SYSC 3303 Real-Time Concurrent Systems

Fall 2016
Course Introduction

Dr. Lynn Marshall
lynnmar@sce.carleton.ca

Course Objectives

- To introduce students to the principles and practice of software development for systems that are are characterized by one or more of the following terms: *real-time*, *concurrent*, *event-driven*, and *embedded*.
- Although a specific implementation technology will be used to provide hands-on programming experience, the goal is to present techniques that are applicable to a diverse range of applications, hardware/software components, programming languages and operating systems.

Lecture, Lab, and Office Hours Schedule

- Section A: Dr. Lynn Marshall, Room **ME4230**, lynnmar@sce.carleton.ca
  - 24 Lectures: Mon/Wed 4:05-5:25pm: **ME4499**: Wed Sept 7th to Wed Dec 7th (although Fri Dec 9th will follow a Monday schedule, the current plan is no lecture that day)
  - 11 Labs Mon/Tue 9:35-11:25am: **CB5109;** Mon 1:35-3:25pm: **AA508**: Mon Sept 12th to Tue Dec 8th (we may use the time between 3:25 and 3:55pm on the Monday afternoon lab; no lab Tue Oct 11th; we may use lab and lecture time on Fri Dec 9th, if needed)
  - You may use any SCE lab whenever it is not reserved
  - Office Hours: Mon 1:35-3:25pm in AA508 and by appointment

Lecture, Lab, and Office Hours Schedule

- The "Term Calendar" document on the course web site has a summary of all lectures, labs, deliverables, etc., and should be studied closely!

Prerequisites

- Engineering students must have credit for:
  SYSC 2003
  and
  SYSC 2004 (CSE / Comm Eng) **or** SYSC 2100 (SE)
- Computer Science students must have credit for
  COMP 2003 **or** COMP 2401
  and
  COMP 2002 **or** COMP 2402
- BIT and U of Ottawa: special requirements

Prerequisites

- No prerequisite waivers
- Students without the prerequisites must withdraw from the course
- Please come and see me if there are issues with your pre-reqs!

Background Knowledge

- Experience has shown that students who do well in this course:
  - are competent at developing sequential, object-oriented programs in C++ or Java
  - have a solid understanding of fundamental abstract data types (e.g., bags, sets, lists, queues, stacks, maps) and elementary data structures (e.g., arrays, linked lists, trees)

Background Knowledge

  - have a good understanding of the principles of computer system organization, as provided by SYSC 2001 or COMP 2003
  - have some knowledge of concurrency, mutual exclusion and condition synchronization, as provided by:
    - SYSC 2003 (background threads, interrupt service routines, shared buffers), or
    - SYSC 3001, COMP 3000 (concurrent processes/threads/tasks, semaphores)

Background Knowledge

  - are open to the idea that we can use pictorial modelling languages to reason about concurrent real-time programs as systems composed of interacting components (some concurrent, some not), at a level of abstraction above the coding details
    - we assume that you are familiar with the UML for modelling small-scale sequential programs
    - we'll introduce elements of the UML for modelling concurrent systems

Background Knowledge

- The course material will be presented at a level that is appropriate for a senior undergraduate course
  - we assume that you have completed all the required software courses in the first two years of your program, including courses not in the direct prerequisite chain, and *that you are competent at developing sequential programs that apply the concepts taught in these courses*
  - you should be comfortable designing, coding, and testing a 1000 line program *on your own.*

Background Knowledge

  - if you discover gaps in your background knowledge in software design, coding, testing and debugging, it is *your* responsibility to do the extra work required to obtain this knowledge

Textbook, Reference Books, & Lecture Slides

- No required textbook
- Supplementary references are listed in the course outline
- Lecture slides are / will be posted on the Web site as PDF files
- **Additional material that is not on these slides will be presented in class**, so students are cautioned that downloading the slides is not a substitute for attending lectures

Web Site

- http://www.sce.carleton.ca/courses/sysc-3303/f16
- Parts of the site are password protected
  - Userid / password will be announced in class
- Please, do not make this information publicly available

