

自制编译器

How to Develop a Compiler

[日] 青木峰郎 / 著 严圣逸 绝云 / 译

TURING

图灵程序
设计丛书



从零开始制作真正的编译器

贯穿编译、汇编、链接、加载的全过程！

比“龙书”更具实践性！



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS

TURING

图灵程序
设计丛书

自制编译器

[日] 青木峰郎 / 著 严圣逸 绝云 / 译



How to Develop a Compiler

人民邮电出版社
北京

图书在版编目(CIP)数据

自制编译器 / (日)青木峰郎著;严圣逸,绝云译

. -- 北京:人民邮电出版社,2016.6

(图灵程序设计丛书)

ISBN 978-7-115-42218-7

I. ①自… II. ①青… ②严… ③绝… III. ①C语言
—编译器—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2016)第083048号

内 容 提 要

本书将带领读者从头开始制作一门语言的编译器。笔者特意为本书设计了Cb语言,Cb可以说是C语言的子集,实现了包括指针运算等在内的C语言的主要部分。本书所实现的编译器就是Cb语言的编译器,是实实在在的编译器,而非有诸多限制的玩具。另外,除编译器之外,本书对以编译器为中心的编程语言的运行环境,即编译器、汇编器、链接器、硬件、运行时环境等都有所提及,介绍了程序运行的所有环节。

从单纯对编译器感兴趣的读者到以实用为目的的读者,都适合阅读本书。

◆ 著 [日]青木峰郎

译 严圣逸 绝云

责任编辑 乐 馨

执行编辑 杜晓静

责任印制 彭志环

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京天宇星印刷厂印刷

◆ 开本:800×1000 1/16

印张:29.5

字数:605千字 2016年6月第1版

印数:1-3 000册 2016年6月北京第1次印刷

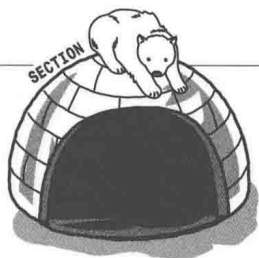
著作权合同登记号 图字:01-2014-5502号

定价:99.00元

读者服务热线:(010)51095186转600 印装质量热线:(010)81055316

反盗版热线:(010)81055315

广告经营许可证:京东工商广字第8052号



译者序

算上这本《自制编译器》，图灵的“自制”系列应该已经有 6 本了。从 CPU 到操作系统，从编译器到编程语言，再到搜索引擎等具体的应用，俨然已经可以自制一套完整的计算机体系了。

“自制”系列图书都是从日本引进并翻译出版的，本人也有幸读过其中几本。可能有很多读者和曾经的我一样对“自制”抱有疑惑：“在时间就是金钱、时间就是生命的 IT 行业，为什么会存在这样的自制风潮？为什么要自制呢？CPU 可以用 Intel、AMD，操作系统已经有了 Windows、Linux，搜索引擎已经有了 Google、Yahoo，编程语言及其对应的编译器、解释器更是已经百花齐放、百家争鸣……”直到翻译完本书，我才逐渐体会到自制是最好的结合实践学习的方式之一。拿来的始终是别人的，要吃透某项技术、打破技术垄断，最好的方法就是自制。并且从某种程度上来说，自制也是一种创新，可能下一个 Google 或 Linux 就孕育在某次自制之中。

自制编译器的目的是了解当前实用的编程语言、编译器和 OS 的相关知识，绝对不能闭门造车。因此作者使用的 Cb 语言是 C 语言的子集，实现了包括指针运算在内的 C 语言的主要部分，通过自制一个 Cb 语言的编译器，能够让我们了解 C 语言程序编译、运行背后的细节；OS 选用 Linux，能够让我们知晓 Linux 上的链接、加载和程序库；汇编部分采用最常见的 x86 系统架构。作者自制的编译器 cbc 能够运行在 x86 架构的任何发行版本的 Linux 上，编译 Cb 代码并生成可执行的 ELF 文件。

作者青木先生在致谢中提到了 Linux 和 GNU 工具等开源软件的开发。这也是本书的另一大特色：充分利用开源软件和工具。从 GCC 到 GNU Assembler 再到 JavaCC 以及 Linux，并非每一行代码都是自己写的才算自制，根据自己的设计合理有效地利用开源软件，既可以让我们更快地看到自制的成果，又能向优秀的开源软件学习。如果要深入学习、研究，那么开源软件的源代码以及活跃的社区等都是非常有帮助的。而如果把自制的软件也作为开源软件上传到 Github 上供大家使用，并根据其他开发者提出的 Pull Request 不断改进软件，那就更好了。

