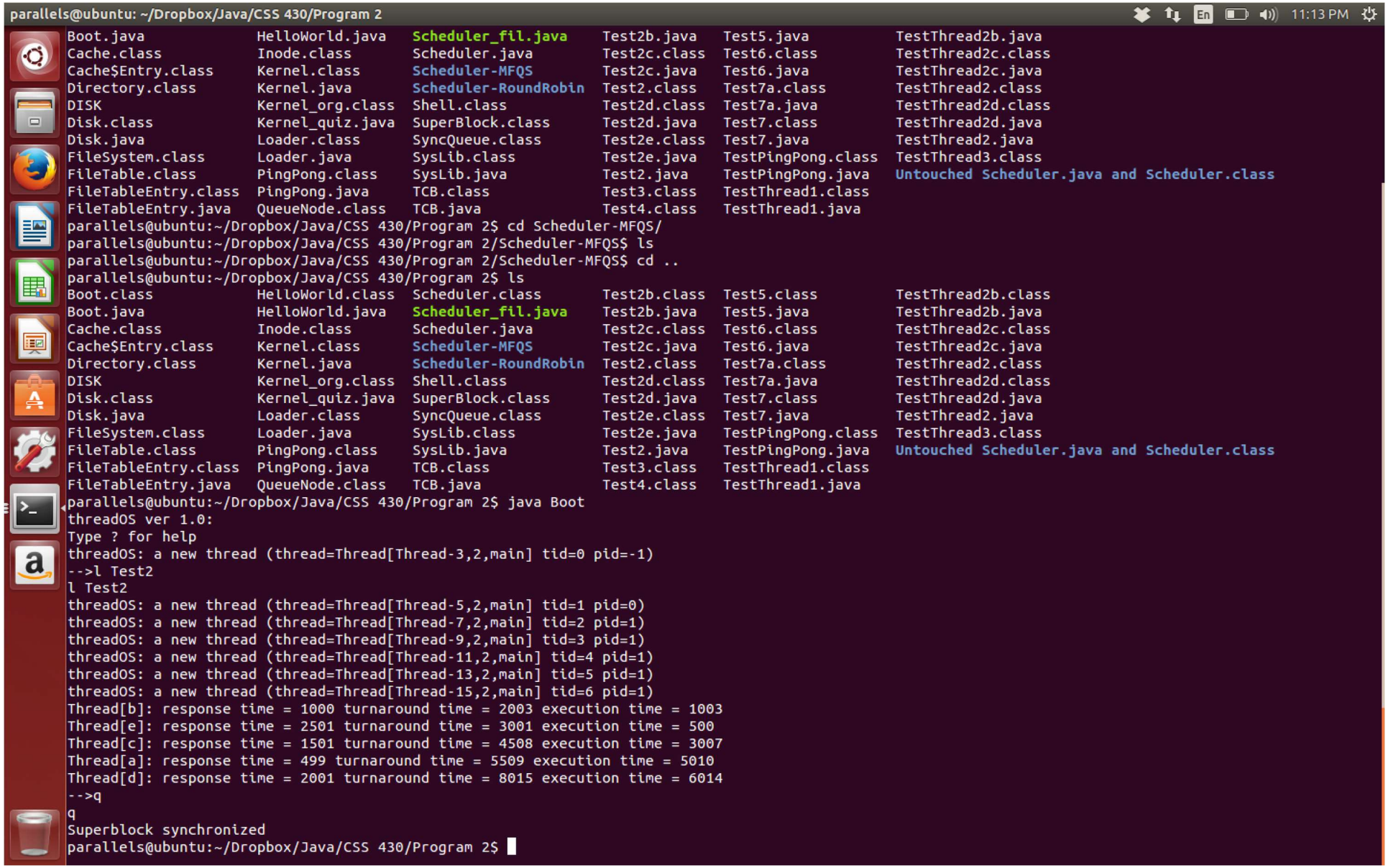

parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:12 PM
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[c]: response time = 4001 turnaround time = 21011 execution time = 17010
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[a] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[a]: response time = 1998 turnaround time = 29017 execution time = 27019
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d] is running
Thread[d]: response time = 5001 turnaround time = 33020 execution time = 28019
Test2b finished
-->

