学习报告

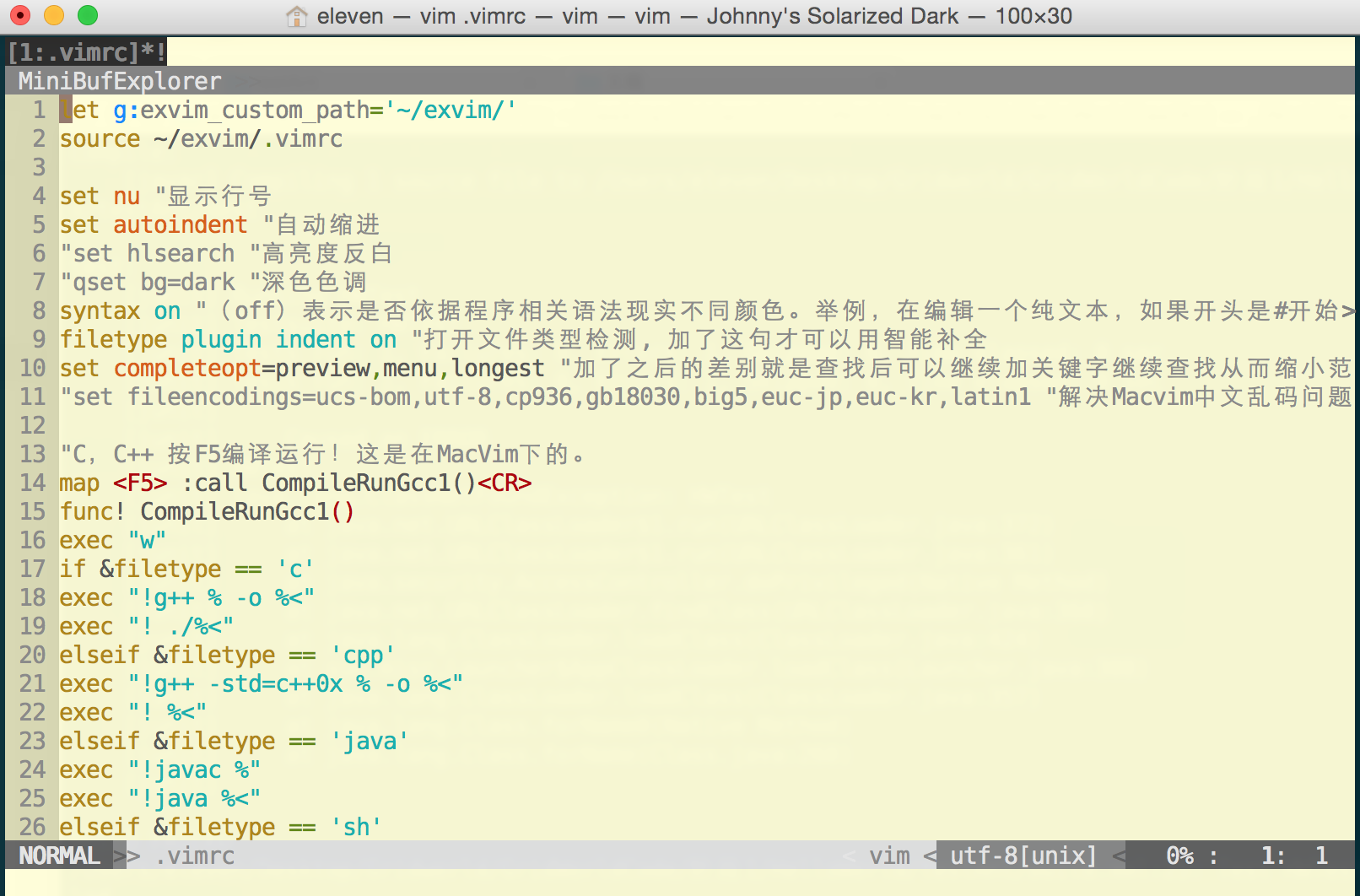
陈炜健 — 13331018

2015 年 8 月 19 日

# VIM 编辑器

不用多说，Vim是一个十分强大的编辑器工具。幸好之前大一的时候已经学了不少Vim的基本知识，所以实训开始要学习并运用Vim这一个任务对于我来说难度并不大。

首先，我在终端中打开Vimtutor回顾了一些基本的功能，譬如说：移动光标、进入及退出Vim、删除文本、插入文本、置换文本、搜索文本、执行外部命令等等简单的命令，整个流程下来，不到20分钟，但已经可以掌握到Vim的使用方法了。由于之前已经折腾过Vim的环境配置，所以在自己电脑中打开Vim的时候已经默认具备显示行号、自动缩进、检测文件类型自动插入文件头、依据程序相关语法显示不同颜色、修改部分功能键的映射等等的功能。