最后我要由衷地感谢本书的另一位译者绝云老师以及图灵的编辑。还要特别感谢我的外公，一位毕生耕耘于教育出版行业的老编辑。自己能有幸参加翻译，和从小对出版工作的耳闻目染是密不可分的。

严圣逸

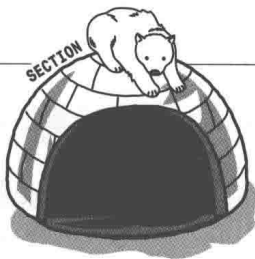
2016 年 4 月于上海

本书中的系统名称、产品名称等一般为各公司的商标或注册商标。正文中一概省略 TM、® 等标识。

©2009 Minerou Aoki

本书包括程序在内的所有内容均受著作权法的保护，未经作者和出版社许可，不得擅自挪用本书内容。

试读结束 需要全本请在线购买：www.ertongbook.com



前 言

本书有两大特征：第一，实际动手实现了真正的编译器；第二，涉及了以往编译器相关书籍所不曾涉及的内容。

先说第一点。

本书通篇讲述了“Cb”这种语言的编译器的制作。Cb基本上是C语言的子集，并实现了包括指针运算等在内的C语言的主要部分。因此可以说，本书实现的是实实在在的编译器，而并非有诸多限制的玩具。

更具体地说，本书实现的Cb编译器是以运行在x86系列CPU上的Linux为平台的。之所以选择x86系列的CPU，是因为这是最普及的CPU，相应的硬件非常容易找到。选择Linux是因为从标准库到程序运行环境的代码都是公开的，只要你有心，完全可以自己分析程序的结构。

可能有些作者不喜欢把话题局限于特定的语言或者OS，而笔者却恰恰更倾向于在一开始就对环境进行限定。因为比起一般化的说明，从具体的环境出发，再向一般化扩展的做法要简单、直观得多。笔者赞成最终把话题往一般化的方向扩展，但并不赞成一开始就一定要做到一般化。

再说第二点。

本书并不局限于书名中的“编译器”，对以编译器为中心的编程语言的运行环境，即编译器、汇编器、链接器、硬件、运行时环境都有所涉及。

编译器生成的程序的运行不仅和编译器相关，和汇编器、链接器等软件以及硬件都密切相关。因此，如果了解编译器以及程序的运行结果，对上述几部分内容的了解当然是必不可少的。不过这里的“当然”现在看起来也逐渐变得没那么绝对了。

只讲编译器或者只讲汇编语言的书已经多得烂大街了，只讲链接器的书也有一些，但是贯穿上述所有内容的书至今还没有。写编译器的书，一涉及具体的汇编语言，就会注上“请参考其他书籍”；写汇编语言的书，对于OS的运行环境问题却只字不提；写链接器的书，如果读者不了解编译器等相关知识，也就只能被束之高阁了。

难道就不可能完整地记述编程语言的运行环境吗？笔者认为可能的。只要专注于具体的语言、具体的OS以及具体的硬件，就可以对程序运行的所有环节进行说明了。基于这样的想法，笔者进行了稍显鲁莽的尝试，并最终写成了本书。

以上就是本书的基本原则。下面是本书的读者对象。

- 想了解编译器和解释器内部结构的人
- 想了解 C 语言程序运行机制的人
- 想了解 x86 CPU (Pentium 或 Intel Core、Operon 等) 的结构的人
- 想了解 Linux 上的链接、加载和程序库的人
- 想学习语法分析的人
- 想设计新的编程语言的人

综上，本书是一本基于具体的编程语言、具体的硬件平台以及具体的 OS 环境，介绍程序运行的所有环节的书。因此，从单纯对编译器感兴趣的读者到以实用为目的的读者，都适合阅读本书。

必要的知识

本书的读者需要具备以下知识。

- Java 语言的基础知识
- C 语言的基础知识
- Linux 的基础知识

本书中制作的 Cb 编译器是用 Java 来实现的，所以能读懂 Java 代码是阅读本书的前提条件。不只是语言，书中对集合等基本库也都没有任何说明，因此需要读者具备相关的知识储备。

