Introduction.
What are concurrent systems?
SE 3BB4

Ryszard Janicki

Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

Basic Information

Instructor: Dr. Ryszard Janicki, ITB 217, e-mail: janicki@mcmaster.ca, tel: 525-9140 ext: 23919

Teaching Assistants:

- Atiyeh Sayadi: atiye.sayadi@yahoo.com
- Nancy Kansal: kansan1@mcmaster.ca
- Heywon Jo: joh6@mcmaster.ca
- Ahmad Al Labadi : allabada@mcmaster.ca

Course website: `http://www.cas.mcmaster.ca/~se3bb4`,

Lectures: Monday: 2:30pm-3:20pm, in MDCL 1102, Wednesday: 2:30pm-3:20 pm, in MDCL 1102, Friday: 4:30pm - 5:20pm, MDCL 1102
Tutorials: Tuesday: 8:30am-10:20am, T13 107, Friday: 8:30am - 10:20am, BSB B103, Friday: 2:30pm - 4:20pm, T13 107,start September 9, 2025
Office hours: Monday: TBA
Midterm: The week of October 20 – 24, 8:00 pm, take home, submission via Avenue, details later.

Course Details

Calendar Description:
Processes, threads, concurrency; synchronization mechanisms, resource management and sharing; objects and concurrency; design, architecture and testing of concurrent systems.

Mission:
The mission of this course is to give students an understanding of the basic techniques of concurrent system design and to introduce students to the appropriate modelling and analysis techniques.

Learning Objectives: Preconditions

Preconditions:
Students should have basic knowledge of discrete mathematics (especially of logic, sets and relations), basic knowledge of data structures and algorithms. They should be familiar with principal ideas of software development and know how to program in at least one programming language.

Learning Objectives: Postconditions

Postconditions:

A. Students should know and understand:

1. Basic models of concurrency, both algebraic and graphical
2. The concepts of process and thread
3. Resource management and sharing
4. Basic ideas of design and architecture of concurrent systems
5. Problems with testing and verification of concurrent systems

B. Students should be able to:

1. Specify concurrent systems with process algebras and Petri nets.
2. Implement concurrent systems with processes, threads and other available tools.
3. Verify and test crucial parts of concurrent systems.

Outline and Texts

Course Outline (Tentative):
Nature of Concurrency. Processes and Threads. Concurrent Execution. Labelled Transition Systems. Shared Objects and Mutual Exclusion. Monitors and Condition Synchronization. Deadlock. Safety and Liveness Properties. Model Based Design. Dynamic Systems. Message Passing. Concurrent Architectures. Timed Systems. Program Verification. Logical Properties. Petri Nets and other models.

Texts:
J. Magee, J. Kramer, Concurrency: State Models and Java Programming, 2nd Edition, J. Wiley 2006.

The course will not always follow the text-book closely.

Evaluation

- **Evaluation:** The will be a 2.5 hours (in person) final examination (50%), 60 minutes midterm test (20%, take home) and three assignments ($3 \times 10 = 30\%$).
- **Detailed grading scheme:** $Grade = 0.50 \times exam + 0.2 \times midterm + 0.1 \times (assg1 + assg2 + assg3)$
- *Late assignments will not be accepted.*
- Although you may discuss the general concept of the course material with your classmates, your assignment must be your individual effort.

What are concurrent systems?

Definition (Concurrent Systems)

The term concurrent systems is usually employed in reference to systems consisting of several parts acting together.

Sequential vs Concurrent.

- A sequential program has a single thread of control.
- A concurrent program has multiple threads of control allowing it perform multiple computations in parallel and to control multiple external activities which occur at the same time.