（vimrc文件部分代码）

虽然Vim的编辑功能十分强大，但毕竟Vim只是一个编辑器，倘若处理一个目录结构复杂的项目，还是会显得不够方便。其实在很多IDE中都会有官方或第三方Vim插件，使IDE的编辑环境趋同于Vim环境，但我觉得还是有不少不满意的地方，譬如说会有功能键冲突的问题，抑或稳定性不强易崩溃。所以，综合考虑下，平时在自己电脑中，我都会选择exVim，即解决了使用Vim编辑的需求，又满足了拥有IDE基本项目管理功能的要求。

在[官网](http://exvim.github.io/docs-zh/intro/)中，exVim的介绍如下：

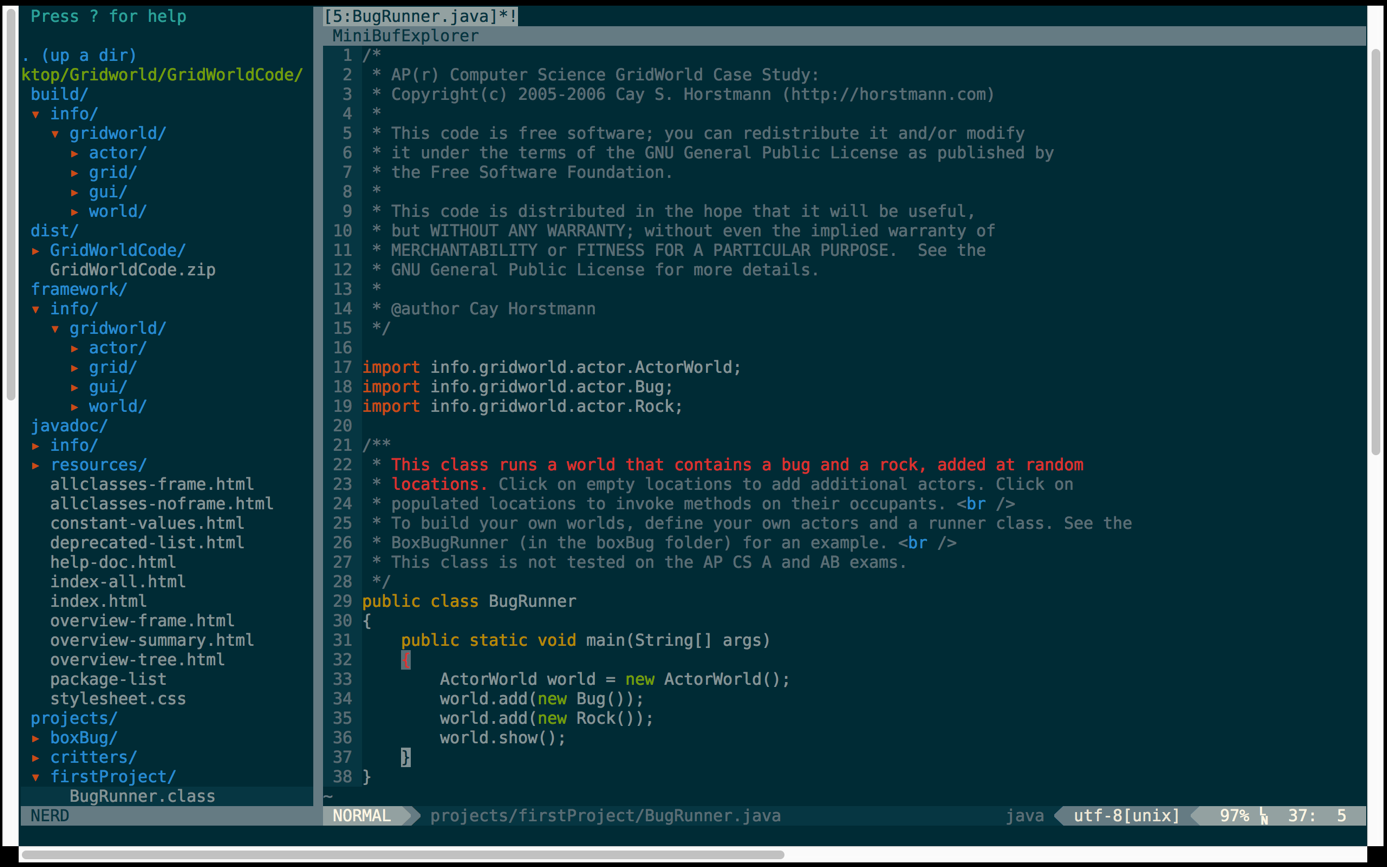
*“ exVim 是一个力求将 Vim 改良成集成编辑环境(IDE)的项目，exVim 在 Vim 中加入项目工程文件(.exvim)的概念, 通过使用 Vim 编辑项目工程文件 (.exvim) 触发 exVim 的插件, 从而让你能够在不同的工程中使用不同的 Vim 配置，插件配置, 甚至按需加载不同的插件. 总的来说, exVim 就是为了使 Vim 成为世界上最好用的 IDE。”*

