EECS 211
Advanced System Software
Winter 2009

Assignment 2

Posted: January 14, 2009
Due: January 28, 2009

Topic: Concurrency and Synchronization in Nachos

Instructions:

The goal of this assignment is to develop and implement concurrency and synchronization primitives in the Nachos system. This assignment is based on and mostly follows the *"Nachos Assignment 1"* described in the file **`doc/thread.ps`** of the Nachos installation (see our previous assignment). The instructions below assume that you read **`doc/thread.ps`** in parallel.

Task 1: Implement the missing locks and condition variables in Nachos

See item 1 in **`doc/thread.ps`**. Go into the **`threads`** directory and complete the code for the classes **`Lock`** and **`Condition`** in files **`synch.h`** and **`synch.cc`**. It will be helpful to look at the code in file **`synchlist.cc`** and **`synchlist.h`** to understand the use of locks (member **`lock`**) and condition variables (member **`listEmpty`**).

Task 2: Test your locks and condition variables

To test your implementation, modify the code in **`threadtest.cc`** such that a new condition variable **`CV`** and a lock **`CL`** are used for the thread synchronization (instead of the call to the **`Yield()`** method. Using two threads, produce the same output as the original code, but without calling the **`Yield()`** method.

More specifically, we want the execution to safely (!) alternate between the two threads so that we are guaranteed to get the original output, as follows:

```
*** thread 0 looped 0 times
*** thread 1 looped 0 times
*** thread 0 looped 1 times
*** thread 1 looped 1 times
*** thread 0 looped 2 times
*** thread 1 looped 2 times
*** thread 0 looped 3 times
*** thread 1 looped 3 times
*** thread 0 looped 4 times
```

`*** thread 1 looped 4 times`

Note that the unmodified/original version in **`threadtest.cc`** produces different output for different values supplied by the **`-rs`** option! For example, the output of **`nachos -rs 4`** starts with thread 1 instead of thread 0.

Your implementation should produce the original output listed above *regardless* of the **`-rs`** option passed to nachos!

Implementation instructions:

For your test implementation, start from the file **`/users/faculty/doemer/eecs211/threadtest.cc.W09templateA2`** .
In this file, you will find two new variables **`CV`** and **`CL`** defined which you should use to implement the desired synchronization. The template file also provides you with two separate functions for the threads **`SimpleThread0`** and **`SimpleThread1`**.

Implement the alternating execution of the threads only by using the provided condition variable **`CV`** and the condition lock **`CL`**. No other variables are allowed. Also, don't modify any given code, just add the needed synchronization calls.

Deliverables:

Submit the completed source files **`synch.h`** and **`synch.cc`**, as well as your modified **`threadtest.cc`** which tests your synchronization implementation.
Briefly explain the safe use of condition variables (few sentences) in the body of your email. Finally, to show that your implementation always produces the same output, provide also a log file **`output.log`** (cut/paste from your shell window) of the output created when you run nachos with the options **`-rs 1`**, **`-rs 2`**, **`-rs 3`**, **`-rs 4`**, and **`-rs 5`**.

Submission instructions:

To submit your homework, send an email with subject "EECS211 HW2" to the course instructor at doemer@uci.edu. Please provide the explanation in the body of your email, but supply the source code and log file as separate attachments.

To ensure proper credit, be sure to send your email before the deadline:
January 28, 2009, 12pm (noon).

--
Rainer Doemer (ET 444C, x4-9007, doemer@uci.edu)