Course 02158

Introduction

Hans Henrik Løvengreen

DTU Compute

Fall 2024

Concurrent Programming

What for?

- Systems with *simultaneous activities*

Examples

- Several apps open on a mobile (e.g. media player + gps navigation)
- Multiple users accessing a database
- Several computation parts taking place on a multi-core machine
- Multiple players in a single computer game

Concurrent Programming

Why?

1. Because the real world is *parallel*:

   many users, many devices $\Rightarrow$ many tasks to do

   By reflecting this parallelism within programs we may get:

   - better *structure*
   - better *response* times
   - higher *performance* (on multi-processors)

2. Because we have *parallel machine architectures*:

   network connected computers, multi-processor computers, GPUs $\Rightarrow$ many hands to keep busy

   Programs must be *(re-)structured* to exploit these architectures

Caveat

- Many pitfalls: Race conditions, deadlocks, starvation

Concurrent Programming

How?

- *Language constructs* for expressing concurrency
- *Mechanisms* for synchronization and communication
- Principles of concurrency *implementation*
- *Models* for reasoning about concurrent behaviour
- Techniques for *proving* and *testing* concurrent programs
- Principles for good *design* of concurrent systems

Concurrent Programming Aims

After the course you should

- Understand *concepts* and *notions* of concurrency.
- Know *abstract models* of concurrency and principles of *verification*
- Be well versed in synchronization and communication *mechanisms*
- Know about underlying *implementation principles*
- Be aware of concurrency *pitfalls* and principles for avoiding them
- Know how to *test* concurrent programs
- Be skilled in writing *multi-threaded Java* programs
- Know a number of concurrency *SW-architectures*
- Identify when and how to *apply concurrency*

Concurrent Programming Prerequisites

Required

- Good command of *sequential Java*
- Good knowledge af *algorithms* and *data structures*
- Discrete mathematics, including *predicate logic* ($\forall$, $\Rightarrow$)

Useful

- General knowledge of *programming language notions*
- Basic knowledge of *machine architecture*
- Basic knowledge of *program representation*
- Knowledge of *databases*
- Knowledge of *automata* (state machines)
- Knowledge of *program semantics*

Concurrent Programming Material

- G. Andrews: *Foundations of Multithreaded, Parallel, and Distributed Programming*. Textbook
- H.H. Løvengreen: *Basic Concurrency Theory*. Note.
- Other notes and auxiliary material will be available online:

www.compute.dtu.dk/courses/02158

Course Plan Fall 2024

Week	Day	Lecture	Ex/Lab
1	Sep 5	Introduction, Petri Nets	Ex. class 1
2	Sep 12	Processes, threads & tasks, atomic actions, interleaving model	Lab. 1
3	Sep 19	Transition systems, safety and liveness, critical regions, invariants	Assignm. 1
4	Sep 26	Temporal logic, fairness, SPIN	Ex. class 2
5	Oct 3	Barriers, semaphores	Lab. 2
6	Oct 10	Semaphore techniques, monitors	Assignm. 2
		Fall Vacation	
7	Oct 24	Monitor techniques	Ex. class 3
8	Oct 31	Testing, deadlocks	Lab. 3
9	Nov 7	Message passing	Assignm. 3
10	Nov 14	Remote operations	Ex. class 4
11	Nov 21	Parallel computing, concurrency paradigms	Lab. 4
12	Nov 28	Advanced topics	Assignm. 4
13	Dec 5	Ending	Ex. class 5

Core topics
Supplementary topics
Lab activities

Concurrent Programming Evaluation Fall 2024

Four Mandatory Assignments (pass/no pass)

- Test ability to apply concepts and notions in practice
- Every third week, set $1\frac{1}{2}$ weeks in advance
- Carried out in *groups of 2-3 students*
- Must be documented by brief *reports*
- Three out of four assignments must be passed for attending exam
- Failed assignments may be resubmitted

Written Exam

- Tests understanding of concepts and notions
- Monday **December 9**
- 4 hours, only non-electronic aids allowed, i.e. *no computers*