

---

---

# ***SYSC 3303 Real-Time Concurrent Systems***

## **Summer 2014 Course Introduction**

**Dr. Lynn Marshall**  
**[lynnmar@sce.carleton.ca](mailto:lynnmar@sce.carleton.ca)**

- Copyright © 2014 L.S. Marshall, Systems and Computer Engineering, Carleton University
- revised May 6<sup>th</sup>, 2014

---

---

## ***Course Objectives***

- To introduce students to the principles and practice of software development for systems that 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](mailto:lynnmar@sce.carleton.ca)
  - 12 Lectures: Mon/Wed 10am-1pm: **AA508**
  - 10 Labs Mon/Wed 1-3pm: **AA508**:
    - 5 Weekly team meetings (mandatory)
    - 3 Assignment help sessions (optional)
    - 2 Quizzes (mandatory)
  - Project demo (schedule TBD)
  - You may use any SCE lab whenever it is not reserved
  - Office Hours: Mon/Wed 2:15-3:00pm 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 by the last date for withdrawal from late Summer term courses, otherwise they will be deregistered before the end of term
- 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 2.5 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*
  - if designing, coding, and testing a 1000 - 5000 line program *on your own* represents a significant challenge for you, you shouldn't be here!

---

## ***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 but Burns and Wellings is the recommended optional text
- 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/s14>
- Parts of the site are password protected
  - Userid and Password will be announced in the first lecture
- 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 1606

---

---

## ***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 a weekly meeting with a TA, as per the term calendar
- In addition, there will usually be an optional lab for help a few days before each assignment is due
  - You may also work on or ask for help with the project during this time



---

## ***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 logging in to your SCE account and changing your password is given on the background screen of the machines in the undergrad labs
  - Please report any difficulties/problems ASAP
- Programs will be submitted using the “submit” system
  - Details are provided on the web site and will be discussed in class near the first assignment due date

---

## ***Java IDE***

- You may use any Java IDE available in the labs, e.g. JCreator or Eclipse
  - Note that JBuilder is not recommended

---

---

## ***Evaluation***

- You will be evaluated by means of a project (done in teams of 3 or 4), assignments, quizzes, 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)
  - 2 quizzes: 10% (5% each)
  - midterm: 15%
  - project: 25% (20% for final deliverables and 5% for meeting participation)
  - final exam: 40%

---

---

## ***Quizzes***

- There will be two quizzes, each worth 5%
- Details on what is covered on each quiz will be discussed in the lectures.

---

---

## ***Assignments***

- There will be five assignments each worth 2%
- Assignments will be graded out of 4
- Note that #2 is a 3-4 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 a co-op term or internship following 3rd year?

---

## ***Assignments***

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

---

## ***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 must be done in teams of 4 (with some groups of 3 if the class size is not divisible by 4)
  - **e-mail me by 8pm Thu May 8<sup>th</sup> with your list of team members**
  - those who do not send an e-mail will be assigned to a team
  - teams and meeting schedule will be e-mailed to you and posted on the web site on **Fri May 9<sup>th</sup>**

---

## ***Project***

- Students who refuse to join a team 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 **Mon May 26<sup>th</sup>** (during the second half of the regular lecture time)
  - 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 an assignment due date or project milestone 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 certificates are not required for a missed assignment or project milestone; however, if you do not contact us about it in a timely manner (see above point) the missed component will be considered to be "late without a valid reason" and receive 0

---

## ***Final Exam***

- A closed-book three-hour final exam will be held during the University's June examination period (June 20-26<sup>th</sup>).
- 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***

- At least two of the questions on the final exam will be clearly labelled "Core Programming", and will be used to evaluate students' understanding of the fundamental programming skills taught in this course
- To pass the final exam, students must:
  - obtain at least a 50% average on the 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 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