Part 2: Implementing a multilevel feedback queue (MFQS) scheduler

The multilevel feedback queue scheduler operates according to the following specifications:

1. It has three queues, numbed from 0 to 2.
2. A new thread's TCB is always enqueued into queue0.
3. MFSQ scheduler first executes all threads in queue0 whose time quantum is half of the time quantum in Part 1's round-robin scheduler.
4. If a thread in queue0 does not complete its execution for queue0's time quantum, then scheduler moves the corresponding TCB to queue1.
5. If queue0 is empty, it will execute threads in queue1 whose time quantum is the same as the one in Part1's round-robin scheduler. However, in order to react new threads in queue0 your MFQS scheduler should execute a thread in queue1 for only one-half the queue1 time quantum and then check if queue0 has new TCBs pending. If so, it will execute all threads in queue0 first. The interrupted thread will be resumed when all of the queue0 threads are handled. The resumed thread will only be given the remaining part of its time quantum.
6. If a thread in queue1 does not complete its execution and was given a full queue1's time quantum, the scheduler then moves the TCB to queue2.
7. If both queue09 and queue1 are empty, the MFQS swchedulers will execute threads in queue2 whos time quantum is a double of the one in Part 1's round-robin scheduler. However, in order to react to threads whith highter priority in queue0 and queue1, your scheduler should execute a thread in queue2 for q09 increments and check if queue =0 and queue1 have new TCBs. The rest of the behavior is the same as that for queue1.
8. If a thread in queue2 does not complete its execution for queue2's time slice, the scheduler puts it back to the tail of queue2.


parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:13 PM
parallels@ubuntu:~$ cd Dropbox/Java/CSS\ 430/Program\ 2/
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ ls
Boot.class HelloWorld.class Scheduler.class Test2b.class Test5.class TestThread2b.class
Boot.java HelloWorld.java Scheduler_fil.java Test2b.java Test5.java TestThread2b.java
Cache.class Inode.class Scheduler.java Test2c.class Test6.class TestThread2c.class
Cache$Entry.class Kernel.class Scheduler-MFQS Test2c.java Test6.java TestThread2c.java
Directory.class Kernel.java Scheduler-RoundRobin Test2.class Test7a.class TestThread2.class
DISK Kernel_org.class Shell.class Test2d.class Test7a.java TestThread2d.class
Disk.class Kernel_quiz.java SuperBlock.class Test2d.java Test7.class TestThread2d.java
Disk.java Loader.class SyncQueue.class Test2e.class Test7.java TestThread2.java
FileSystem.class Loader.java SysLib.class Test2e.java TestPingPong.class TestThread3.class
FileTable.class PingPong.class SysLib.java Test2.java TestPingPong.java Untouched Scheduler.java and Scheduler.class
FileTableEntry.class PingPong.java TCB.class Test3.class TestThread1.class
FileTableEntry.java QueueNode.class TCB.java Test4.class TestThread1.java
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,5,main] tid=0 pid=-1)
-->l Test2
l Test2
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)
Thread[e]: response time = 6001 turnaround time = 6502 execution time = 501
Thread[b]: response time = 2998 turnaround time = 10002 execution time = 7004
Thread[c]: response time = 3999 turnaround time = 21006 execution time = 17007
Thread[a]: response time = 1999 turnaround time = 29009 execution time = 27010
Thread[d]: response time = 5000 turnaround time = 33010 execution time = 28010
-->q
q
Superblock synchronized
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$