Contacting the Course Instructor via E-mail

- The instructor and TAs are only permitted to reply to e-mail from accounts originating at Carleton (e.g., cmail accounts, engsoc accounts, ieee accounts, and accounts in the sce.carleton.ca and scs.carleton.ca domains)
- E-mail filters are used to screen e-mail originating off-campus
- Please put the course (SYSC 3303) and topic in the subject line of your e-mail , as I'm also teaching ECOR 2606

Getting Advice & Assistance

- Instructor office hours are given in this presentation and are posted on the course Web site
  - meetings at other times can sometimes be arranged
- Questions and comments of general interest (e.g., items related to the lectures, assignments & project) can be e-mailed to your instructor; however, these will normally be answered in the next class, not via e-mail
- "Private" questions (e.g., PMC accommodations, marks, illness, etc.) can be dealt with via e-mail or by booking an appointment

Getting Advice & Assistance

- Each team will have formal 20 minute approximately bi-weekly meetings with a TA during the lab, as per the term calendar, and lab meeting schedule.
- Parts of those labs and the labs when no formal meetings are scheduled are available for help with assignments and the project.

Computer Accounts

- Every student registered in at least one SYSC course with computing requirements has an SCE lab account
  - with a few exceptions, one account for all courses
- Students without accounts (primarily Ottawa U students)
  - accounts will be created early in the term from the class registration list
  - if you have not enrolled in this course, please do so ASAP, so that your account will be created

Computer Accounts

- Information about creating your SCE account is posted on the web page
  - Please report any difficulties/problems ASAP
- Assignments and project iterations will be submitted using the "submit" program
  - Details are provided on the web site and will be discussed in class near the first assignment due date
  - Note that you must run Windows or a Windows emulator to use the "submit" program.

Java IDE

- For ease of marking, you must use the **Eclipse** Java IDE for all programming assignments and the project

Evaluation

- You will be evaluated by means of a project (done in teams of 4 or 5), assignments, a midterm exam and a final exam
- You must pass the project and the final exam
- Assuming the above, your final grade will be calculated using these weights:
  - 5 assignments: 10% (2% each)
  - midterm: 15%
  - project: 35% (10% for iterations/demo, 20% for final deliverables, and 5% for meeting participation)
  - final exam: 40%

Bonus Marks

- You can earn up to 2 bonus marks by attending workshops at the Centre for Student Academic Support. Details here: http://carleton.ca/csas/incentive-program/.
  - You will receive one bonus mark for each workshop attended (to a maximum of 2 marks)
  - I recommend:
    - Working in Groups
    - Effective Presentations
- You can also earn up to 3 bonus marks by presenting your Assignment #3 (paper summary) in class.

Assignments

- There will be five assignments each worth 2%
- Assignments will be graded out of 10
- Note that Assignment #3 is a 5 page summary of an article (newspaper, magazine, web, conference, …) on any topic relating to the course
- *Here's what we're looking for:* are your work products of the caliber that we would normally expect a student to produce during 3rd year?

Assignments

- assignment grading will take into account:
  - Assignments #1, 2, 4, 5 (programming, diagrams)
    - algorithm correctness (especially correct handling of concurrency issues)
    - programming style and documentation
    - design documentation and diagrams
    - result correctness
    - accurate notation
  - Assignment #3 (paper summary)
    - accuracy of summary
    - English (spelling and grammar)

Assignments

- Assignments are worth 10% of your final grade
- Assignments are intended as a way for you to learn the course material and to learn from your mistakes without concerns about whether difficulties will affect your final grade
- ***Do the assignments*:** the practice is invaluable preparation for subsequent assignments, the project and the exams

Project

- Project will be done in teams of 5 (with some groups of 4 if necessary)
  - **e-mail me by 8pm Tue Sept 13th with your list of team members**
    - **all team members must be able to attend the same lab section (or at least part of it)**
  - those who do not send an e-mail will be assigned a team
  - teams and meeting schedule will be e-mailed to you and posted on the web site on **Wed Sept 14th**
  - the first team meeting with a TA will be in the lab on **Mon Sept 19th / Tue Sept 20th**