本书所使用的 Java 版本是 5.0。关于泛化 (generics) 和 foreach 语句等 Java 5 特有的功能，在第一次出现时会进行简单的说明。

另外，之所以需要读者具有 C 语言的基础知识，是因为 Cb 语言是 C 语言的子集，另外，以 C 语言的知识为基础，对汇编器的理解也将变得容易得多。不过读者不需要深究细节，只要能够理解指针和结构体可以组合使用这种程度就足够了。

最后，关于 shell 的使用方法以及 Linux 方面的常识，这里也不作介绍。例如 cd、ls、cp、mv 等基本命令的用法，都不会进行说明。

不必要的知识

本书的读者不需要具备以下知识。

- 编译器和解释器的构造
- 解析器生成器的使用方法
- 操作系统的详细知识
- 汇编语言
- 硬件知识

即使读者对编译器和解释器的构造一无所知，也没有关系，本书会对此进行详尽的说明。

另外，OS 及 CPU 相关的前提知识也基本不需要。能用 Linux 的 shell 进行文件操作，用 gcc 命令编译 C 语言的“Hello,World”程序，这样就足够了。

本书的结构

本书由以下章节构成。

章		内容
第 1 章	开始制作编译器	本书概要以及了解编译器所需要的基础知识
第 2 章	Cb 和 cbc	本书制作的 Cb 编译器的概要
第 1 部分	代码分析	
第 3 章	语法分析的概要	语法分析的概念及方法
第 4 章	词法分析	cbc 的词法分析（扫描）
第 5 章	基于 JavaCC 的解析器的描述	JavaCC 的使用方法（语法部分）
第 6 章	语法分析	cbc 的语法分析
第 2 部分	抽象语法树和中间代码	
第 7 章	JavaCC 的 action 和抽象语法树	JavaCC 的使用方法（action 部分）
第 8 章	抽象语法树的生成	根据语法分析的结果生成语法树的方法
第 9 章	语义分析（1）引用的消解	变量的引用和具体定义之间的消解
第 10 章	语义分析（2）静态类型检查	编译时的类型检查
第 11 章	中间代码的转换	从抽象语法树生成中间代码
第 3 部分	汇编代码	
第 12 章	x86 架构的概要	使用 Intel 系列 CPU 的系统的构造
第 13 章	x86 汇编器编程	x86 CPU 的汇编语言的读法
第 14 章	函数和变量	x86 CPU 架构中函数调用的形式
第 15 章	编译表达式和语句	和栈帧无关的汇编代码的生成
第 16 章	分配栈帧	和栈帧相关的汇编代码的生成
第 17 章	优化的方法	优化程序的方法的概要
第 4 部分	链接和加载	
第 18 章	生成目标文件	ELF 文件的构造和生成
第 19 章	链接和库	链接的种类和库
第 20 章	加载程序	内存中程序的加载及动态链接
第 21 章	生成地址无关代码	地址无关代码及共享库的生成
第 22 章	扩展阅读	为读者的后续学习介绍相关知识

编译器自身也是一款程序，它将程序的代码逐次进行转换，最终生成可以运行的文件。因此前面章节的内容会成为后续章节的前提，推荐从头开始依次阅读本书的所有章节。

但是，如果你对编译器有一定程度的了解，并且只对特定的话题感兴趣，也可以选取相应的章节来阅读。本书做成的 Cb 编译器可以显示每个阶段生成的数据结构，因此你也可以实际运行一下 Cb 编译器，一边确认前一阶段生成的结果，一边往下阅读。

例如，即使跳过语法分析的章节，只要用 `--dump-ast` 选项显示前一阶段生成的抽象语法

树，就可以理解下一阶段的语义分析和中间代码的相关内容。同样，还可以用 `--dump-ir` 选项显示中间代码，用 `--dump-sam` 选项显示汇编代码。

致谢

首先感谢 RHG 读书会的成员阅读了第 2 部分之前的草稿，并提出了很多宝贵意见。感谢笹田、山下、酒井、向井、shelarcy、志村、岸本、丰福、佐野。

还要感谢 3 年来一直耐心地等待笔者交稿的 SB Creative 株式会社的杉山，以及在短时间内对本书 600 余页的稿件进行编辑的 Top Studio Corporation 株式会社的武藤。非常感谢！

