Fachhochschul-Masterstudiengang
SOFTWARE ENGINEERING
4232 Hagenberg, Austria

MiniC++ Compiler with Java Technologies

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science in Engineering

Eingereicht von

Andreas Zauner, BSc

Betreuung: FH-Prof. DI Dr. Dobler Heinz
Begutachtung: FH-Prof. DI Dr. Dobler Heinz

Hagenberg, Juni 2025

software engineering
IKM Fakultät Hagenberg

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

This printed thesis is identical with the electronic version submitted.

Date Signature

Contents

Kurzfassung

Peer-to-Peer-Netzwerke bieten eine Alternative zum klassischen Client-Server-Modell, um Daten auszutauschen. In Peer-to-Peer-Netzwerken kommunizieren alle Clients miteinander. Dadurch kann auf den Server als zentrale Schnittstelle verzichtet werden. Diese Charakteristik ermöglicht es Peer-to-Peer-Netzwerken zu funktionieren, obwohl einzelne Teilnehmer im Netzwerk ausfallen. Zudem nutzen Peer-to-Peer-Netzwerke die (meist bei traditionellem Filesharing ungenutzte) Upload-Bandbreite der einzelnen Clients.

Diese Bachelorarbeit setzt sich detailliert mit Peer-to-Peer-Netzwerken auseinander. Dabei werden zuerst bekannte Peer-to-Peer-Netzwerke vorgestellt und deren Charakteristiken erläutert. Weiters wird gezeigt, wo Unternehmen und Organisationen Peer-to-Peer-Netzwerke einsetzen. Unterschieden wird hierbei zwischen frei verfügbaren Netzwerken und von Unternehmen eigens entwickelten Netzwerken. Abschließend wird ein Client für das Netzwerk BitTorrent entwickelt. Dieser Client ist in der Lage, unter Verwendung des BitTorrent-Protokolls eine Datei von anderen Peers herunter- und hochzuladen. Dadurch wird gezeigt wie der Datenaustausch in einem Peer-to-Peer-Netzwerk auf technischer Ebene funktioniert und welche Technologien dazu benötigt werden.

Abstract

Peer-to-peer networks offer an alternative to the classic client-server model for exchanging data. In peer-to-peer networks, all clients communicate with each other. This means that the server, as the central element, can be omitted. This characteristic enables peer-to-peer networks to function even if individual participants in the network fail. In addition, peer-to-peer networks use the upload bandwidth of the individual clients, which is usually unused in traditional file sharing.

This bachelor thesis deals in detail with peer-to-peer networks. First, known peer-to-peer networks are introduced and their characteristics are explained. Then it is shown where companies and organisations utilize peer-to-peer networks. A distinction is made between freely available networks and networks developed by companies themselves. Finally, a client for the BitTorrent network is developed. This client is able to exchange a file with other peers using the BitTorrent protocol. This shows how data exchange in a peer-to-peer network works on a technical level and which technologies are required for this.

Chapter 1

Introduction

1.1 Motivation

Compilers function as the backbone for computer programming. A compiler takes care of translating human-readable source code into something a computer can understand. This allows the application developer to focus only on writing the application, without having to worry about the technicalities of the concrete computer where the software will run on. For one programming language there may exist multiple compilers targeting different kinds of computers. This allows the same source code to run for example on Linux and Windows. This flexibility saves the developer a lot of work, because they don't need to rewrite their application in the case they also want to target another operating system. Furthermore, there also exist compilers that target virtual machines like the Java Virtual Machine (JVM). Generating code for a virtual machine has the advantage that there don't need to be compilers written for every target operating system. Instead, for each operating system an implementation of the virtual machine is provided.

The process of compiling source code begins in the frontend of the compiler. In this step the source code is read, and an abstract syntax tree (AST) is constructed. The AST is a runtime representation of the source code in memory. It contains only the necessary information that is later on needed to generate machine code. The process of constructing the AST is based the grammar of the programming language. Based on this grammar a lexer and parser are either written manually or get generated by a parser generator tool like ANTLR. In the case of ANTLR the generated parser and lexer construct a full parse tree from the input. From this an AST can be constructed using for example the visitor-pattern.

The AST functions then as the input for the backend of the compiler. In this section the machine code for the target system is generated. In the case of the JVM this is the so called Bytecode. The Bytecode could be written by hand, however this is rather difficult. For this a detailed understanding of the instruction set is needed, and the actual generation would have to be performed on a byte array. Therefore, APIs exist, that provide an abstraction layer to the code generation. One API for Bytecode generation is the open source project ObjectWeb ASM or just ASM. It provides an API that utilizes the visitor-pattern to generate Bytecode instructions.

References