我觉得，支持插件是Vim之所以强大的原因之一，因为它能丰富Vim的编辑功能及提高编辑效率，但如果安装插件过多，就会显得Vim比较臃肿，而且管理起来十分麻烦，而exVim就支持统一的插件管理，而且默认已经安装了ex-tags和ex-symbol插件，能够全局分析代码，这对于项目的管理极其方便。

好吧，再继续说exVim下去好像有点偏题了，总之exVim对于Vim使用者来说真的是一个挺好的应用。

其实Vim的学习曲线是挺陡峭的，一开始很容易入手，但到了后期每前进一小步都需要付出不少时间和精力，但只要能前进一步，就能学习到很多的知识。

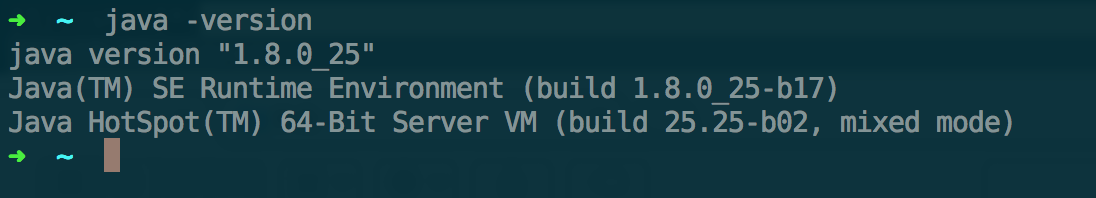
最后趁有空位附上exVim界面，图中目录为GridWorldCode目录。

  
（exVim 界面）

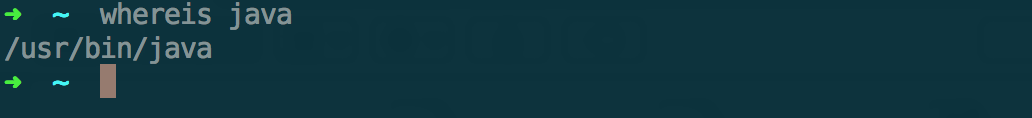
# JAVA

首先，说一下环境的配置。由于自己使用OS X系统的，为了方便，我没有选择手动配置JAVA环境，而是选择在官网下载dmg文件，即安装包的形式安装JDK。但我很好奇JDK安装在哪个目录下，毕竟等下配置Ant和JUnit时需要修改环境变量，会涉及到JDK目录。

