手把手教你构建 C 语言编译器（9）- 总结

Table of Contents

恭喜你完成了自己的 C 语言编译器，本章中我们发一发牢骚，说一说编写编译器值得注意的一些问题；编写编译器时遇到的一些难题。

手把手教你构建 C 语言编译器系列共有10个部分：

1. 手把手教你构建 C 语言编译器（0）——前言
2. 手把手教你构建 C 语言编译器（1）——设计
3. 手把手教你构建 C 语言编译器（2）——虚拟机
4. 手把手教你构建 C 语言编译器（3）——词法分析器
5. 手把手教你构建 C 语言编译器（4）——递归下降
6. 手把手教你构建 C 语言编译器（5）——变量定义
7. 手把手教你构建 C 语言编译器（6）——函数定义
8. 手把手教你构建 C 语言编译器（7）——语句
9. 手把手教你构建 C 语言编译器（8）——表达式
10. 手把手教你构建 C 语言编译器（9）——总结

虚拟机与目标代码

整个系列的一开始，我们就着手虚拟机的实现。不知道你是否有同感，这部分对于整个编译器的编写其实是十分重要的。我认为至少占了重要程度的50%。

这里要说明这样一个观点，学习编译原理时常常着眼于词法分析和语法分析，而忽略了同样重要的代码生成。对于学习或考试而言或许可以，但实际编译项目时，最为重要的是能“跑起来”，所以我们需要给予代码生成高度的重视。

同时我们也看到，在后期解析语句和表达式时，难点已经不再是语法分析了，而是如何为运算符生成相应的汇编代码。

词法分析

我们用了很暴力的手段编写了我们的词法分析器，我认为这并无不可。

但你依旧可以学习相关的知识，了解自动生成词法分析器的原理，它涉及到了“正则表达式”，“状态机”等等知识。相信这部分的知识能够很大程度上提高你的编程水平。

同时，如果今后你仍然想编写编译器，不妨试试这些自动生成工具。

语法分析

长期以来，语法分析对我而言一直是迷一样的存在，直到真正用递归下降的方式实现了一个。

我们用了专门的一章讲解了“递归下降”与 BNF 文法的关系。希望能减少你对理论的厌恶。至少，实现起来并不是太难。

如果有兴趣，可以学习学习这些文法，因为已经有许多自动生成的工具支持它们。这样你就不需要重复造轮子。可以看看 yacc 等工具，更先进的版本是 `bsion`。同时其它语言也有许多类似的支持。

题外话，最近知道了一个叫“PEG 文法”的表示方法，无论是读起来，还是实现起来，都比 BNF 要容易，你也可以学习看看。

关于编代码

这也是我自己的感慨吧。无论多好的教程，想要完全理解它，最好的方式恐怕还是要自己实现它。

只是在编写代码的过程中，我们会遇到许多的挫折，例如需要考虑许多细节，或是调试起来十分困难。但也只有真正静下心来去克服它，我们才能有所成长吧。

例如在编写表达式的解析时，大量重复的代码特别让人崩溃。还有就是调试编译器，简直痛苦地无话可说。

P.S. 如果你按这个系列自己编写代码，记得事先写一些用于输出汇编代码的函数，很有帮助的。

还有就是写这个系列的文章，开始的冲动过了之后，每写一篇都特别心烦，希望文章本身没有受我的这种情绪影响吧。

结语

编程有趣又无趣，只有身在其中的我们才能体会吧。