CSCI 40300/ECE 40800 Operating Systems
Project 2: Synchronization
Due Date: Sunday, November 6, 2016, 11:59pm

1 Overview

This assignment will familiarize you with the Nachos synchronization primitives **Semaphore**, **Lock**, and **Condition**. You will be using these primitives to solve two classic synchronization problems: **Bounded Buffer**, and **Readers/Writers**. Upon completion you should be more comfortable creating robust multithreaded applications that use shared variables.

2 The Assignment

2.1 Preparation

2.1.1 Nachos related background

In order to extract it, type the following command at the shell (on tesla or some other linux/unix system):

```
tar xvfz mp2given.tgz
```

You will see tar listing out the files it's creating for you; the output will be placed in the directory mp2given. **Do not issue this command again from this directory unless you want to overwrite all changes you have made!** If you wish to replace individual files, untar the mp2given.tgz file in a different directory and copy the relevant files.

2.1.2 Project related background

To do the project, you need to read about synchronization in your textbook and the lecture slides. Pay special attention to the sections about the Bounded Buffer and Readers/Writers problems. Read the Nachos source code. You will be doing your programming work in the threads directory for this assignment. You will need a complete understanding of Semaphore.java, Lock.java, and Condition.java for this assignment. It would also be good to read SynchTest.java which is the code to test your implementation of the algorithms. Learning about Java synchronization is useful for your personal enlightenment, but you are NOT allowed to use it in this assignment.

2.2 Implementation Details

Files to be modified: BoundedBuffer.java (for producer/consumer problem), and Database.java (for readers/writers problem).

Look for the places in the source code given to you where it says MP2. It means you might (or might not depending on your implementation) have to add some code at that place.

2.2.1 Bounded Buffer

You will be making changes to BoundedBuffer.java for this part of the MP.

The size of the buffer you are to implement will be passed into the BoundedBuffer constructor.

Implement the functions BoundedBuffer.produce(char c) and BoundedBuffer.consume(). *You need to use the Nachos synchronization primitives given to you in your implementation*. If you don't use any synchronization and by some miracle your output matches the standard output, you will not get credit for this part of the assignment. Make sure you make a call to SynchTest.addToOutputString(char c) in your consume method so that the test program can test your implementation.

2.2.2 Readers/Writers

You will be making changes to Database.java for this part of the MP.

The database has the functions startRead, endRead, startWrite, and endWrite. You are to implement the Readers/Writers problem using these functions. The test classes Reader and Writer are located in SynchTest.java. These routines will test whether you have implemented the problem successfully.

The solution of the text book and slides gives waiting readers priority over waiting writers. For example, if a writer is waiting for access to the database and a reader is currently reading, another reader can come and start reading without waiting at all. The writer must wait for all readers that arrive and start executing to finish reading before it can start writing. For this MP, we will be changing the priority. We want updates to the database to have priority. So, if there is a writer waiting to write, any readers that arrive must wait for all writers that are waiting to write as well as any writers that arrive in the meantime. As long as there is a reader reading *and no writer is waiting*, or new reader arrives while another reader is reading, readers get to read the data. But when a reader arrives, and if there is at least one writer waiting to write, all the writers that follow (while the reader is waiting) should go through before the reader.

A good idea would be to implement the algorithm in the book, get it working, and then to update it accordingly to implement the writer priority. Not only will this be easier, but it will help you to understand what the assignment is asking you to do.

2.3 Testing and submitting your MP

The final version of your project (correctly compiling and running, and properly debugged) should be submitted on the project due date. For the final version of your project code, you must submit the entire source directory as described below, so that we can uncompress, compile and test it.

Your code must compile and run on the tesla server (linux) in the department for this and all the following assignments. It should not matter where you develop your code for this MP but please test it on this machine before handing it in.

There is only 1 test case for this MP. The command ./test0 will test both BoundedBuffer and Readers Writers.

When you are satisfied that your code works, submit your solution on oncourse. In order to submit your MP 2 implementation, run

```
make cleanclass
```

under the threads directory, which will remove all the compiled (.class) files. Then go to the mp2given/.. directory (i.e., one level above the mp2given source code) and give the command:

```
tar cvfz mp2given.tgz mp2given
```

Then, submit this compressed source code on oncourse, in the assignment section to be graded. Please do not send it as attachments to email, etc.

2.4 Grading criteria:

Your grade will be based on the correctness of your code and largely based on your code's ability to produce correct output in the test cases. Style and legibility is important, especially if the code is hard to find correct: if we can't see that your code is correct, we can't give you points. **Crashing or failing to compile is of course not correct behavior.**

The final project submission	
Bounded Buffer Producer/Consumer	50%
Readers/Writers	50%
Total	**100%**

3 Some Nachos-Specific Details

The files to look at for this assignment are:

Semaphore.java -- definition of the nachos semaphore
Lock.java -- definition of nachos locks.
Condition.java -- definition of nachos condition variables.
SynchTest.java -- Runs the test program for Bounded buffer and readers/writers problem.
BoundedBuffer.java -- Your implementation of the bounded buffer algorithm.
Database.java -- Your implementation of the readers/writers problem.

4 General Rules and Guidelines

You should definitely read this whole handout as well as the relevant parts of the textbook before you begin coding. If you write a lot of code before you understand your assignment, it will show; small code is also much easier to make clear, elegant, and well-commented.

The MPs for this class are to be done individually; you may discuss general operating system concepts or system related problems with one another, including those concepts which are covered on this MP, but not the details of the assignment solution with one another.