



Language :	English > French-friendly
Place of the course	Louvain-la-Neuve
Main themes	<p>For a long time, general-purpose CPUs focused on supporting efficiently a single thread of execution. Improvements in chip manufacturing allowed packing more transistors on the same surface of a silicon wafer, and run resulting CPUs at higher frequencies. Single-threaded applications would simply run faster with every new processor generation. This era is now over. The industry hit several limitations known as the power wall, the memory wall and the ILP wall. No longer able to vertically scale-up CPUs supporting a single thread of execution, chip manufacturers have started packing together multiple, simpler units of execution, or cores. Exploiting the power of multiple cores requires exploiting parallelism in applications using multiple threads. Writing concurrent code requires identifying and managing concurrency, and introducing the necessary synchronization for correctness. Writing scalable and performant concurrent code requires understanding this tradeoff between synchronization and parallelism and mastering efficient implementations of shared data structures and algorithms for concurrent execution. Finally, multicore CPUs employ a complex memory layout, and the assumption of uniform memory access times is no longer valid. Understanding the impact of non-uniform memory accesses (NUMA) is therefore important to write efficient code for multicore CPUs.</p> <p>This course will provide students with the necessary tools and knowledge to write efficient and scalable code for modern multicore CPUs. It will detail the mechanisms available for synchronization, starting from the implementation of language constructs such as locks, monitors or condition variables, to the direct uses of CPU-provided synchronization primitives (e.g. compare-and-swap) to build efficient and scalable data structures. It will emphasize the performance aspects of multicore programming: the impact of synchronization primitives, the impact of non-uniform memory access, and the impact of multiple-level memory hierarchies. It will finally offer an opening to the future of multicore programming with an introduction to transactional memory and to the support for speculative execution in modern CPUs (e.g. Intel Haswell), and discuss the execution model for concurrent code running on GPUs.</p>
Learning outcomes	<p>At the end of this learning unit, the student is able to :</p> <p>Given the learning outcomes of the "Master in Computer Science and Engineering" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:</p> <ul style="list-style-type: none"> • INFO2.1-4 • INFO4.1-4 • INFO5.1-3 • INFO6.2-4 <p>Given the learning outcomes of the "Master [120] in Computer Science" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:</p> <ul style="list-style-type: none"> • SINF1.M3 • SINF2.1-4 • SINF4.1-4 • SINF5.1-3 • SINF6.2-4 <p>1 Given the learning outcomes of the "Master [60] in Computer Science" program, this course contributes to the development, acquisition and evaluation of the following learning outcomes:</p> <ul style="list-style-type: none"> • 1SINF1.M3 • 1SINF2.1-4 • 1SINF3.1-4 • 1SINF4.1-3 • 1SINF5.2-4 <p>At the outcome of this course, the students will have acquired the necessary competences to build a large-scale software system under semi-professional working conditions. More specifically, students having completing this course with success will be able to:</p> <ul style="list-style-type: none"> • Describe the differences among several major process models (e.g., waterfall, iterative, and agile); • Differentiate among the phases of software development (specification, architecture, design, implementation, validation, documentation); • Complete, in a rigorous and systematic way, the artefacts produced in these different software life cycle phases;

	<ul style="list-style-type: none"> • Apply a software development methodology currently practiced in industry; • Work efficiently in a team to develop a medium-to large-scale software system; • Manage the coordination and communication between the different team members; • Interact with a client to identify his requirements, to clarify imprecise specifications, and to take into account requested modifications throughout the development process; • Describe the functional requirements of a software system using, for example, use cases or users stories; • Estimate the time and resources needed to complete such a software development project, plan the tasks to be executed and the deliverables to be produced, and respect this planning; • Use some project management tool to assign and follow the planned software development tasks; • Put in practice different methods and techniques to assure the quality of the produced software; • Understand the problems inherent to the development of large software systems having different stakeholders and that consist of multiple components.
Evaluation methods	<p>Course evaluation will be based on:</p> <ul style="list-style-type: none"> • Individual participation in group work and weekly group meetings with an assistant or tutor, who plays the role of project manager; • The realization of three intermediate prototypes with corresponding technical reports; • The final report, the delivered system and its documentation, as well as a presentation and demonstration of the final product to the customer. • The integration of the groups' solutions. <p>Please note: As this course is based on participation in a team project throughout the year, the project notes will automatically be kept in the second session.</p>
Teaching methods	<p>Development (analysis, design, implementation, validation, documentation, integration and deployment) of a large software system for a client, in teams of 6 to 8 students supervised by a project manager. Weekly meetings will be held with the project manager, and different prototypes and reports will need to be produced throughout the project.</p>
Content	<p>This software engineering project consists of the development (analysis, design, implementation, validation, documentation, integration and deployment) of a realistic and non-trivial software application, if possible proposed by and with the participation of a real client, under semi-professional working conditions. The topic of the application to be constructed is proposed by an industrial partner or a non-profit organization that participates in the organisation of this course.</p> <p>Teams of 6 to 8 students (required to achieve such a large project), will collaborate, supervised by a project manager.</p> <p>Weekly meetings will be held with the project manager (an assistant or tutor) to present the progress and difficulties encountered, to evaluate alternatives, and discuss the distribution and planning of the work among team members.</p> <p>The application to be developed will most likely be a web application, but the choice of the programming language, the environment, the application framework, and the development tools will depend on the requirements of the project client.</p>
Inline resources	<p>http://moodleucl.uclouvain.be/course/view.php?id=7599</p>
Bibliography	<p>French Des lectures supplémentaires seront suggérées dans le plan de cours qui décrit les produits livrables et l'organisation du projet. Les supports de cours pertinents, des slides et des informations pratiques seront accessibles sur Moodle, qui sera également le principal moyen de communication entre l'enseignant et les étudiants.</p> <p>English Additional reading material will be suggested in the course plan which describes the deliverables and organisation of the project. All relevant course material, slides and practical information will be available on Moodle, which will also be the main means of communication between the teacher and the students.</p>
Other infos	<p>Prerequisites:</p> <ul style="list-style-type: none"> • Have good knowledge of and experience with the concepts of object-oriented programming, algorithms and data structures. • Have participated in the development of a small to medium-sized software system.
Faculty or entity in charge	<p>INFO</p>

Programmes containing this learning unit (UE)				
Program title	Acronym	Credits	Prerequisite	Learning outcomes
Master [120] in Computer Science and Engineering	INFO2M	6		
Master [120] in Computer Science	SINF2M	6		
Master [60] in Computer Science	SINF2M1	6		