Concurrent Programming Languages
COMP 623A
McGill University, Fall 2005

Course Details

Time: Monday, Wednesday 1:05pm–2:25pm
Place: McConnell Eng 103
Instructor: Professor Clark Verbrugge
Office: McConnell, room 230
Office hours: Wednesday 9:00am–10:30am, Friday 11:30am–1:00pm
Phone: 398-2411
Email: clump@cs.mcgill.ca

Teaching Assistant: Grzegorz Prokopski
Office: McConnell, room 234
Office hours: Thursday 10:00am–1:00pm
Email: gadek@debian.org

Email, Website

Students are expected to monitor their McGill email address for course-related news and information.
The course website is: http://www.cs.mcgill.ca/~cs623

Pre-requisites

Students should have a solid understanding of basic data structures, operating systems concepts, and
programming languages. There is a non-trivial programming requirement; ability to program in C or Java
will be required.

Textbooks

Supplementary texts: Multithreaded Programming with Java Technology by Lewis and Berg or Multithreaded Programming with PThreads by Lewis and Berg. These are optional texts with additional
information on multithreaded programming. Each of these supplementary texts covers much the same
material, but emphasizes either Java or POSIX (PThreads). Class examples will be primarily in Java, but
unless otherwise stated students may complete assignments using either environment/language. The last
assignment requires Java.

There is no required text for this course. Lectures (and some supplemental handouts) will be your primary
reference source.

Resources: The following texts are available on reserve in the Schulich library, as further reference for
some of the concepts covered in class:

• Lewis & Berg. Multithreaded Programming with Java Technology. Covers Java’s threading model
  and API’s as well as PThreads.
• Norton & Dipasquale. Threadtime: The Multithreaded Programming Guide. Covers POSIX (PThreads)
  as well as Java. Excellent guide to PThreads (now out of print).
• Andrews. Foundations of Multithreaded, Parallel, and Distributed Programming. Covers basics of
  concurrency primitives and constructs, numerous concurrency problems.
Description

Students will learn basic and advanced means for expressing and analyzing concurrency. This includes practical implementation issues and designs, as well as more abstract concepts or paradigms presented using a wide variety of theoretical formalisms and languages. The course includes practical experience with concurrent programs, and both Java and POSIX (PThreads) environments will be discussed. This course will be useful to students interested in theoretical aspects of concurrency, as well as those interested in correctly exploiting it in practice. Upon completion of the course students should have a good understanding of concurrent programming constructs, theoretical formalisms for concurrency and associated problems/issues.

Evaluation

3 Assignments: 45%
Project Proposal: 5%
Project Report: 50%

Assignment and Project Policy: Assignments must be submitted on time. Late assignments will only be accepted in highly-exceptional circumstances and only with written permission of the instructor. No assignment submissions will be accepted after marked assignments have been returned, or after solutions have been discussed in class.

McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offenses under the Code of Student Conduct and Disciplinary Procedures (see http://www.mcgill.ca/integrity/ for more information). Work submitted for this course must represent your own efforts. Copying assignments or tests, or allowing others to copy your work, will not be tolerated. Note that introducing syntactic changes into a copied program is still considered plagiarism.

Course Content

Note: schedule is approximate and should be used as a rough guideline only; some topics may shift dates.

Week 1:
- Introduction
- Hardware
- Atomicity & Independence
- Mutual Exclusion

Week 2:
- Basic concurrency constructs

Week 3:
- Java, POSIX (PThreads)
- Scheduling, Termination, Deadlock
Week 4:
- Consistency
- The Java memory model (JSR-133)
- Lock design

Week 5:
- Glitches
- Async vs sync
- Message passing

Week 6–7
- CSP, CCS
- Labelled Transition Systems
- Equivalence

Week 7–8:
- II-calculus
- Ambient calculus

Week 8–9:
- Dataflow, streams
- Dataflow expressiveness

Week 9–10:
- Petri Nets

Week 10–11:
- Graph Grammars, concurrent rewrite systems

Week 11–12:
- “True” Concurrency
- SpMT

Week 12–13:
- CHAM
- Linda
- L-Systems