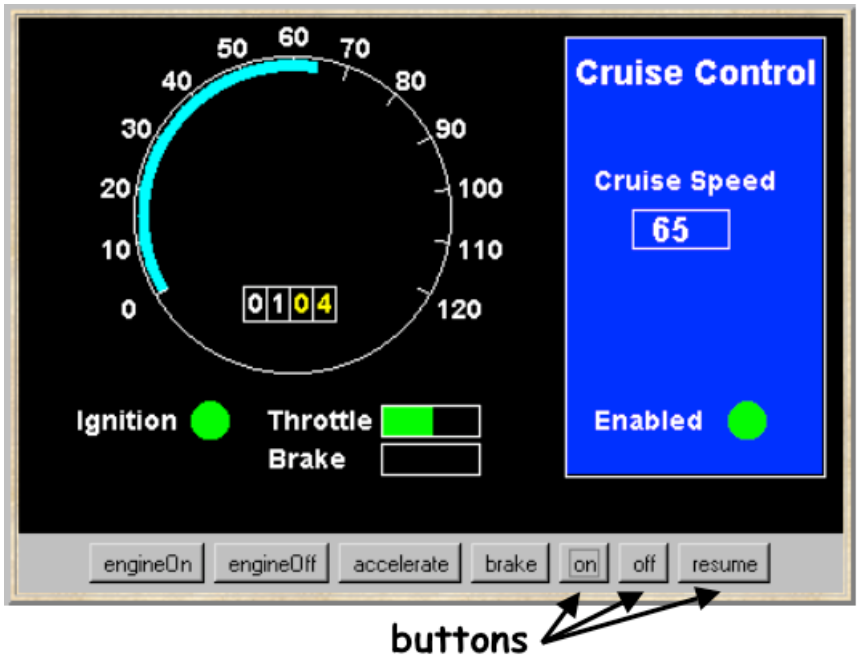
Why concurrent programs?

- Performance gain from multiprocessing hardware (parallelism).
- Increased application throughput (an I/O call need only block one thread).
- Increased application responsiveness (high priority thread for user requests).
- More appropriate structure for programs which interact with the environment, control multiple activities and handle multiple events.

Why do I need to know about concurrent programs?

- Concurrency is widespread but error prone.
  - Therac-25 computerised radiation therapy machine. Concurrent programming errors contributed to accidents causing deaths and serious injuries.
  - Mars Rover. Problems with interaction between concurrent tasks caused periodic software resets reducing availability for exploration.

A cruise control system



When the car ignition is switched on and the *on* button is pressed, the current speed is recorded and the system is enabled; it maintains the speed of the car at the recorded setting. Pressing the *brake*, *accelerator* or *off* button disables the system. Pressing *resume* enables the system.

Questions:

- Is the cruise control system safe?
- Would testing be sufficient for discovering all the errors?

Models

Definition (Model)

A model is a simplified representation of the real world.

- Software designers and engineers use models to gain confidence in the adequacy and validity of a proposed design. They are useful to:
  - focus on an aspect of interest (concurrency).
  - model animation to visualize a behaviour.
  - mechanical verification of properties (safety and progress).
- In this course, models are described using state machines, known as Labelled Transition Systems LTS, and Petri nets. The former are described textually as finite state processes (FSP, a kid of a process algebra) and displayed and analysed by the LTSA analysis tool.

Course objective

- This course is intended to provide a sound understanding of the concepts, models and practice involved in designing concurrent software.
- The emphasis on principles and concepts provides a thorough understanding of both the problems and the solution techniques.
- Modeling provides insight into concurrent behavior and aids reasoning about particular designs.
- Concurrent programming in Java provides the programming practice and experience.

Course outline

Basics

- Processes and Threads.
- Concurrent Execution.
- Shared Objects and Interference
- Monitors and Condition Synchronization.
- Deadlock.
- Safety and Liveness Properties.
- Model-based Design Concepts.

Advanced

- Dynamic systems.
- Message Passing.
- Concurrent Software Architectures.
- Timed Systems.
- Program Verification.
- Logical Properties.

Summary

- Concepts: we adopt a model-based approach for the design and construction of concurrent programs.
- Models: we use finite state and Petri nets models to represent concurrent behavior.
- Practice: we use Java for constructing concurrent programs.