EECS 211
Advanced System Software
Winter 2006

Assignment 2

Posted: January 26, 2006
Due: February 2, 2006

Topic: Concurrency and Synchronization in Nachos

Instructions:

The goal of this second assignment is to develop, implement and test 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 our 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: (10 points)

Brief description **`task1.txt`** (about 5 sentences) describing the execution path of this unmodified program and the functionality of the **`SWITCH`** function.
Briefly describe also (again, in about 5 sentences), what changes if you supply option **`-rs 1`** and why this happens.

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`**).

Note that in order to test your code you will need to implement Task 3 below.

Deliverable 2: (20 points)

Completed source files **`synch.h`** and **`synch.cc`** (with proper comments!).

Task 3: Implement a producer-consumer example with a bounded buffer

See item 2 in **`doc/threads.ps`**. To implement this, replace/modify/extend the code in file **`threadtest.cc`** such that it creates producer and consumer threads which communicate via a bounded buffer (see also chapter 6.6.1 in the textbook). The producer and consumer threads should run as two functions named **`Producer`** and **`Consumer`**, respectively. The producer thread should put a string character by character into the buffer, whereas the consumer thread reads a string character by character and prints it to the screen.
The bounded buffer should be implemented as a class **`Buffer`** which allows the maximum number of characters in the buffer to be set at the time of instantiation (constructor parameter). The class **`Buffer`** should provide two methods named **`Put`** and **`Get`** which place a character into the buffer, or take one character out, respectively. Make sure to properly synchronize these methods using your locks and condition variables implemented above. These should be instantiated as members of the class **`Buffer`** and properly be used by the **`Put`** and **`Get`** methods (*not* by the **`Producer`** and **`Consumer`** functions!).

Test your bounded buffer example using the following 3 cases:

1. 1 producer, 1 consumer, buffer size 10
2. 1 producer, 2 consumers, buffer size 5
3. 2 producers, 1 consumer, buffer size 10

For each case, use the following test message:
"**`The quick brown fox jumps over the lazy dog.`**"

For switching between these test cases, you may use preprocessor directives. Test case 1 should be run if **`TEST1`** is defined at compile time, case 2 should be run if **`TEST2`** is defined, etc.
Test your code thoroughly! Make use of the **`-rs <seed>`** option to test context switches at different times. Your program should run flawlessly in any case. Also,

please add sufficient comments and DEBUG statements in the code to make your debugging (and my grading!) easier.

Deliverable 3: (40 points)

a) Modified source code **`threadtest.cc`** with class **`Buffer`** and the **`Producer`** and **`Consumer`** functions
b) A type-script **`bounded_buffer.log`** (copy of your shell outputs) showing that your program can correctly run the three test cases listed above.

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 2, 2006, 12am (midnight).

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