ID1217 Concurrent Programming 7.5 credits

Programmering av parallella system

This is a translation of the Swedish, legally binding, course syllabus.

If the course is discontinued, students may request to be examined during the following two academic years

Establishment

Course syllabus for ID1217 valid from Spring 2019

Grading scale

A, B, C, D, E, FX, F

Education cycle

First cycle

Main field of study

Information Technology,Technology

Language of instruction

The language of instruction is specified in the course offering information in the course catalogue.

Intended learning outcomes

The overall aim of this course is provide the necessary knowledge in programming models, concepts, techniques, synchronization and communication mechanisms, and environments used in concurrent programming with threads and processes, i.e. in multithreaded, parallel and distributed programming.

After completion of the course the student should have a good understanding of the problems and solution strategies of process-oriented programming.More specifically, after completion of the course, students should be able to:

Know concurrent programming concepts (e.g. task, process, thread, process state, context switch, etc.), models (e.g. shared memory, message passing) and paradigms (e.g. iterative parallelism, recursive parallelism, producers and consumers, pipelines, dataflow, clients and servers, peers); understand interleaving semantics and non-determinism of concurrent execution.

- Analyze a concurrent program with shared variables, e.g. count and construct possible histories of the program and its possible results; reason about a concurrent program: whether it is correct, whether it terminates, whether it exposes race conditions; prove a safety property, such as partial correctness, using axiomatic semantics of concurrent programs with shared variables.
- Know how concurrent threads can be synchronized. Know and evaluate different solutions to the critical section problem. Understand and apply mutual exclusion and condition synchronization in multithreaded programs with shared variables. Understand, choose and use different synchronization mechanisms, such as locks, condition variables, barriers, semaphores and monitors.
- Develop an outline of a concurrent program with shared variables using mutual exclusion and condition synchronization to synchronize threads. Develop and use monitors with condition variables in a multithreaded program.
- Design, develop and implement concurrent programs with shared variables using existing concurrent programming environments such as pthreads in C, Java threads and monitors, Java concurrent utilities.
- Understand and use parallel algorithms in concurrent programs.
- Know how distributed processes can communicate with each other. Understand, choose and use different communication mechanisms, such as asynchronous and synchronous message passing, Remote Procedure Calls and Rendezvous, Remote Method Invocations in concurrent programs with distributed processes.
- Develop an outline of a distributed program using message passing, RPC, rendezvous or RMI for inter-process communication.
- Design, develop and implement distributed programs with processes using a distributed programming environment such as Socket API or Massage Passing Interface (MPI) in C, Sockets or Remote Method Invocation (RMI) in Java.
- Measure and estimate speedup which can be achieved due to parallel execution.
- Know operating system support for processes, blocking synchronization mechanisms such condition variables and semaphores in a single-processor and in a shared memory multiprocessor; support for message passing in a distributed memory platform.

Course contents

Concurrent programming with threads and shared variables.
Processes and synchronization.
Critical sections, locks, barriers, semaphores and monitors.
Case studies: threads in Java, Pthreads.
Parallel and distributed programming with processes.
Message passing, RPC, RMI and rendezvous. Case study: Java RMI.
Paradigms for process interaction.
An overview parallel and distributed programming environments such as MPI, PVM and OpenMP. Performance issues.

Specific prerequisites

Completed upper secondary education including documented proficiency in Swedish corresponding to Swedish B and English corresponding to English A. For students who received/will receive their final school grades after 31 December 2009, there is an additional entry requirement for mathematics as follows: documented proficiency in mathematics corresponding to Mathematics A. And the specific requirements of mathematics, physics and chemistry corresponding to Mathematics D, Physics B and Chemistry A.

Course literature

Foundations of Multithreaded, Parallel, and Distributed Programming, Gregory R. Andrews, Addison-Wesley, 2000; ISBN 0-201-35752-6.
Kursböcken kan i framtiden komma att ersättas av annan litteratur.
Eget material.

Examination

- LABA - Laboratory Work, 3.0 credits, grading scale: P, F
- TEN1 - Examination, 4.5 credits, grading scale: A, B, C, D, E, FX, F

Based on recommendation from KTH's coordinator for disabilities, the examiner will decide how to adapt an examination for students with documented disability.

The examiner may apply another examination format when re-examining individual students.

Other requirements for final grade

Written examination (TEN1;4,5 hp) and two programming assignments (LABA;3.0 hp)

Ethical approach

- All members of a group are responsible for the group's work.

- In any assessment, every student shall honestly disclose any help received and sources used.
- In an oral assessment, every student shall be able to present and answer questions about the entire assignment and solution.