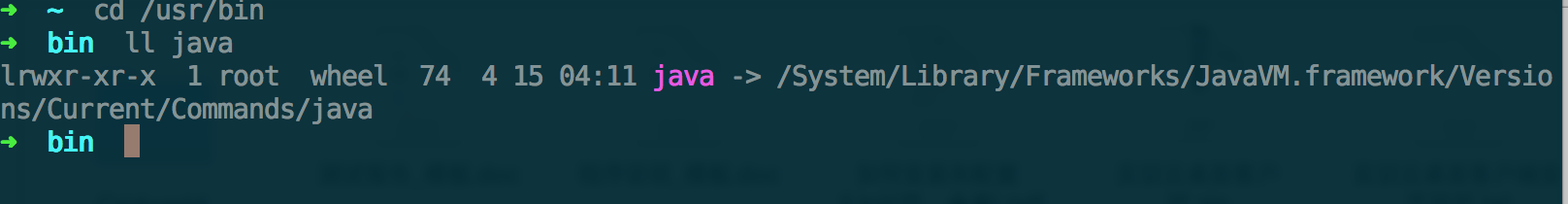
1. 首先，我要先检查是否成功安装了JAVA。



1. 安装成功，接着，用whereis命令查看JAVA文件目录。



1. 进入 /usr/bin 目录，查看java文件夹的信息。



1. 发现该文件夹存在映射，实际文件在——［/System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands/］
2. Commands目录下存在java可执行文件，但其实该文件只是调用了真正目录下的java文件——［/Library/Java/JavaVirtualMachines/jdk1.8.0\_25.jdk/Contents/Home/bin/java］

挺神奇的，兜兜转转才找到真正的JAVA文件目录。（参考自：[Macintosh](http://blog.csdn.net/davidhsing/article/details/8297105)）

至于JAVA语法方面，由于之前参加比赛的缘故，也稍微地提早接触了JAVA语言，所以一些基本的语法自己还是掌握了的，记得一开始学的时候，我惊讶地发现Java很多地方都和C++语法是一样的，并且能体现面相对象的思想（C++不一定是面向对象，Java纯粹面向对象）。但用Java编码一段时间后，还是发现有不少区别的，譬如说，Java由于安全的考虑，不再支持在C++中灵活性很强的指针操作；由于要体现完全的面相对象编程，Java不能在所有类之外定义全局变量，只能在在某个类中定义一种公用静态的变量来实现全局变量的功能.；出于提高可读性的目的，不再支持头文件；为了便于维护代码，也不支持头文件……

其实深究一下，还是可以发现不少区别的。尽管之前已经写了一些Java代码，但我觉得还是有不少知识要学的，幸好，自己可以通过这一次实训加深对Java这一门编程语言的了解。

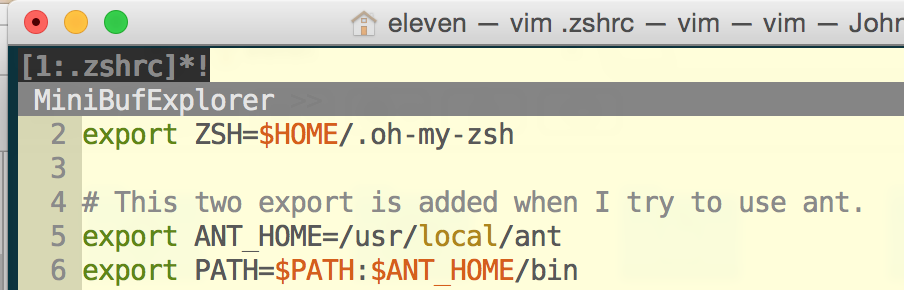
第一天实训，题目便就要求我们写一个Java计算器程序，计算器看似简单，逻辑性不太复杂，但其实难点在于GUI界面，一开始我看着这计算器示例图片，觉得有点不可思议，不太明白这界面究竟在Java中该如何实现。其实我一开始接触Java，是在Qt环境下编写程序的，而Qt会提供GUI的库，调用其中的函数，可以实现软件布局的设计。所以我想，应该是要调用内置的库吧。果然，Java有一个抽象窗口工具包AWT，它提供了一套与本地图形界面进行交互的接口。通过在网上搜索AWT，我找到了Swing，一个在AWT的基础上构建的一套新的图形界面系统。通过Swing，我又了解到它的布局管理，有FlowLayout、BorderLayout、GridLayout、GridBagLayout这四种布局。我惊喜地发现，其实示例图中所展示的布局就是GridLayout。定下大致方向后，计算器也就不难了，通过一个多小时的编码后，雏形也就完成了。后来修改了一下细节，总算符合示例的样式。

