

General Information

Foundation 3: Compiler Construction (FND3)

In this course we consider techniques, data structures, and algorithms for translating programming languages into executable code. A modern compiler is often organized into many phases, each on a different abstraction. These abstractions are words or tokens, parsed phrases or parse trees, abstract syntax tree and intermediate representations (IR). To practice the analysis of a language-based problem and to design and to realize a compiler, we consider structured documents. Thus we define a grammar representing the a certain subset of structured documents. For this set of structured document we finally implement the front-end of a compiler. The front-end of a compiler implements the lexical analysis by a scanning mechanism that groups symbols into certain strings so that it generates so-called tokens and builds a symbol table. Then parsing actions are performed to construct a hierarchical structure, a parsing tree (or derivation tree). The parse tree is enriched with attributes to prepare semantic analysis. The semantic analysis is processed for e.g. type checking of the variables/attributes and other methods, and the front-end finally generates intermediate code which should be executable by a machine or a certain environment, even if it is not optimized.

Requirements

- Visit of the weekly lectures (2 h)
- Joining the weekly practical lessons in the computer lab (2 h)
- Joining a group of two student to develop a (front-end of a) simple XML compiler (+ circa 2 h)

The list student of working groups is made in the second week during the practical lessons in the computer lab.

Assessment

- Exam after the first quarter (30%)
- Exam after the second quarter (30%)
- $(\text{Grade of the first exam} + \text{grade of the second exam}) \geq 5.5$
If you not reach an overall grade of the both exams that is greater or equal 5.5, then you can attend an repeating exam. This exam will take place after the complete semester.

- Final assessment on your simple XML compiler (40%, grade ≥ 3.5) that takes place at the end of the semester

Assessment on a simple XML compiler

To prepare the assessment on your simple XML compiler the following is at least needed

1. A short introduction to the functionality of your simple XML compiler
2. A documented version of your XML DTD
3. A finite automaton representing the design of a (sub-)part of your scanning mechanism
4. The documented input file of the jflex tool for generating a scanner
5. The documented input file of the byaccj tool for generating a parsing mechanism
6. A class diagram of your implementation
7. A running example of an XML document that can be processed by your compiling mechanism and its intermediate results of every phase of the compiling process (don't forget the parse tree)
8. A CD, including your compilable code of your simple XML compiler

Remarks:

- You as a student team can decide, whether you prefer to use the programming environment *eclipse* or *NetBeans* for your development.
- The generating tools for a scanner and a parser are often controlled from the command line.
- Take care that you prepare the requested documentation during the development phases in the computer lab (and not only at the end).
- The requirements listed above in 1. to 7. has to be submitted in a printed version.
- Don't forget to give your names!

Have success!