最后，感谢为本书出版付出努力的各位，以及所有维护 Linux 和 GNU 工具等自由软件的人。正是因为有了你们，本书才得以出版。

青木峰郎

目 录

第 1 章

开始制作编译器

1

1.1	本书的概要	2
	本书的主题	2
	本书制作的编译器	2
	编译示例	2
	可执行文件	3
	编译	4
	程序运行环境	6
1.2	编译过程	8
	编译的 4 个阶段	8
	语法分析	8
	语义分析	9
	生成中间代码	9
	代码生成	10
	优化	10
	总结	10
1.3	使用 Cb 编译器进行编译	11
	Cb 编译器的必要环境	11
	安装 Cb 编译器	11
	Cb 的 Hello, World!	12

第 2 章

Cb 和 cbc

13

2.1	Cb 语言的概要	14
	Cb 的 Hello, World!	14
	Cb 中删减的功能	14
	import 关键字	15
	导入文件的规范	16
2.2	Cb 编译器 cbc 的构成	17
	cbc 的代码树	17

cbc 的包	18
compiler 包中的类群	18
main 函数的实现	19
commandMain 函数的实现	19
Java5 泛型	20
build 函数的实现	20
Java 5 的 foreach 语句	21
compile 函数的实现	21

第 1 部分 代码分析

第 3 章

语法分析的概要

24

3.1 语法分析的方法	25
代码分析中的问题点	25
代码分析的一般规律	25
词法分析、语法分析、语义分析	25
扫描器的动作	26
单词的种类和语义值	27
token	28
抽象语法树和节点	29
3.2 解析器生成器	30
什么是解析器生成器	30
解析器生成器的种类	30
解析器生成器的选择	31
3.3 JavaCC 的概要	33
什么是 JavaCC	33
语法描述文件	33
语法描述文件的例子	34
运行 JavaCC	35
启动 JavaCC 所生成的解析器	36
中文的处理	37

第 4 章

词法分析

39

4.1 基于 JavaCC 的扫描器的描述	40
-----------------------------	----

本章的目的	40
JavaCC 的正则表达式	40
固定字符串	41
连接	41
字符组	41
排除型字符组	41
重复 1 次或多次	42
重复 0 次或多次	42
重复 n 次到 m 次	42
正好重复 n 次	43
可以省略	43
选择	43
4.2 扫描没有结构的单词	44
TOKEN 命令	44
扫描标识符和保留字	44
选择匹配规则	45
扫描数值	46
4.3 扫描不生成 token 的单词	48
SKIP 命令和 SPECIAL_TOKEN 命令	48
跳过空白符	48
跳过行注释	49
4.4 扫描具有结构的单词	50
最长匹配原则和它的问题	50
基于状态迁移的扫描	50
MORE 命令	51
跳过块注释	52
扫描字符串字面量	53
扫描字符字面量	53

第 5 章

基于 JavaCC 的解析器的描述

55

5.1 基于 EBNF 语法的描述	56
本章的目的	56
基于 JavaCC 的语法描述	56
终端符和非终端符	57
JavaCC 的 EBNF 表示法	58
连接	58
重复 0 次或多次	59
重复 1 次或多次	59

选择	60
可以省略	60
5.2 语法的二义性和 token 的超前扫描	61
语法的二义性	61
JavaCC 的局限性	62
提取左侧共通部分	63
token 的超前扫描	63
可以省略的规则和冲突	64
重复和冲突	65
更灵活的超前扫描	66
超前扫描的相关注意事项	66

第 6 章

语法分析

68

6.1 定义的分析	69
表示程序整体的符号	69
语法的单位	69
import 声明的语法	70
各类定义的语法	71
变量定义的语法	72
函数定义的语法	73
结构体定义和联合体定义的语法	74
结构体成员和联合体成员的语法	75
typedef 语句的语法	76
类型的语法	76
C 语言和 Cb 在变量定义上的区别	77
基本类型的语法	77
6.2 语句的分析	79
语句的语法	79
if 语句的语法	80
省略 if 语句和大括号	80
while 语句的语法	81
for 语句的语法	81
各类跳转语句的语法	82
6.3 表达式的分析	83
表达式的整体结构	83
expr 的规则	83
条件表达式	84
二元运算符	85

6.4	项的分析	88
	项的规则	88
	前置运算符的规则	88
	后置运算符的规则	89
	字面量的规则	89

第 2 部分 抽象语法树和中间代码

第 7 章

JavaCC 的 action 和抽象语法树

92

7.1	JavaCC 的 action	93
	本章的目的	93
	简单的 action	93
	执行 action 的时间点	93
	返回语义值的 action	95
	获取终端符号的语义值	95
	Token 类的属性	96
	获取非终端符号的语义值	98
	语法树的结构	99
	选择和 action	99
	重复和 action	100
	本节总结	102
7.2	抽象语法树和节点	103
	Node 类群	103
	Node 类的定义	105
	抽象语法树的表示	105
	基于节点表示表达式的例子	107

