

System Programming (C003776)

Course size *(nominal values; actual values may depend on programme)*

Credits 6.0

Study time 180 h

Course offerings and teaching methods in academic year 2026-2027

A (semester 2)

Dutch

Gent

seminar

group work

lecture

Lecturers in academic year 2026-2027

De Turck, Filip

TW05

lecturer-in-charge

Volckaert, Bruno

TW05

co-lecturer

Offered in the following programmes in 2026-2027

[Bachelor of Science in Computer Science](#)

6

A

[Bachelor of Science in Mathematics](#)

6

A

[Master of Science in Bioinformatics\(main subject Systems Biology\)](#)

6

A

Teaching languages

Dutch

Keywords

Procedural programming, C programming language, hybrid programming language, C++ , software design paradigms.

Position of the course

After prior introductory courses in programming, the objective of this course is to widen the knowledge and understanding of programming languages, as well as to treat the basic principles of paradigms for software design in a more generic way : procedural, object oriented with particular attention to memory-safe programming and software development for embedded Linux environments.

The major objective is to give an extended overview of current paradigms for software engineering. It fits in the bachelor curriculum since programming techniques are put in a broader context. Students are prepared for later courses which are more about development methodology and also for carrying out realistic development projects in the final phase of the bachelor curriculum.

Contents

1 Procedural programming (using C) and hybrid languages (using C++).

This implies following skills:

- using pointers
- reference types versus value semantics
- operator overloading
- inheritance, both single and multiple
- separation of interface and implementation, illustrating a weakness of C++
- generic programming and data abstraction
- the STL library
- exception handling
- moderne C++ additions as threading and move semantics

2 Paradigms of software design

- an introduction to the Rust programming language for memory-safe programming based on the ownership model
- platforms for support of large software projects with multiple programmers

- Version Systems
- Generation of Makefiles
- overview of currently important software technologies

3 Lab-assignments (individually and in group), aimed at the use of programming languages like C and C++, and programming in inbedded Linux environment.

Initial competences

- A good knowledge of at least one programing language, preferably an object oriented language (Java or Python)
- Initial experience with the basic principles of object orientation
- Some knowledge of computer architecture
- Experience with using a computer
- These objective are met by the courses "Programming" and "Object Oriented Programming" which come earlier in the curriculum

Final competences

- 1 A good view on several paradigms for software design.
- 2 A good knowledge of several programming languages.
- 3 Insight in the available platforms for support of large projects.
- 4 An overview of currently important software technologies.

Conditions for credit contract

Access to this course unit via a credit contract is determined after successful competences assessment

Conditions for exam contract

This course unit cannot be taken via an exam contract

Teaching methods

Group work, Seminar, Lecture

Study material

Type: Slides

Name: System programming

Indicative price: Free or paid by faculty

Optional: no

Language : Dutch

Number of Slides : 500

Available on Ufora : Yes

Online Available : Yes

Available in the Library : No

Available through Student Association : No

References

- A Book on C, vierde editie (Al Kelley, Ira Pohl), ISBN: 90.430-0497.9 (English version)
- De programmeertaal C++ (Bjarne Stroustrup), ISBN: 90.430-0231.3 (optional)
- The Rust Programming Language (Steve Klabnik, Carol Nichols, and Chris Krycho, with contributions from the Rust Community), ISBN: 9781718504448 (optional)

Course content-related study coaching

An e-learning environment (Ufora) including Git-environments.

Dodona for extra exercises

Lab sessions : assistants are available to help the students during these sessions and give feedback on the submitted solutions

Teacher and assistants can be reached by e-mail and online discussion fora

Assessment moments

end-of-term and continuous assessment

Examination methods in case of periodic assessment during the first examination period

Skills test

Examination methods in case of periodic assessment during the second examination period

Skills test

Examination methods in case of permanent assessment

Skills test, Assignment

Possibilities of retake in case of permanent assessment

examination during the second examination period is possible in modified form

Extra information on the examination methods

- During examination period: written open-book exam. Usage of internet or generative AI during the examination is not permitted and is treated as fraud.
- During semester: graded lab-assignments (code and documentation)

Calculation of the examination mark

Combination of NPE (25% of final grade) and PE (75% of final grade). In case of a clearly different amount of contributions from the team members, the score of the students from the same team can be different.

In case a score of less than 9/20 is obtained for the examination, the student can not pass the course. If the total score would then be 10/20 or more, the total score will be reduced to 9/20.