Implementing, Optimizing, and Compiling Concurrent Languages
COMP 599

McGill University, Fall 2012

Course Details

Time: Tuesday, Thursday, 9:35am–10:55am
Place: ENGMC 103

Instructor: Professor Clark Verbrugge
Office: McConnell, room 230
Office hours: Tuesday 11:00–12:30, Friday 10:00–11:30, or by appointment.
Phone: 514-398-2411
Email: `clump@cs.mcgill.ca`

Email, Website

Students are expected to monitor their McGill email account for course-related news and information. The course website is: `http://www.sable.mcgill.ca/~clump/comp599`

Pre-requisites

- An undergraduate OS course, such as COMP 310 (Computer Systems and Organization) *or* ECSE 427 (Operating Systems).
- An interest in design and implementation of concurrent languages.

Previous experience or courses in concurrent programming, parallel programming, or advanced operating systems is helpful, but is neither required nor assumed.

Textbook

There is no required text for this course. For basic issues, the following text is recommended, but material will be primarily drawn from research papers:
The Art of Multiprocessor Programming (Revised First Edition) by Maurice Herlihy and Nir Shavit.

Description

This course will focus on concerns, problems, and techniques related to the implementation of modern, shared-memory concurrent programming languages. This includes consideration of core parallel programming idioms, safety concerns, implementation design, and efficiency. The course will focus on practical, low-level implementation issues, but will also include discussion of theoretical properties and programming models.
Upon completion of the course, students should have a good understanding of current and research-based concurrent programming models and their related implementation, correctness, and efficiency concerns.

Evaluation

3 presentations:	50%
Participation :	10%
Project proposal :	5%
Project report:	35%

In accord with McGill University's Charter of Students' Rights, students in this course have the right to submit in English or in French any written work that is to be graded.

Project and Presentation Policy: All work must be submitted on time. Late work will only be accepted in highly-exceptional circumstances and only with **written** permission of the instructor.

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).

More specifically, **work submitted for this course must represent your own efforts.** Copying course work, or allowing others to copy your work, will not be tolerated. Note that introducing syntactic changes into a copied program or project is still considered plagiarism.

Course Content

Note: changes to dates/topics will be announced in class.

Sept 6	Introduction
	Expressiveness
Sept 11, 13	Atomicity
	Mutual exclusion
	Presentation signups: Sept. 14
	Race conditions
	Synchronization
Sept 18, 20	Lock design
	Linearization
	Deadlocks
	Dependency
Presentations begin	
Sept 25, 27, Oct 2	Wait-freedom
	Lock-freedom
	Race detection
Oct 4, 9, 11	Memory consistency
	Memory models: Java, C++
	Consistency concerns
	Project Proposals due Oct.15
Oct 16, 18, 23	Concurrent languages
	PGAS languages
Oct 25, 30, Nov 1	Work-stealing
	Automated locking
	Transactional programming
Nov 6, 8, 13, 15	Thread-level speculation
	Hardware speculation
	Software speculation & RVP
	Optimizing speculation
Nov 20, 22, 27	Optimistic parallelism
	Measuring parallelism
Nov 29, Dec 4	(Note: optional, as time permits)
	Process algebra
	True concurrency