

УДК 004.4'422LLVM
ББК 32.973.33
Л77

Л77 Бруно Кардос Лопес, Рафаэль Аулер
 LLVM: инфраструктура для разработки компиляторов. / пер. с англ.
 Киселев А. Н. – М.: ДМК Пресс, 2015. – 342 с.: ил.

ISBN 978-5-97060-305-5

LLVM – новейший фреймворк для разработки компиляторов. Благодаря простоте расширения и организации в виде множества библиотек, LLVM легко поддается освоению даже начинающими программистами, вопреки устоявшемуся мнению о сложности разработки компиляторов.

Сначала эта книга покажет, как настроить, собрать и установить библиотеки, инструменты и внешние проекты LLVM. Затем познакомит с архитектурой LLVM и особенностями работы всех компонентов компилятора: анализатора исходных текстов, генератора кода промежуточного представления, генератора выполняемого кода, механизма JIT-компиляции, возможностями кросс-компиляции и интерфейсом расширений. На множестве наглядных примеров и фрагментов исходного кода книга поможет вам войти в мир разработки компиляторов на основе LLVM.

Издание предназначено энтузиастам, студентам, а также разработчикам компиляторов, интересующимся LLVM. Читатели должны знать язык программирования C++ и, желательно, иметь некоторые представления о теории компиляции.

Original English language edition published by Published by Packt Publishing Ltd., Livery Place, 35 Livery Street, Birmingham B3 2PB, UK. Copyright © 2014 Packt Publishing. Russian-language edition copyright (c) 2015 by ДМК Пресс. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-78216-692-4 (англ.)
 ISBN 978-5-97060-305-5 (рус.)

Copyright © 2014 Packt Publishing
 © Оформление, перевод на русский язык,
 ДМК Пресс, 2015



ОГЛАВЛЕНИЕ

Об авторах	11
О рецензентах.....	12
Предисловие	14
Содержание книги.....	17
Что потребуется для работы с книгой	19
Кому адресована эта книга.....	19
Типографские соглашения	20
Отзывы и пожелания	21
Скачивание исходного кода примеров	21
Список опечаток.....	22
Нарушение авторских прав	22
Глава 1. Сборка и установка LLVM	23
Порядок нумерации версий LLVM.....	24
Установка скомпилированных пакетов LLVM	25
Установка скомпилированных пакетов с официального сайта.....	25
Установка с использованием диспетчера пакетов.....	27
Сборка из исходных текстов	28
Системные требования.....	28
Получение исходных текстов.....	29
Сборка и установка LLVM	30
Windows и Microsoft Visual Studio	37
Mac OS X и Xcode	41
В заключение	45
Глава 2. Внешние проекты.....	47
Введение в дополнительные инструменты Clang.....	47
Сборка и установка дополнительных инструментов Clang	49
Compiler-RT	50
Compiler-RT в действии.....	51
Расширение DragonEgg.....	52
Сборка DragonEgg	53

Конвейер компиляции с применением DragonEgg и инструментов LLVM	54
Пакет тестов LLVM	56
Использование LLDB.....	57
Введение в стандартную библиотеку libc++	59
В заключение	63
Глава 3. Инструменты и организация	64
Введение в основные принципы организации LLVM	64
LLVM сегодня	67
Взаимодействие с драйвером компилятора.....	71
Использование автономных инструментов.....	72
Внутренняя организация LLVM	75
Основные библиотеки LLVM.....	76
Приемы программирования на C++ в LLVM	78
Эффективные приемы программирования на C++ в LLVM.....	80
Демонстрация расширяемого интерфейса проходов.....	83
Реализация первого собственного проекта LLVM	84
Makefile.....	85
Реализация.....	87
Общие советы по навигации в исходных текстах LLVM	89
Читайте код как документацию	89
Обращайтесь за помощью к сообществу	90
Знакомьтесь с обновлениями – читайте журнал изменений в SVN как документацию	90
Заключительные замечания.....	93
В заключение	93
Глава 4. Анализатор исходного кода	94
Введение в Clang.....	94
Работа анализатора исходного кода	95
Библиотеки.....	97
Диагностика в Clang.....	100
Этапы работы анализатора Clang	105
Лексический анализ.....	105
Синтаксический анализ	112
Семантический анализ.....	119
Все вместе.....	122
В заключение	126
Глава 5. Промежуточное представление LLVM	127
Обзор.....	127
Зависимость LLVM IR от целевой архитектуры	130

