SIENA*college*
Computer Science

Computer Science 330
Operating Systems
Siena College
Spring 2012

Lab 4: Concurrency

Due: 9:20 AM, Friday, February 24, 2012

You have two short programs to develop for this week's lab. You may work alone or with a partner on these programs.

Process Interleavings

Write a C program that will list all possible orderings of the machine instructions generated for the critical sections of the Producer-Consumer example from class. Recall that the statements `counter++` and `counter--` actually generate machine code such as

	Producer		Consumer
P_1	`R0 = counter;`	C_1	`R1 = counter;`
P_2	`R0 = R0 + 1;`	C_2	`R1 = R1 - 1;`
P_3	`counter = R0;`	C_3	`counter = R1;`

Your program should list all possible interleavings of the statements P_1, P_2, P_3, C_1, C_2, and C_3. Also have your program print which interleavings produce a correct result (that `counter` has the same value it started with).

Write your program in a file called `interleaving.c`.

Bounded Buffer with Semaphores

Write a C program that implements the bounded buffer problem for an arbitrary number of producers and consumers and items to be processed (each specified by command-line parameters). If i items are to be processed by p producers and c consumers, each producer should produce $\frac{i}{p}$ items and each consumer should consume $\frac{i}{c}$ items. You may either check that these divide evenly and report an error if not, or account for the uneven division by having some producers or consumers process one extra or one fewer item.

Use POSIX threads to create your producers and consumers and use POSIX semaphores for synchronization. Your solution should avoid the busy wait from the class examples (other than any busy waiting that might takes place inside a semaphore `wait` or `signal` operation, which is not your fault).

Write your program in a file called `prodcons.c`. You may (and should!) use the source code from any of the class examples as your starting point. Be sure to indicate clearly which code you borrow and which code you add or modify.

See the examples and man pages for more information on POSIX threads and POSIX semaphores. Please don't hesitate to ask questions!

Submission and Evaluation

This lab will be graded out of 20 points.

By 9:20 AM, Friday, February 24, 2012, submit documented source code and a `Makefile` to allow easy compilation. All necessary files should be submitted by email to *jteresco@siena.edu*.

Grading Breakdown	
`Makefile`(s)	1 point
All interleavings considered	3 points
Interleavings generated rather than listed	2 points
Simulation of instructions for each interleaving	3 points
Arbitrary numbers of producers/consumers/items	3 points
Correct and efficient usage of POSIX semaphores	5 points
Documentation	3 points