# ANT

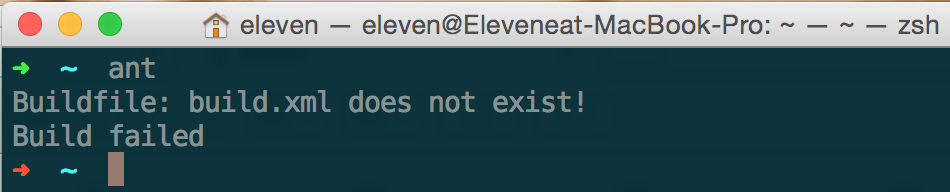
首先，还是要配置一下环境。

忍不住要吐槽，看着别人都能直接使用apt-get命令来安装程序包，自己只能一步一步地根据教程来配置环境，中途遇到的坑实在不少。但换个角度来想，虽然自己耗费了比别人多的时间来配置环境，但过程中我还是学到不少系统内层程序管理以及文件操作原理的知识，这对于自己专业能力的提升肯定会有积极的帮助。

1. 把ant文件放到 /usr/local 目录下，这一步没有任何问题。
2. 设置系统环境变量。教程中是建议修改 /etc 目录下的profile文件，但我只想在当前用户中修改环境变量，并且平时我主要维护的是用户下的 .zshrc 文件（我的shell为zsh），所以我还是在该文件下修改吧。



1. 保存配置文件，退出重新打开shell，检查ant是否配置成功。



1. 成功，ant已可用。

其实ant的功能挺强大的，自动构建和部署Java程序，其实以前很少用自动生成程序，最多就是写一写makefile，所以通过ant的学习，我学到了不少项目管理的知识。

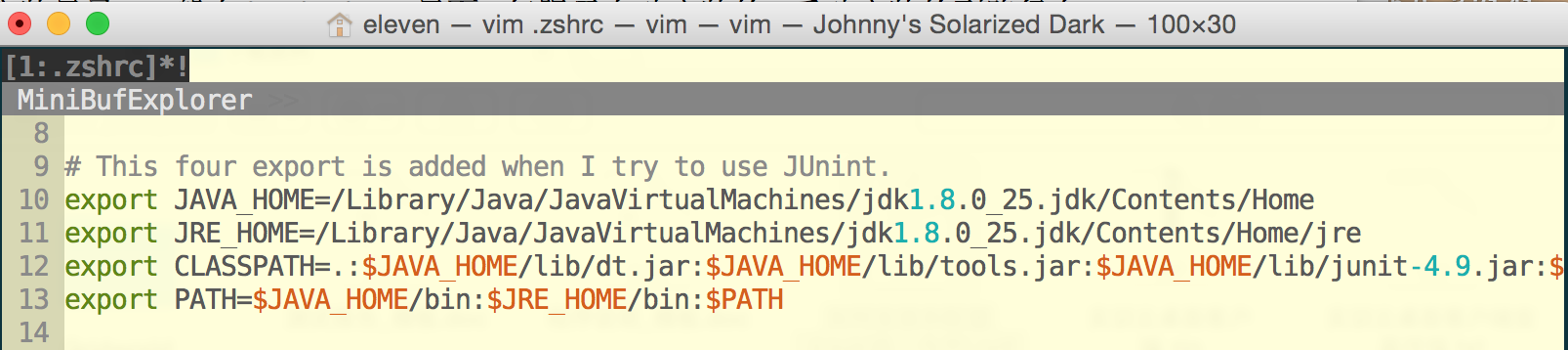
语法也不算特别难，用的是xml格式的标志语言，难点在于弄清楚每个标签的具体功能，以及弄清配合Java所涉及的逻辑。

当然，这次了解的只是ant其中比较简单的功能，课后深究一下还是必须的。

# Junit

JUnit是一个Java语言的单元测试框架。

首先配置环境，和ant环境配置大同小异，下图是 .zshrc 文件中修改的环境变量。



语法也不是特别难，根据教程很快就写出了一个例子了。

然而，第二天实训早上的时间，我一直在弄ant实现JUnit的自动测试，可是一直卡在了一个错误，找不到测试文件的目录，上午百度谷歌了很多教程，终于发现原来是在ant中classpath中一些语法写错了，导致找不到要测试的类的路径。更改过后，就可以利用ant来实现JUnit的自动测试了～

