Minor Deep Programming 2017-2018
Hier vind je de beschrijvingen van de vakken in de minor. Meer inhoudelijke informatie over de minor vind je op minor.vu.nl.
### Inhoudsopgave

| Vak: Compiler Construction (Periode 3) | 1 |
| Vak: Concurrency & Multithreading (Periode 2) | 2 |
| Vak: Equational Programming (Periode 2) | 3 |
| Vak: Secure programming (Periode 1) | 4 |
| Vak: Systems Programming (Periode 1) | 5 |
Doel vak
Get hands-on experience with the internals of modern compilers.

Inhoud vak
This course gives an introduction to the internals of modern compilers and allows students to gather practical, hands-on experience on building compiler components with weekly assignments. The course will focus on the design of all the major components of a modern compiler pipeline: frontend (focus on lexing and parsing), intermediate representation or IR (focus on optimizations and transformations), and backend (focus on code generation). The course emphasises what compiler builders encounter in practice rather than the details of all manners of parsing algorithms. Specifically, the assignments will focus on building a simple compiler for a C-like language using the LLVM compiler framework. All the assignments build on top of the given framework and require localized extensions in Python and/or C++ to implement language features, optimizations, transformations, and code generation features.

Note: this is a very intensive, hands-on course. It is important to be able to start immediately. For this, you need access to a computer with the VirtualBox x86 virtualization environment installed.

Onderwijsvorm
Lectures, Practical

Toetsvorm
Assignments

Aanbevolen voorkennis
Basic knowledge of Python and C/C++ is strongly recommended.
**Doelvak**

This course provides a comprehensive presentation of the foundations and programming principles for multicore computing devices.

Specific learning objectives are:

* To provide insight into fundamental notions of multicore computing and their relation to practice: locks, read-modify-write operations, mutual exclusion, consensus, construction of atomic multi-reader-multi-writer registers, lost wakeups, ABA problem.

* To provide insight into algorithms and frameworks for multicore computing and their application in multi-threaded programs: mutual exclusion algorithms, spin locks, monitors, barriers, AtomicStampedReference class in Java, thread pools in Java, transactional memory.

* Analyzing algorithms for multicore computing with regard to functionality and performance: linearizability, starvation- and wait-freeness, Amdahl’s law, compute efficiency gain of parallelism.

* Mastering elementary datastructures in the context of multicore computing: lists, queues, stacks.

* Programming in multi-threaded Java, and performing experiments with such programs.

**Inhoud vak**

The course consists of the following topics: Shared memory, mutual exclusion, synchronization operations, concurrent data structures, scheduling, transactional memory, and a multithreaded programming assignment.

**Onderwijsvorm**

4 hours per week HC, 4 hours per week WC.

**Toetsvorm**

The written exam counts for 75% and the programming assignment for 25% of the final mark.

Both for the written exam and the programming assignment at least a 5.0 must be obtained (and the overall average mark should be at least 5.5).

Only students that achieved at least a 3.0 for their initial programming
assignment are offered a resit opportunity for this assignment.

Literatuur

Aanbevolen voorkennis
Datastructures & Algorithms
Programming in Java

Doelgroep
3CS

Overige informatie
The homepage of the course is at http://www.cs.vu.nl/~tcs/cm/

The lectures and written exam of the BSc and MSc variant of Concurrency & Multithreading coincide. The difference is that the BSc variant has a smaller programming assignment than the MSc variant.

Equational Programming

<table>
<thead>
<tr>
<th>Vakcode</th>
<th>X_401011 (401011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periode</td>
<td>Periode 2</td>
</tr>
<tr>
<td>Credits</td>
<td>6.0</td>
</tr>
<tr>
<td>Voertaal</td>
<td>Engels</td>
</tr>
<tr>
<td>Faculteit</td>
<td>Faculteit der Exacte Wetenschappen</td>
</tr>
<tr>
<td>Coördinator</td>
<td>dr. F. van Raamsdonk</td>
</tr>
<tr>
<td>Examinator</td>
<td>dr. F. van Raamsdonk</td>
</tr>
<tr>
<td>Docent(en)</td>
<td>dr. F. van Raamsdonk</td>
</tr>
<tr>
<td>Lesmethode(n)</td>
<td>Hoorcollege, Werkcollege, Practicum</td>
</tr>
<tr>
<td>Niveau</td>
<td>300</td>
</tr>
</tbody>
</table>

Doel vak
To obtain basic knowledge of functional programming (using Haskell) and its foundations via lambda calculus and equational reasoning.

Inhoud vak
In the practical work we use the functional programming language Haskell.
We practice with the basics such as lists, recursion, data-types, and a bit of monads.

The theoretical part is concerned with the foundations of functional programming in the form of lambda calculus and equational reasoning. We study in untyped lambda calculus beta reduction, reduction strategies, confluence, encoding of data-types, fixed point combinators and recursive functions. In addition we study the lambda-calculus with simple types, its typing system and a type inference algorithm, and possibly strong normalization of simply typed lambda-calculus.
In equational reasoning we work towards the results that all initial models are equal up to isomorphism, and that the term model is an initial model.
Onderwijsvorm
The theoretical part is taught in the lectures and exercise classes.
In addition, there is a programming lab for programming in Haskell.

