Linnæus University

Dnr: LNU-2025/771

# Course syllabus

Faculty of Technology
Department of Computer Science and Media Technology

1DT911 Systemprogrammering, 5 högskolepoäng
Systems Programming, 5 credits

**Main field of study**
Computer Engineering

**Subject**
Informatics/Computer and Systems Sciences

**Level**
First cycle

**Progression**
G1F

**Date of Ratification**
Approved 2025-03-03.
The course syllabus is valid from autumn semester 2025.

**Prerequisites**
Programming and Data Structures (1DT910), 7.5 credits, Databases and data modeling (1DT903), 5 credits and Object-oriented programming (1DT904, 5 credits or 1DT905, 7,5 credits) or equivalent.

## Objectives

After completing the course the student shall be able to:
*Knowledge and understanding*

- A.1 Explain problems that can occur when using shared resources such as locks, starvation and race conditions
- A.2 explain some common methods for handling locks (e.g., semaphores), including their properties and limitations

- A.3 explain the parts required to create runnable programs, such as build system, compiler, and linker
- A.4 explain common APIs used to access system services, such as POSIX
- A.5 explain the difference between shared memory and message passing, and
- A.6 explain the difference between parallelism, concurrency, and asynchronous execution.

*Competence and skills*

- B.1 Implement concurrent applications for a computer with shared memory, using threads and locking or asynchronous execution
- B.2 use build systems to build software
- B.3 develop programs that uses a system API, such as POSIX
- B.4 analyze a problem and implement a suitable concurrent solution that is correctly synchronized.

*Judgement and approach*

- C.1 Choosing and reasoning about whether shared memory, message passing, or asynchronous execution is appropriate for a given problem.

## Content

The course introduces concurrent programming and the problems this entails, e.g., locks and race conditions. Different ways to deal with these problems, e.g., locking algorithms and message handling, are discussed and their limitations. The content and algorithms are exemplified using threads, asynchronous execution, or similar.
The following topics are covered:

- Processes and synchronization
- Scheduling
- Shared memory and messages
- Concurrent programming with threads and shared variables
- Critical sections
- Locks, barriers, semaphores, and monitors
- Asynchronous programming
- Build systems
- The compiler and the steps required to create a program
- Systems APIs, such as POSIX
- Unit testing

## Type of Instruction

The instruction consists of lectures and teacher-led laboratory work. The programming labs are carried out in pairs.

## Examination

The course is assessed with the grades A, B, C, D, E or F.

The grade A constitutes the highest grade level, the remaining grades follow in descending order where the grade E constitutes the lowest grade level for passing. The grade F means that the student's performance has been assessed as failed.

Assessment of student performance is made through programming assignments and a written exam.

To pass the course, grade E or higher is required for all parts. The final grade is decided from: written exam (60%) and programming assignments (40%).

Resit examination is offered in accordance with Linnaeus University’s Local regulations for courses and examination at the first- and second-cycle levels.
In the event that a student with a disability is entitled to special study support, the examiner will decide on adapted or alternative examination arrangements.

## Objectives achievement

The examination of the course is divided as follows:
Module 2501 Written exam 3.0 credits with the grading system AF
Module 2502 Programming assignment 2.0 credits with the grading system AF

The examination elements are linked to the course objectives in the following ways:
Module 2501 links to the course objectives: A.1, A.2, A.3, A.4, A.5, A.6, B.4, C.1
Module 2502 links to the course objectives: A.2, B.1, B.2, B.3, B.4, C.1

## Course Evaluation

A course evaluation should be conducted during the course or in connection with its conclusion. The results and analysis of the completed course evaluation should be promptly communicated to students who have completed the course. Students participating in the next course instance should be informed of the results of the previous course evaluation and any improvements that have been made, no later than at the start of the course.

## Overlap

The course cannot be included in a degree along with the following course/courses of which the content fully, or partly, corresponds to the content of this course:
1DT906, 5 credits, 1DT909, 5 credits

## Other Information

Grade criteria for the AF scale are communicated to the student through a special document. The student is to be informed about the grade criteria for the course by the start of the course at the latest. The course is conducted in such a way that the course participants' experiences and knowledge are made visible and developed. This means, for example, that we have an inclusive approach and strive for no one to feel excluded. This can be expressed in different ways in a course, for example by using the gender-neutral example.

## Required Reading and Additional Study Material

Required reading:

- Herlihy, Maurice. och Shavit, Nir., *The Art of Multiprocessor Programming*, Morgan Kaufmann, latest edition. Pages: 400 of 536.