CS 5334/4390 Spring 2015
Shirley Moore, Instructor
Midterm 1 Review Guide

Principles of Concurrency

The midterm will be closed notes, closed computer. You are not expected to memorize syntax. If you are asked to write code, either the syntax for the functions will be given, or pseudocode that illustrates the concepts correctly will suffice.

For the midterm exam, you should be able to do the following:

1. Explain the difference between concurrency and parallelism.

2. Compare and contrast processes and threads.

3. Give an example of how using concurrency can improve the average service time of a server that runs on a single processing core (i.e. without any parallelism).

4. Write a C program that correctly forks and manages child processes.

5. Explain the concepts of coroutines and continuations. Explain Scheme code that implements continuations.

6. Analyze sequential consistency with respect to a multithreaded code.
   a. Given the assumption of sequential consistency, determine the possible results from a given multithreaded code.
   b. Given a result from executing a given multithreaded code, draw a conclusion (if possible) about whether the underlying system enforces sequential consistency.

7. Correctly enforce mutual exclusion to a shared data structure using locks.

8. Define data race. Analyze a multi-threaded code with respect to data races.

9. Define deadlock. Analyze a multi-threaded code with respect to deadlock and starvation.

10. Define linearizability. Analyze a multi-threaded code that access a shared data structure with respect to linearizability.

11. Explain the concept of software transactions, also called transactional memory. Give an example where using transactions could enable greater concurrency than using locks.

12. Compare and contrast shared memory versus message passing for communication between threads. Give an example of a library or language for multithreaded programming that uses each.

13. Describe the Erlang programming and communication models. Explain an example Erlang program.

14. Describe the Cilk programming model. Convert a sequential recursive program to a concurrent Cilk program. Draw the execution tree for an example Cilk program and analyze the execution with respect to critical path length and parallel runtime.

15. Describe the concepts of deterministic parallelism and futures. Explain an example program that uses Java Futures.