Toetsvorm
Written examination,
programming assignments in Haskell,
and (possibly obligatory) hand-in theory exercises.

Literatuur
Course notes.

Aanbevolen voorkennis
It helps to be familiar with formal reasoning as for example taught in
the course Logic and Modeling.

Doelgroep
3CS, 3LI, 3IMM, 3W

Intekenprocedure
The registration procedure is the standard one.

Overige informatie
This course is part of the minor Deep Programming.

Secure programming

<table>
<thead>
<tr>
<th>Vakcode</th>
<th>XB_40005 ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periode</td>
<td>Periode 1</td>
</tr>
<tr>
<td>Credits</td>
<td>6.0</td>
</tr>
<tr>
<td>Voertaal</td>
<td>Engels</td>
</tr>
<tr>
<td>Faculteit</td>
<td>Faculteit der Exacte Wetenschappen</td>
</tr>
<tr>
<td>Coördinator</td>
<td>dr. S. Rawat</td>
</tr>
<tr>
<td>Examinator</td>
<td>dr. S. Rawat</td>
</tr>
<tr>
<td>Lesmethode(n)</td>
<td>Hoorcollege</td>
</tr>
<tr>
<td>Niveau</td>
<td>300</td>
</tr>
</tbody>
</table>

Doel vak
This is an introductory course on information security. The emphasis
will be on how to develop applications with security in mind. At the
end of the course, students should have be familiar with the following:
1. Importance of security in modern engineering.
2. How common cryptographic primitives work, and why they are essential.
3. How bugs can degrade the security of software.
4. Common memory corruptions bugs and their (security) side-effects in
   software.

Inhoud vak
The course is devided into the following modules:

A. Understanding Cryptographic primitives.
   1. Confidentiality, Integrity and authentication (CIA) properties.
   2. Symmetric/asymmetric/stream ciphers.
3. Digital certificates/signatures.
5. OpenSSL engineering.

B. Understanding (and avoiding) low-level bugs.
1. Introduction to C (if needed) and assembly.
4. Integer overflow/format strings.
5. Bug detection and Mitigation
5. Secure Development lifecycle (SDL).
C. Special topics in Security (optional).

Onderwijsvorm
Lectures and practical assignments.

Toetsvorm
Written Exam (60%). Practical assignments (40%).

Literatuur
3. Online materials (articles)

Vereiste voorkennis
Knowledge of computer programming, preferably in C.

Aanbevolen voorkennis
Background in mathematics (number theory), working knowledge of web, python programming language.

Systems Programming

<table>
<thead>
<tr>
<th>Vakcode</th>
<th>X_400377 (400377)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periode</td>
<td>Periode 1</td>
</tr>
<tr>
<td>Credits</td>
<td>6.0</td>
</tr>
<tr>
<td>Voertaal</td>
<td>Engels</td>
</tr>
<tr>
<td>Faculteit</td>
<td>Faculteit der Exacte Wetenschappen</td>
</tr>
<tr>
<td>Coördinator</td>
<td>dr. A. Bakker</td>
</tr>
<tr>
<td>Examinator</td>
<td>dr. A. Bakker</td>
</tr>
<tr>
<td>Docent(en)</td>
<td>dr. A. Bakker</td>
</tr>
<tr>
<td>Lesmethode(n)</td>
<td>Hoorcollege, Werkcollege</td>
</tr>
<tr>
<td>Niveau</td>
<td>300</td>
</tr>
</tbody>
</table>

Doel vak
The goal of this course is to prepare students for lab assignments and scientific research in computer systems (operating systems, compiler construction, network programming, computer networks, parallel programming, etc.) After attending this course, students should be able to develop, test, and debug "systems" programs written in C under Linux or BSD.
Inhoud vak
The course is a combination of lectures and lab assignments.

During the course, the student is taught how to program in C, use POSIX APIs for process control and networking, understand memory management, use low-level debugging and verification tools, and use performance profiling tools.

Onderwijsvorm
7 lectures of 2h, in combination with several lab assignments to be returned during the study period. Extensive help will also be provided by ways of questions and answers sessions (1h), and a discussion mailing list.

Toetsvorm
The student will be graded based on the lab assignments he or she handed in (i.e., a Practicum). Exact grading scheme announced at the start of the course. There is a resit opportunity later in the year.

Vereiste voorkennis
- must have studied algorithms (incl. sorting, basic graph processing) and data structures (incl. lists, trees, priority queues);
- must have basic understanding of Unix concepts (directory tree, file permissions, terminal).

Aanbevolen voorkennis
Prior experience with another language from the C family (eg. Java, Arduino-C, C++, Objective-C, C# or D) is strongly recommended.

Doelgroep
3CS

Overige informatie
Registration for this course is also compulsory via Canvas one week before the start. The course will be given in English.
The coordinator and teacher of this course is Arno Bakker (arno@cs.vu.nl)