编辑器与IDE

编辑器与IDE

无谓的编辑器战争

很多人都喜欢争论哪个编辑器是最好的。其中最大的争论莫过于 Emacs 与 vi 之争。vi 的支持者喜欢说：“看 vi 打起字来多快，手指完全不离键盘，连方向键都可以不用。”Emacs 的支持者往往对此不屑一顾，说：“打字再快又有什么用。我在 Emacs 里面按一个键，等于你在 vi 里面按几十个键。”

其实还有另外一帮人，这些人喜欢说：“对于 Emacs 与 vi 之争，我的答案是 {jEdit， Geany, TextMate, Sublime…}”这些人厌倦了 Emacs 的无休止的配置和 bug，也厌倦了 vi 的盲目求快和麻烦的模式切换，所以他们选择了另外的更加简单的解决方案。

临时解决方案 - IDE

那么我对此的答案是什么呢？在目前的情况下，我对程序编辑的临时答案是：IDE。

写程序的时候，我通常根据语言来选择最能“理解”那种语言的“IDE”（比如 Visual Studio, Eclipse, IntelliJ IDEA 等），而不是一种通用的“文本编辑器”（比如 Emacs, vi, jEdit, …）。这是因为“文本编辑器”这种东西一般都不真正的理解程序语言。很多 Emacs 和 vi 的用户以为用 etags 和 ctags 这样的工具就能让他们“跳转到定义”，然而这些 tags 工具其实只是对程序的“文本”做一些愚蠢的正则表达式匹配。它们根本没有对程序进行 parse，所以其实只是在进行一些“瞎猜”。简单的函数定义它们也许能猜对位置，但是对于有重名的定义，或者局部变量的时候，它们就力不从心了。

很多人对 IDE 有偏见，因为他们认为这些工具让编程变得“傻瓜化”了，他们觉得写程序就是应该“困难”，所以他们眼看着免费的 IDE 也不试一下。有些人写 Java 都用 Emacs 或者 vi，而不是 Eclipse 或者 IntelliJ。可是这些人错了。他们没有意识到 IDE 里面其实蕴含了比普通文本编辑器高级很多的技术。这些 IDE 会对程序文本进行真正的 parse，之后才开始分析里面的结构。它们的“跳转到定义”一般都是很精确的跳转，而不是像文本编辑器那样瞎猜。

这种针对程序语言的操作可以大大提高人们的思维效率，它让程序员的头脑从琐碎的细节里面解脱出来，所以他们能够更加专注于程序本身的语义和算法，这样他们能写出更加优美和可靠的程序。这就是我用 Eclipse 写 Java 程序的时候相对于 Emacs 的感觉。我感觉到自己的“心灵之眼”能够“看见”程序背后所表现的“模型”，而不只是看到程序的文本和细节。所以，我经常发现自己的头脑里面能够同时看到整个程序，而不只是它的一部分。我的代码比很多人的都要短很多也很有很大部分是这个原因，因为我使用的工具可以让我在相同的时间之内，对代码进行比别人多很多次的结构转换，所以我往往能够把程序变成其他人想象不到的样子。

对于 Lisp 和 Scheme，Emacs 可以算是一个 IDE。Emacs 对于 elisp 当然是最友好的了，它的 Slime 模式用来编辑 Common Lisp 也相当不错。然而对于任何其它语言，Emacs 基本上都是门外汉。我大部分时间在 Emacs 里面是在写一些超级短小的 Scheme 代码，我有自己的一个简单的配置方案。虽然谈不上是 IDE，Emacs 编辑 Scheme 确实比其它编辑器方便。R. Kent Dybvig 写 Chez Scheme 居然用的是 vi，但是我并不觉得他的编程效率比我高。我的代码很多时候比他的还要干净利落，一部分原因就是因为我使用的 ParEdit mode 能让我非常高效的转换代码的“形状”。

当要写 Java 的时候，我一般都用 Eclipse。最近写 C++ 比较多，C++ 的最好的 IDE 当然是 Visual Studio。可惜的是 VS 没有 Linux 的版本，所以就拿 Eclipse 凑合用着，感觉还比较顺手。个别情况 Eclipse “跳转定义”到一些完全不相关的地方，对于 C++ 的 refactor 实现也很差，除了最简单的一些情况（比如局部变量重命名），其它时候几乎完全不可用。当然 Eclipse 遇到的这些困难，其实都来自于 C++ 语言本身的糟糕设计。

终极解决方案 - 结构化编辑器

想要设计一个 IDE，可以支持所有的程序语言，这貌似一个不大可能的事情，但是其实没有那么难。有一种叫做“结构化编辑器”的东西，我觉得它可能就是未来编程的终极解决方案。

跟普通的 IDE 不同，这种编辑器可以让你直接编辑程序的 AST 结构，而不是停留于文本。每一个界面上的“操作”，对应的是一个对 AST 结构的转换，而不是对文本字符的“编辑”。这种 AST 的变化，随之引起屏幕上显示的变化，就像是变化后的 AST 被“pretty print”出来一样。这些编辑器能够直接把程序语言保存为结构化的数据（比如 S表达式，XML 或者 JSON），到时候直接通过对 S表达式，XML 或者 JSON 的简单的“解码”，而不需要针对不同的程序语言进行不同的 parse。这样的编辑器，可以很容易的扩展到任何语言，并且提供很多人都想象不到的强大功能。这对于编程工具来

说将是一个革命性的变化。

- 已经有人设计了这样一种编辑器的模型，并且设计的相当不错。你可以参考一下这个结构化编辑器，它包含一些 Visual Studio 和 Eclipse 都没有的强大功能，却比它们两者都要更加容易实现。你可以在这个网页上下载这个编辑器模型来试用一下。
- 我之前推荐过的 TeXmacs 其实在本质上就是一个“超豪华”的结构化编辑器。你可能不知道，TeXmacs 不但能排版出 TeX 的效果，而且能够运行 Scheme 代码。
- IntelliJ IDEA 的制造者 JetBrains 做了一个结构化编辑系统，叫做 MPS。它是开源软件，并且可以免费下载。
- 另外，Microsoft Word 的创造者 Charles Simonyi 开了一家叫做 Intentional Software 的公司，也做类似的软件。