Основные инструменты для работы с форматами IR	131
Введение в синтаксис языка LLVM IR	132
Представление LLVM IR в памяти	136
Реализация собственного генератора LLVM IR	139
Сборка и запуск генератора IR	143
Как генерировать любые конструкции IR с использованием генератора кода C++	144
Оптимизация на уровне IR	145
Оптимизация времени компиляции и времени компоновки	145
Определение проходов, имеющих значение	147
Зависимости между проходами	149
Прикладной интерфейс проходов	151
Реализация собственного прохода	152
В заключение	157
Глава 6. Генератор выполняемого кода.....	158
Обзор	158
Инструменты генераторов кода	161
Структура генератора кода	162
Библиотеки генераторов кода	163
Язык TableGen	165
Язык	167
Использование файлов .td с генераторами кода	168
Этап выбора инструкций	174
Класс SelectionDAG	175
Упрощение	178
Объединение DAG и легализация	179
Выбор инструкций с преобразованием «DAG-to-DAG»	181
Визуализация процесса выбора инструкций	184
Быстрый выбор инструкций	185
Планирование инструкций	186
Маршруты инструкций	186
Определение опасностей	188
Единицы планирования	188
Машинные инструкции	188
Распределение регистров	189
Объединение регистров	191
Замена виртуальных регистров	196
Архитектурно-зависимые обработчики	197
Пролог и эпилог	198
Индексы кадров стека	199
Инфраструктура машинного кода	199

Инструкции MC	200
Эмиссия кода	200
Реализация собственного прохода для генератора кода	203
В заключение	206
Глава 7. Динамический компилятор	208
Основа механизма динамической компиляции в LLVM	209
Введение в механизм выполнения	210
Управление памятью	212
Введение в инфраструктуру llvm::JIT	213
Запись блоков двоичного кода в память	213
JITMemoryManager	214
Механизмы вывода целевого кода	214
Информация о целевой архитектуре	215
Практика применения класса JIT	217
Введение в инфраструктуру llvm::MCJIT	222
Механизм MCJIT	223
Как MCJIT компилирует модули	224
Диспетчер памяти	227
Использование механизма MCJIT	228
Инструменты компиляции LLVM JIT	231
Инструмент lli	231
Инструмент llvm-rtld	232
Дополнительные ресурсы	233
В заключение	234
Глава 8. Кросс-платформенная компиляция	235
Сравнение GCC и LLVM	236
Триады определения целевой архитектуры	238
Подготовка инструментария	240
Стандартные библиотеки C и C++	240
Библиотеки времени выполнения	241
Ассемблер и компоновщик	242
Анализатор исходного кода Clang	242
Кросс-компиляция с аргументами командной строки Clang	244
Параметры драйвера, определяющие архитектуру	244
Зависимости	245
Кросс-компиляция	246
Изменение корневого каталога	248
Создание кросс-компилятора Clang	250
Параметры настройки	250
Сборка и установка кросс-компилятора на основе Clang	251
Альтернативные методы сборки	252

Тестирование	254
Одноплатные компьютеры	254
Симуляторы	255
Дополнительные ресурсы	255
В заключение	256

Глава 9. Статический анализатор Clang 257

Роль статического анализатора.....	258
Сравнение классического механизма предупреждений со статическим анализатором Clang	258
Возможности механизма символического выполнения.....	262
Тестирование статического анализатора	265
Использование драйвера и компилятора	265
Получение списка доступных средств проверки.....	266
Использование статического анализатора в Xcode IDE	268
Создание графических отчетов в формате HTML	269
Анализ больших проектов	269
Расширение статического анализатора Clang собственными средствами определения ошибок	275
Архитектура проекта	275
Разработка собственного средства проверки	277
Дополнительные ресурсы	287
В заключение	289

Глава 10. Инструменты Clang

и фреймворк LibTooling 290

Создание базы данных команд компиляции	290
Clang-tidy	292
Проверка исходного кода с помощью Clang-tidy	293
Инструменты рефакторинга	294
Clang Modernizer	295
Clang Apply Replacements.....	296
ClangFormat	298
Modularize	300
Module Map Checker.....	308
PPTrace	309
Clang Query	311
Clang Check	313
Удаление вызовов c_str().....	314
Создание собственного инструмента	314
Определение задачи – создание инструмента рефакторинга кода на C++	315
Определение структуры каталогов для исходного кода.....	315

Шаблонный код инструмента	317
Использование предикатов AST	321
Создание обработчиков	326
Тестирование нового инструмента рефакторинга	328
Дополнительные ресурсы	329
В заключение	329
Предметный указатель	331