parallels@ubuntu: ~/Dropbox/Java/CSS 430/Program 2
11:13 PM
Boot.java HelloWorld.java Scheduler_fil.java Test2b.java Test5.java TestThread2b.java
Cache.class Inode.class Scheduler.java Test2c.class Test6.class TestThread2c.class
Cache$Entry.class Kernel.class Scheduler-MFQS Test2c.java Test6.java TestThread2c.java
Directory.class Kernel.java Scheduler-RoundRobin Test2.class Test7a.class TestThread2.class
DISK Kernel_org.class Shell.class Test2d.class Test7a.java TestThread2d.class
Disk.class Kernel_quiz.java SuperBlock.class Test2d.java Test7.class TestThread2d.java
Disk.java Loader.class SyncQueue.class Test2e.class Test7.java TestThread2.java
FileSystem.class Loader.java SysLib.class Test2e.java TestPingPong.class TestThread3.class
FileTable.class PingPong.class SysLib.java Test2.java TestPingPong.java Untouched Scheduler.java and Scheduler.class
FileTableEntry.class PingPong.java TCB.class Test3.class TestThread1.class
FileTableEntry.java QueueNode.class TCB.java Test4.class TestThread1.java
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ cd Scheduler-MFQS/
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2/Scheduler-MFQS$ ls
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2/Scheduler-MFQS$ cd ..
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ ls
Boot.class HelloWorld.class Scheduler.class Test2b.class Test5.class TestThread2b.class
Boot.java HelloWorld.java Scheduler_fil.java Test2b.java Test5.java TestThread2b.java
Cache.class Inode.class Scheduler.java Test2c.class Test6.class TestThread2c.class
Cache$Entry.class Kernel.class Scheduler-MFQS Test2c.java Test6.java TestThread2c.java
Directory.class Kernel.java Scheduler-RoundRobin Test2.class Test7a.class TestThread2.class
DISK Kernel_org.class Shell.class Test2d.class Test7a.java TestThread2d.class
Disk.class Kernel_quiz.java SuperBlock.class Test2d.java Test7.class TestThread2d.java
Disk.java Loader.class SyncQueue.class Test2e.class Test7.java TestThread2.java
FileSystem.class Loader.java SysLib.class Test2e.java TestPingPong.class TestThread3.class
FileTable.class PingPong.class SysLib.java Test2.java TestPingPong.java Untouched Scheduler.java and Scheduler.class
FileTableEntry.class PingPong.java TCB.class Test3.class TestThread1.class
FileTableEntry.java QueueNode.class TCB.java Test4.class TestThread1.java
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$ java Boot
threadOS ver 1.0:
Type ? for help
threadOS: a new thread (thread=Thread[Thread-3,2,main] tid=0 pid=-1)
-->l Test2
l Test2
threadOS: a new thread (thread=Thread[Thread-5,2,main] tid=1 pid=0)
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=1)
threadOS: a new thread (thread=Thread[Thread-11,2,main] tid=4 pid=1)
threadOS: a new thread (thread=Thread[Thread-13,2,main] tid=5 pid=1)
threadOS: a new thread (thread=Thread[Thread-15,2,main] tid=6 pid=1)
Thread[b]: response time = 1000 turnaround time = 2003 execution time = 1003
Thread[e]: response time = 2501 turnaround time = 3001 execution time = 500
Thread[c]: response time = 1501 turnaround time = 4508 execution time = 3007
Thread[a]: response time = 499 turnaround time = 5509 execution time = 5010
Thread[d]: response time = 2001 turnaround time = 8015 execution time = 6014
-->q
q
Superblock synchronized
parallels@ubuntu:~/Dropbox/Java/CSS 430/Program 2$

Part 3: Conducting performance evaluations

According to the documentation and instructions provided to us for Program 2, the following table contains the CPU burst time for the five thread use in Test2.java programs:

The following table shows their CPU burst time:

Thread name	CPU burst (in milliseconds)
Thread[a]	5000
Thread[b]	1000
Thread[c]	3000
Thread[d]	6000
Thread[e]	500

FCFS mean "First Come First Serve", and if I recall that's a system where processes are executed to completion in the order that they are provided to the scheduler. Since FCFS doesn't interrupt execution of a thread, so threads will continue execution until completion instead of stopping and getting relegated to lower queues. First Come First Serve is that it does have some advantages easy and simple. This scheduling method is not proactive, that is, the process will run until it finishes. So, short processes which are at the back of the queue have to wait for the long process at the front to finish

Question 2 Test
Results round
robin vs MFQS:
In all cases, response, turnaround, and execution times were all improved by using MFQS instead of round robin. Note that because I also built a MFQS using setPriority instead of suspend and resume, that produced magnitudes faster response, turnaround, and execution times than the suspend and resume versions.

Question 3 FCFS
theoretical question:
So what's interesting about First Come First Serve is that it does have some advantages. If the Scheduler was implemented as FCFS, execution time would go