第 8 章

抽象语法树的生成

110

8.1	表达式的抽象语法树	111
	字面量的抽象语法树	111
	类型的表示	112
	为什么需要 TypeRef 类	113
	一元运算的抽象语法树	114
	二元运算的抽象语法树	116

条件表达式的抽象语法树	117
赋值表达式的抽象语法树	118
8.2 语句的抽象语法树	121
if 语句的抽象语法树	121
while 语句的抽象语法树	122
程序块的抽象语法树	123
8.3 声明的抽象语法树	125
变量声明列表的抽象语法树	125
函数定义的抽象语法树	126
表示声明列表的抽象语法树	127
表示程序整体的抽象语法树	128
外部符号的 import	128
总结	129
8.4 cbc 的解析器的启动	132
Parser 对象的生成	132
文件的解析	133
解析器的启动	134

第 9 章

语义分析 (1) 引用的消解

135

9.1 语义分析的概要	136
本章目的	136
抽象语法树的遍历	137
不使用 Visitor 模式的抽象语法树的处理	137
基于 Visitor 模式的抽象语法树的处理	138
Vistor 模式的一般化	140
cbc 中 Visitor 模式的实现	141
语义分析相关的 cbc 的类	142
9.2 变量引用的消解	144
问题概要	144
实现的概要	144
Scope 树的结构	145
LocalResolver 类的属性	146
LocalResolver 类的启动	146
变量定义的添加	147
函数定义的处理	148
pushScope 方法	149
currentScope 方法	149

popScope 方法	150
添加临时作用域	150
建立 VariableNode 和变量定义的关联	151
从作用域树取得变量定义	151
9.3 类型名称的消解	153
问题概要	153
实现的概要	153
TypeResolver 类的属性	153
TypeResolver 类的启动	154
类型的声明	154
类型和抽象语法树的遍历	155
变量定义的类型消解	156
函数定义的类型消解	157

第 10 章

语义分析 (2) 静态类型检查

159

10.1 类型定义的检查	160
问题概要	160
实现的概要	161
检测有向图中的闭环的算法	162
结构体、联合体的循环定义检查	163
10.2 表达式的有效性检查	165
问题概要	165
实现的概要	165
DereferenceChecker 类的启动	166
SemanticError 异常的捕获	167
非指针类型取值操作的检查	167
获取非左值表达式地址的检查	168
隐式的指针生成	169
10.3 静态类型检查	170
问题概要	170
实现的概要	170
Cb 中操作数的类型	171
隐式类型转换	172
TyperChecker 类的启动	173
二元运算符的类型检查	174
隐式类型转换的实现	175

第 11 章**中间代码的转换****178**

11.1	cbc 的中间代码	179
	组成中间代码的类	180
	中间代码节点类的属性	181
	中间代码的运算符和类型	182
	各类中间代码	183
	中间代码的意义	184
11.2	IRGenerator 类的概要	185
	抽象语法树的遍历和返回值	185
	IRGenerator 类的启动	185
	函数本体的转换	186
	作为语句的表达式的判别	187
11.3	流程控制语句的转换	189
	if 语句的转换 (1) 概要	189
	if 语句的转换 (2) 没有 else 部分的情况	190
	if 语句的转换 (3) 存在 else 部分的情况	191
	while 语句的转换	191
	break 语句的转换 (1) 问题的定义	192
	break 语句的转换 (2) 实现的方针	193
	break 语句的转换 (3) 实现	194
11.4	没有副作用的表达式的转换	196
	UnaryOpNode 对象的转换	196
	BinaryOpNode 对象的转换	197
	指针加减运算的转换	198
11.5	左值的转换	200
	左边和右边	200
	左值和右值	200
	cbc 中左值的表现	201
	结构体成员的偏移	202
	成员引用 (expr.memb) 的转换	203
	左值转换的例外: 数组和函数	204
	成员引用的表达式 (ptr->memb) 的转换	205
11.6	存在副作用的表达式的转换	206
	表达式的副作用	206
	有副作用的表达式的转换方针	206
	简单赋值表达式的转换 (1) 语句	207
	临时变量的引入	208