Project

- Students who refuse to join a team, or do not participate with their team-mates, will receive a project mark of 0 and a final grade of FND
- If it is apparent that not all team members participated equally, adjustments will be made to each team member's project mark

Midterm Exam

- A closed-book midterm will be held on **Wed Oct 12th** (during the lecture)
  - the exam will not be rescheduled if you have another midterm exam at the same time (because you've enrolled in another course with lectures that conflict with SYSC 3303)
  - the University does not consider two or three midterms in the same day to be an overload

Accommodations for Missed Deadlines

- If you miss a test or deadline for valid medical or compassionate reasons, please contact your instructor immediately via e-mail (or if that's not possible, as soon as you return to school) to arrange appropriate accommodations
- Medical notes must be dated within one day of the test or deadline and handed in within 5 business days

Final Exam

- A closed-book three-hour final exam will be held during the University's December examination period (December 10th to 22nd).
- With the exception of students who receive FND based on their refusal to participate in a project team, **and** those who did not write the midterm (or make-up essay), all students are eligible to write the final examination.
- Those who have less than 40% on the midterm and **miss** the final exam, will receive FND and thus be ineligible to apply to the Registrar's Office for deferral of the final examination

Final Exam

- To pass the final exam, students must:
  - obtain at least a 50% average on the clearly identified "Core Programming" questions
  - obtain at least a 50% average on the entire exam
- Understanding the theory part of the course material isn't enough - you must demonstrate at least a minimal level of competence in writing multithreaded programs

Final Exam

- The final exam is for evaluation purposes only and will not be returned to students
  - See the course outline for more details
- Deferred final exams: see the current Undergraduate Calendar, *Academic Regulations of the University*, Section 2.2, The Course Outline; Section 2.3, Standing in Courses/Grading System; and Section 2.5, Deferred Final Examinations

Accommodations

- The Faculty of Engineering requires students to have a conflict-free timetable, so requests to accommodate missed exams, assignment due dates, project milestones, etc. because of conflicts with other courses, jobs or vacation plans will not be considered
- Students with disabilities - see the course outline, and the current Undergraduate Calendar, *Academic Regulations of the University*, Section 2.9
- Students with religious obligations - see the current Undergraduate Calendar, *Academic Regulations of the University*, Section 2.10

Overview of the Course

- Principles of Concurrent Programming
  - threads: creation and execution, communication and synchronization, life cycle, scheduling
  - modelling concurrent systems with the UML
  - primary programming technology will be Java

Overview of the Course

- Soft Real-Time Systems
  - characteristics, design techniques, application of the techniques to the area of computer communications protocols
  - project: design and implementation of a concurrent, real-time, distributed system

Overview of the Course

- Advanced Topics
  - introduction to the theory of hard real-time systems and scheduling
  - very large scale software development and true multi-processing systems
  - other applications of real-time concurrent systems development
  - theory only, no hands-on programming

Systems Thinking, Not Programming, is Key

- You can't pass this course through programming skill alone
  - even sequential programming "experts" sometimes have difficulties with this course
- The "theory part" is important; i.e., using analytical techniques to ensure that time-critical systems meet their deadlines
- Another important skill to develop is learning how to think about and design concurrent, real-time, distributed systems, using design notations to help visualize their structure and behaviour

…But Don't Ignore the Programming Part

- Programming helps novice real-time system developers understand the run-time behaviour implied by the design diagrams
  - analogy: would you be be able to use the UML effectively to model large-scale, sequential OO programs if you hadn't first learned how to write OO programs in Java, C++, Smalltalk, etc.?
  - similarly, how can you model concurrent, event-driven systems using abstract design notations if you don't have hands-on experience observing the run-time behaviour implied by the abstractions?

Programming Requires a Paradigm Shift

- The difficulties in developing a concurrent, real-time system arise because we can’t code the desired overall system behaviour directly
  - behaviour emerges at run-time through the interaction of cooperating autonomous components
  - designing the components to work together is the challenge