Assignment 3: February 23 through March 2

For Lecture, Monday, February 23:

In preparation for today's lecture on concurrency, read Ward chapter 19.

For Recitation, Tuesday, February 24:

Read Savage's paper, "Eraser: A Dynamic Data Race Detector for Multithreaded Programs" (reading #12). If you read this paper sequentially, you might not reach all the important issues. Skim over the section describing the Lockset algorithm. After you understand the main concepts of the paper, return to the Lockset algorithm. We do not expect you to memorize this algorithm. Rather, we expect you to learn about synchronization issues.

In figure 2 of the paper, remember that "y:=y+1" is not necessarily atomic. First, the processor loads the value of y into a register. Next, the processor increments the register value. Finally, the value is written back to y. Can you find interleavings which result in different values of y?

After reading the paper, write a one-page report on "Thread synchronization problems: dynamic detection versus static prevention":

> Birrell's "Introduction to Programming with Threads" offers advice to help *prevent* deadlock, data races, and performance degradation. Alternatively, source code can be analyzed for potential problems. For instance, Sun's lock_lint program statically analyzes source code for data races at compile-time. On the other hand, Eraser dynamically *detects* data races during execution.
>
> With respect to synchronization problems, discuss one advantage of dynamic detection over static prevention and one advantage of static prevention over dynamic detection. Under what circumstances will one method be better than the other? Issues may include performance, effectiveness, and correctness. Reviewing Birrell's threads paper will help you draw comparisons.

For your convenience, here are some useful definitions:

Static prevention
: When all operations occur before execution to help prevent problems.

Dynamic detection
: When operations occur during execution to help detect problems.

Bohr bug /bohr buhg/
: Jim Gray calls a bug with a repeatable, deterministic behavior a Bohr bug (e.g., a program which

always fails when you divide by zero).

heisenbug /hi:'zen-buhg/

Jim Gray calls a bug with non-deterministic (or timing dependent) behavior a heisenbug. These are particularly nasty bugs (e.g., a data race).

The Eraser paper refers to "static detection" rather than "static prevention." However, you should concentrate on the question of static versus dynamic. In dynamic detection, a program will try to catch errors when they happen, but will halt execution when errors arise. In static prevention, a program will try to catch errors before they happen -- preventing errors from happening in the first place.

For Lecture, Wednesday, February 25:

This lecture is the first lecture of four covering networking. In preparation, read part I of Halstead's "6.033 networking notes" (reading #13). The optional reading in Tanenbaum section 10.1 covers layered protocols.

For Recitation, Thursday, February 26:

There is only one paper assigned for today, but it is Birrell and Nelson's classic paper on the remote procedure call (RPC) (reading #14). Study the paper carefully. We will discuss it in detail during recitation. The goal of RPC is to make a call to a remote server look like a regular procedure call. For fun, think about the ways in which RPC and regular procedure calls differ.

For Lecture, Monday, March 2:

In preparation for lecture, read the second part of Halstead's "6.033 notes on networking on communication" (reading #13). The optional reading in Tanenbaum chapter 9 provides more coverage of this material.

System aphorism of the week

A complex system that works is invariably found to have evolved from a simple system that works" (J. Gall, Systemantics).
