EECS 211
Advanced System Software
Winter 2011

Assignment 3

Posted: February 11, 2011
Due: February 23, 2011

Topic: Priority scheduling in Nachos

Instructions:

The goal of this assignment is to develop, implement and test task scheduling in the Nachos system. This assignment continues the previous assignments based on the “Nachos Assignment 1” described in the file **`doc/thread.ps`** of the Nachos installation. Again, the instructions below assume that you read **`doc/thread.ps`** in parallel.

Task 1: Implement a priority-based scheduler

See item 8 in **`doc/thread.ps`**.
Again, we will work in the **`threads`** directory. As you have noticed in the previous assignments, the original Nachos scheduler implements a straight-forward first-come-first-served (FCFS) scheduling policy. We will change that now into a priority-based policy. That is, with each thread, we will associate a priority between 0 and 9, 0 being the highest priority (first choice).

In order to implement the priority scheduler, you will need to modify the Nachos source code only in the files **`thread.cc`**, **`thread.h`**, and **`scheduler.cc`**.

Hints: Add an integer value for the priority to the Thread class which gets initialized when a Thread object is created. Then, change the order of the threads in the scheduler ready queue according to the thread priority. You will see, there is not much new code to write!

Task 2: Add synchronization to a bounded buffer template for safe inter-thread communication

See item 2 in **`doc/threads.ps`**.
For safe synchronization in the bounded buffer, use the locks and condition variables implemented in Assignment 2 (*don’t* use semaphores!). Note that the bounded buffer described in chapter 6.6.1 in the textbook is implemented using semaphores, so that is *not* a solution to this assignment.

To start, you may use the following template file:
`/users/faculty/doemer/eecs211/threadtest.cc.W11templateA3`
Copy this file into your **`threads`** directory as file **`threadtest.cc`**.

At the beginning of the template file, you find a bounded buffer implemented as a new class **`Buffer`**. The buffer size (maximum queue length) is set at the time of instantiation (as a parameter to the constructor). The class **`Buffer`** provides two public methods named **`Load`** and **`Store`** which take a single character (type **`char`**) out of the buffer, or place a character into the buffer, respectively (for details, see the provided template file **`threadtest.cc`**).

You will need to add statements to properly synchronize the **`Load`** and **`Store`** methods using locks and condition variables (only!). Your added locks and condition variables should be instantiated as members inside the **`Buffer`** class, and should be properly called by the necessary methods so that the user of the buffer does not need to worry about any synchronization.

Test your buffer implementation using a producer-consumer example. In the provided template file **`threadtest.cc`**, 2 producer and 2 consumer threads are instantiated which communicate via one shared instance of the bounded buffer. Note that the priority of the threads is defined when the thread objects are constructed (priorities 4, 3, 2, and 1, i.e. from low to high).

Hint: Don't modify anything but the **`Buffer`** class and its implementation. There is no need to change any code for the consumer and producer threads.
And again, there is not much new code to write.

Deliverables:

- Briefly explain your added synchronization (few sentences) in the body of your email.
- Submit the modified source files **`thread.cc`** and **`thread.h`** as attachments.
- Submit the modified **`scheduler.cc`** as attachment.
- Submit the completed source file **`threadtest.cc`** as attachment.
- Submit a log file **`log.txt`** as attachment that shows the producer and consumer actions.

Submission instructions:

To submit your homework, send an email with subject “EECS211 HW3” to the course instructor at doemer@uci.edu. Please put your text in the body of the email and supply the source files as attachments.

To ensure proper credit, be sure to send your email before the

Deadline: Wednesday, February 23, 2011, at 2pm (sharp!)

--
Rainer Doemer (EH3217, x4-9007, doemer@uci.edu)