实训学习报告

梁展瑞
中山大学软件学院　　2010 计应　　学号 10389084

August 28, 2012

Vim 学习报告

由于 Vim 的功能超越了古老而光荣的 Vi, 所以我选择使用 Vim. Vim 是个文本编辑器, 其学习曲线陡峭之大, 让一般人刚入门时觉得无所适从. 但是当使用熟练时, 它便是你爱不释手的编辑利器.

在一天的学习过程中，我学习和熟悉了 Vim 使用. 它在平常的编辑工作中作用巨大，其功能有下:

- 查看和修改配置文件
- 编写代码
- 编写学习报告
- 充当文件浏览器，利用其强大的搜索功能.

Vim 的主要特点是带模式的编辑，这点与另一个著名的编辑器 Emacs（Emacs 是否是编辑器这个存在争议，有人说它时一个操作系统）相反。在正常模式下, 我们用 h,j,k,l 分别进行左, 下, 上, 右进行导航。要输入时, 在按了 i, a, r, R 等健后, 我们便从正常模式 (normal) 切换到输入 (insert) 模式, 进行文本输入。Vim 还有其他几种模式, 比较常用的除输入模式外还有命令输入模式 (: 后跟命令), v 模式 (visual mode). 有了模式之后, 可以大大减少组合键的使用, 减少手指的劳累程度。

使用 Vim 工作可以大大提高工作效率, Vim 的作者把常用键位安排在了手指最容易到达的地方, 使编辑时手指很少离开 home 行, 更不用说鼠标了, Vim 的使用是不需要鼠标的。

另一个强大的地方是 Vim 还有自己的 script, 可以编写灵活的配置文件和执行复杂的编辑要求。例如, 可以编写一个脚本函数, 在按了 <F5> 之后在光标所在的地方插入当前日期, 这仅仅需要一行

```
inoremap <F5> <C-R>=strftime("%c")<CR>
```

再例如, 我只想在编辑 c++ 源代码的时候绑定几个键位:

- press F7 to compile, or make.
- press F6 to compile, with DEBUG flag.
- press F5 to run the compile program.

那么我在配置脚本里加上如下几行就行：

```
au BufNewFile,BufRead *.cpp nmap <F5> :!./a < in<CR>
au BufNewFile,BufRead *.cpp nmap <F6> :!g++ % -o a -g -DDEBUG<CR>
au BufNewFile,BufRead *.cpp nmap <F7> :!g++ % -o a<CR>
```

在熟悉 vim 脚本之后, 还可以组合出更加多实用的功能, 在程序员手上发挥更强的功能。

vim 的使用方式和设计方式已经渗入到许多软件当中,

- Firefox 有个插件 Pentadactyl, 可以用 hjkl 进行网页浏览, 使用 yy,pp 进行复制粘贴, 用 m（mark）进行书签, 带有:(coloum）命令模式另外, Chrome 也有类似的插件。
- Linux 下的平铺式窗口管理器（Tiling Windows Manager, also TWM, but not twm), 一般都使用 hjkl 进行窗口跳转, 大大减少鼠标的使用。
- 有一个 TUI 式的文件管理器（实现 windows explorer 的功能）, Ranger, 使用 hjkl 导航, 用 yy,pp,dd,ox 进行复制, 粘贴, 剪切等, 还带有 v 模式。使用非常方便, 功能强大。用 Python 写成。

认真学习 Vim 对我日后的学习和工作将会有极大帮助。

java 学习报告

Java 是一门面向对象的程序设计语言。外观看上去和 C++ 很像。但实际上 Java 是完全 OOP 风格的，就连 main 入口函数都要包在某个类里面，可见其设计者的洁癖程度。

我基本没有接触过 Java，不过在某次做游戏时学习了几个标准类 (String, Array, StringBuilder, Scanner, OutputStreamWriter, Socket) 的用法, 所以谈谈一个初学者的粗浅见解。

在写了一个 HelloWorld 之后, 发现 Java 的类声明中每个成员（变量或函数）都要声明其访问属性，这样有利于灵活性和代码但维护。另外 Java 的一个类只在一个文件里面，省去了在头文件声明一次，又在实现文件里面写一次函数头但麻烦。

到目前为止，我最喜欢 Java 的一点是，其所有变量都是类实例的引用（为了提高速度，数值类型可能例外做了优化)。而且内存是自动管理回收的，让程序员可以把注意力集中到更关键的地方, 不用像写 C 程序一样时刻担心内存管理问题. 另外 java 是运行在 JVM（Java 虚拟机）上的，有点像动态语言但解析器，这样 Java 但可移植性大大提高。这几点使 Java 在使用上和一些动态语言比较相似（例如 Python)。

下面是我配置 Java 环境的过程：

- 由于我使用的 OS 是 Archlinux, 所以我用 pacman（包管理器）来安装 java 环境。
- 根据 Sun 官方网站的安装指导, 知道需要 JRE 和 JDK 这两样东西
- 用命令:

```
# pacman -Ss jdk
```

  搜索得到 arch 官方源里面有 jdk7-openjdk 和 jre7-openjdk 可以用。

- 用命令

```
# pacman -S jdk7-openjdk jre7-openjdk
```

进行安装。

当前两者的版本都是 7.u5 .

至于环境变量的配置, 安装完重启 session 发现环境变量中已经有 JAVA_Home 了, 于是用`pacman -Ql jdk7-openjdk|grep etc` 命令查看了这个包但配置文件, 发现在 /etc/profile.d/ 里面有配置文件 jdk.sh, jre.sh。已经完成了配置。

```
# configs in jdk.sh
export J2SDKDIR=/usr/lib/jvm/java-7-openjdk
export J2REDIR=$J2SDKDIR/jre
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk

# configs in jre.sh
export J2REDIR=/usr/lib/jvm/java-7-openjdk/jre
export JAVA_HOME=${JAVA_HOME:-/usr/lib/jvm/java-7-openjdk/jre}
```

然后写了一个 HelloWorld.java 来测试, 执行:

```
# vim HelloWorld.java
 ...
# javac HelloWorld.java && java HelloWorld.class
```

成功输出。