EECS 211
Advanced System Software
Winter 2008

Assignment 2

Posted: January 16, 2008
Due: January 30, 2008

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: Analyze the given thread mechanism

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 (or function by function). 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 show you confusing results here!) Run the Nachos program also with the debug option **`-d`** and notice the changes. Finally, experiment with the option **`-rs <seed>`** and observe its effects.
(Note: additional options to Nachos are available, see the comments in file **`main.cc`**, but most do apply to later assignments only).

Deliverable 1: (10 points)

Briefly (in about 5 sentences) describe the execution path of the unmodified Nachos program. In particular, explain the functionality of the **`Thread::Yield()`** method and its underlying **`SWITCH`** function. What is the purpose of **`SWITCH`**, and why is this function not implemented in C/C++?

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

See item 1 in **`doc/thread.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. Using two threads, 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.
Briefly explain your implementation (3 sentences) and list the log of your running program (cut/paste from your shell window).

Submission instructions:

To submit your homework, send an email with subject “EECS211 HW2” to the course instructor at doemer@uci.edu. Answer the questions in the body of your email, but supply the source code files as separate attachments.

To ensure proper credit, be sure to send your email before the deadline:
January 30, 2008, 11:59pm (before midnight).

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