# 跟vczh看实例学编译原理——一：Tinymoe的设计哲学

自从《序》胡扯了快一个月之后，终于迎来了正片。之所以系列文章叫《看实例学编译原理》，是因为整个系列会通过带大家一步一步实现Tinymoe的过程，来介绍编译原理的一些知识点。

但是第一个系列还没到开始处理Tinymoe源代码的时候，首先的跟大家讲一讲我设计Tinymoe的故事。为什么这种东西要等到现在才讲呢，因为之前没有文档，将了也是白讲啊。Tinymoe在github的wiki分为两部分，一部分是介绍语法的，另一部分是介绍一个最小的标准库是如何实现出来的，地址在 <https://github.com/vczh/tinymoe/wiki> 不带问号的那些都是写完了的。

## 系列文章的目标

在介绍Tinymoe之前，先说一下这个系列文章的目标。Ideally，只要一个人看完了这个系列，他就可以在下面这些地方得到**入门**：

1. 词法分析
2. 语法分析
3. 语义分析
4. 符号表
5. 全文CPS变换
6. 编译生成高效的其他语言的代码
7. 编译生成自己的指令集
8. 带GC的虚拟机
9. 类型推导（intersection type，union type，concept mapping）
10. 跨过程分析（inter-procedural analysis）

当然，这并不能让你成为一个大牛，但是至少自己做做实验搞一点高大上的东西骗师妹们是没有问题了。

## Tinymoe设计的目标

虽然想法很多年前就已经有了，但是这次我想把它实现出来，是为了完成《如何设计一门语言》的后续。光讲大道理是没有意义的，至少得有一个例子，让大家知道这些事情到底是什么样子的。因此Tinymoe有一点教学的意义，不管是使用它还是实现它。

首先，**处理Tinymoe需要的知识点多，用于编译原理教学**。既然是为了展示编译原理的基础知识，因此语言本身不可能是那种烂大街的C系列的东西。当然除了知识点以外，还会让大家深刻的理解到，**难实现跟难用，是完全没有关系的**！Tinymoe用起来可爽了，啊哈哈哈哈哈。

其次，**Tinymoe容易嵌入其他语言的程序，作为DSL使用，可以调用宿主程序提供的功能**。这严格的来讲不算语言本身的功能，而是实现本身的功能。就算是C++也可以设计为嵌入式，lua也可以被设计为编译成exe的。一个语言本身的设计并不会对如何使用它有多大的限制。为了让大家看了这个系列之后，写出来的东西不总是玩具，因此这也是设计的目标之一。

再次，**Tinymoe语法优化于描述复杂的逻辑**，而不是优化与复杂的数据结构和算法（虽然也可以）。Tinymoe本身是不存在任何细粒度控制内存的能力的，而且虽然可以实现复杂的数据结构和算法，但是本身描述这些东西最多也就跟javascript一样容易——其实就是不容易。但是Tinymoe设计的时候，是为了让大家把Tinymoe当成是一门可以设计DSL的语言，因此对复杂逻辑的描述能力特别强。唯一的前提就是，你懂得如何给Tinymoe写库。很好的使用和很好地实现一个东西是相辅相成的。我在设计Tinymoe之初，很多pattern我也不知道，只是因为设计Tinymoe遵循了科学的方法，因此最后我发现Tinymoe竟然具有如此强大的描述能力。当然对于读者们本身，也会在阅读系列文章的有类似的感觉。

最后，**Tinymoe是一个动态类型语言**。这纯粹是我的个人爱好了。对一门动态类型语言做静态分析那该多有趣啊，啊哈哈哈哈哈哈。

## Tinymoe的设计哲学

小规模的语言核心+大规模的标准库

扩展后的东西跟原生的东西外观一致

## Tinymoe的实现难点

需要三次Parsing操作，动态生成文法，但是仍然按照手写递归下降的方法来写

CPS变换后没有特殊处理直接执行容易导致callstack积累的没用的东西过多

动态类型语言的类型推导

跨过程分析和对程序的控制流的化简（譬如continuation转状态机等）