EECS 211
Advanced System Software
Winter 2007

Assignment 2

Posted: January 25, 2007
Due: February 1, 2007

Topic: Concurrency and Synchronization in Nachos

Instructions:

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

Task 1: Understand the given framework

Go into the **`threads`** directory. Run the given program **`nachos`** to test the given code. Use the debugger **`gdb`** or **`ddd`** to run the program step by step. Trace the execution path by reading through the appropriate source files. Make sure you understand what is going on in the function **`SWITCH`** (the debugger may be confusing!) Run the program also with the debug option **`-d`**. You may also want to experiment with the option **`-rs <seed>.`** (Note: additional options to Nachos are available, see the comments in file **`main.cc`**, but most do apply to later assignments only).

Deliverable 1: (20 points)

Briefly (in about 10 sentences) describe the execution path of the unmodified program (i.e. **`threadtest.cc`**). In particular, explain the functionality of the **`Thread::Yield()`** method and its underlying **`SWITCH`** function. What does **`SWITCH`** do, and why is this not implemented in C?

Briefly describe also (in 5 sentences max.), what changes in the Nachos execution if you supply the option **`-rs 0`**, and what the reasoning behind this is.

Modify the **`threadtest.cc`** example such that 3 threads are competing for execution. Briefly describe your code changes and the outcome (again max. 5 sentences).

Task 2: Implement the missing locks and condition variables in Nachos

See item 1 in **`doc/threads.ps`**. 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`**).

To test your implementation, modify the code in **`threadtest.cc`** such that a new condition variable **`CondVar`** is used for the thread synchronization (instead of the call to the **`Yield()`** method. Try to produce the same output as the original code, but without calling the **`Yield()`** method.

Deliverable 2: (30 points)

Submit the completed source files **`synch.h`** and **`synch.cc`**, as well as your modified **`threadtest.cc`** which tests your condition variables. Please also provide a script of the running program (just cut/paste from your shell window).

Submission instructions:

To submit your homework, send an email with subject “EECS 211 HW 2” to the course instructor at doemer@uci.edu. Please submit the deliverables listed above as attachments.

To ensure proper credit, be sure to send your email before the deadline:
February 1, 2007, 11:59pm (before midnight).

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