Rust programming language and its compilation
Abstract of presentation

Jan Wrona, Kateřina Žmolíková
{xwrona00,xzmoli02}@stud.fit.vutbr.cz

9. listopadu 2015

Rust is a relatively new programming language developed by Mozilla Research focused on safety, speed and concurrency. Due to its properties it is suitable for embedding in other languages, writing programs with special time and space requirements and also writing low level code. This presentation provides an overview of the most interesting characteristics of the language and the proccess of its compilation.

The key features of Rust language include systems of ownership, moves and borrows which are designed to ensure memory safety. These principles guarantee clear lifetime of each value which is checked at compile time and makes garbage collection unnecessary. Same ownership rules also apply to concurrency model and prevent data races at compile time.

The compiler of Rust is written in Rust itself. It can be divided into 6 phases - input parsing, configuration and expanding, semantic analysis, translation to LLVM module and linking. First steps are fairly standard, creating an abstract syntax tree using recursive descent. Semantic analysis is a very extensive part including checking of variable lifetimes, liveness or memory safety rules. The backend leans on LLVM library which takes care of the code generation.