Concurrent Programming
COMP 409

McGill University, Fall 2010

Course Details

Time: Tuesday, Thursday, 1:05pm–2:25pm
Place: ENGTR 2120

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

Teaching Assistant: Chris Dragert
Office: TBD,
Office hours:
Email:

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/comp409`

Pre-requisites

- COMP 251 (Data Structures and Algorithms).
- COMP 302 (Programming Languages and Paradigms).
- COMP 310 (Computer Systems and Organization) *or* ECSE 427 (Operating Systems).
- There is a non-trivial programming requirement; ability to program in C or Java will be required.

Note: students registering without the pre-requisites may find the course removed from their transcript by their Faculty. Please consult the instructor if you do not have all the pre-requisites.

Textbook

There is no required text for this course. There are two recommended texts, and a few further books on reserve in the Schülich library that are not essential but may be useful if you need additional references.

Recommended: *Synchronization algorithms and concurrent programming* by *Gadi Taubenfeld* and *Java concurrency in practice.* by *Brian Goetz et al.*

Others: *Foundations of Multithreaded, Parallel, and Distributed Programming.* by *Gregory Andrews.* This serves as additional basic introduction to some of the main concurrent programming problems. *Multithreaded Programming with PThreads* by *Lewis* and *Berg*. This texts serves as an introduction and reference for programming in POSIX threads (PThreads). A further, more detailed book on PThread programming is *Threadtime* by *Norton* and *Dipasquale* (but is now out of print).

Class examples will be primarily in Java, but unless otherwise stated students may complete assignments using either the C/C++ and PThreads environment/language, or Java. The last assignment will require use of Java.
The recommended texts are available in the bookstore. All texts are on reserve in the Schülich Library.

Description

Students will learn the fundamentals of programming for concurrency. This includes both theoretical foundations, and practical experience with multithreaded programs. Both Java and POSIX (PThreads) environments will be discussed in detail, and significant programming will be required.
Upon completion of the course, students should have a good understanding of concurrent programming constructs and how to use them effectively and correctly. Additionally, students will gain experience with several major theoretical concurrency modelling systems/paradigms.

Evaluation

4 Assignments:	40%
Midterm:	10%
Exam:	50%

Both the exam and midterm will be open-book. A supplemental exam (100%) will be held if required.
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.

Assignment and Exam 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).
More specifically, **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 or assignment is still considered plagiarism.

Course Content

Note: lecture counts are approximate and may shift. Readings from the various texts are indicated (Taubenfeld and Goetz are in bold).

Lectures 1–3	**Readings**
• Introduction.	**T1.1–1.2**[1], **G1,11.1–11.3**[2], A1[3]
• Hardware	LB$_P$16,17[4]
• Atomicity & Independence.	**T1.4**, **G2**, A2.1–2.5
• Intro to multithreading; Java & POSIX.	**T1.5**, **G3,4**, LB$_P$2–4, ND 1–4[5]
Lectures 4–7	
• Mutual Exclusion	**T2.1–2.2,2.4,4.1–4.3**, A2.8
• Locks & Barriers	**T2.3,3.2,5.1–5.2,5.8,15.2**, A3.1–3.4, LB$_P$6, ND5
Lectures 8–11	
• Semaphores (all the flavours)	**T4.6,8.1.3–8.1.4**, A4.1–4.3
• Condition Variables & Monitors	**T4.7,8.1.5–8.1.6**, A5.1, LB$_P$7
• Readers & Writers	**T8.2**, A4.4,5.4.3–5.4.5
Lectures 12–13	
• Concurrency problems	**T8.4–8.5**, A5.2
• Deadlock	**T7**, **G10**
• Scheduling & Priorities	LB$_P$5, ND6
Lectures 14–15	
• Termination & Suspension	**G7**, LB$_P$9, ND8
• TSD, Java & POSIX miscellany	LB$_P$8,10, ND9
Lectures 16–20	
• Memory Consistency	**G16**
• Correctness	
• The Java concurrency API	**G6,8,13**
• **Midterm: Thursday, November 4 (in-class)**	
• Concurrent data structures	**T4.5**, **G5,14**
Lectures 21–23	
• Message passing	A7.1–7.5
• Formal models: process algebra	A7.6
Lectures 24–26	
• Alternative concurrent language designs	

[1] ie Taubenfeld book sections 1.1 through 1.2.
[2] ie Goetz book chapters 1, and 11.1 through 11.3.
[3] ie Andrews book chapter 1.
[4] ie Lewis & Berg book, chapters 16 and 17.
[5] ie Norton & Dipasquale, chapters 1 through 4.