Vim/Cscope 入门指导

翻译：手气不错 <i.feelinglucky@gmail.com>
主页：http://www.gracecode.com

英文原文：http://cscope.sourceforge.net/cscope_vim_tutorial.html

整理日期：2007 年 12 月 7 日

Cscope 是一个非常灵巧的工具，但是它仍然需要使用您最顺手的编辑器去发挥它强大的功能。 幸运的是，Vim 已经包含了 Cscope 支持。

这篇指导主要介绍了 Vim 内在的 Cscope 支持，以及如何更好的通过已配置的 Map（a set of maps）实现搜索功能。

下面假设您有基本的使用类 vi 编辑器的技能 - 不过并不需要 Vim 更特殊和高深的功能（熟悉 ViM 的一些高级功能，比如多窗口等需要一定的精力和时间）。您不需要了解有关 Cscope 的任何信息：这就是我们下面要阐述的。

万变不离其宗，如果您使用类似功能的软件，比如 Vim 的 ctags，您会发现 Cscope 与其非常的相似。不过有一点不同的是，Cscope 比 ctags 有更多的搜索类型和选项。

这是具体步骤的指导，所以请您打开一个终端（shell），并且按照下面的步骤：

如果您的系统还未安装 Cscope ，您第一步需要做的就是获取和安装它。理想上如果您使用 Vim 6.x 版本应该无此问题，不过 Vim 5.x 版本仍然可以获得大部分的功能函数（垂直分割似乎不起作用，不过水平分割通过在文件中指定 Maps 可以使用）。

> *注意：如果您的 Vim 在编译时没有加入 '--enable-cscope' 选项，请您加入此选项并重新编译安装。大部分 Linux 发行版中 Vim 中的此编译选项是被禁止的。*

下载 cscope_maps.vim 文件，并在您的 Vim 启动配置文件中加入此文件。如果您使用的是 Vim 6.x 版本，您可以将这个文件安排在 $HOME/.vim/plugin 目录下（如果是其他的目录，请您了解您的 'runtimepath' 环境变量）。 如果您正在使用的是 Vim5.x 的版本，您可以将此文件的所有内容加入您的 $HOME/.vimrc 文件中，或者在 .vimrc 中加入 "source cscope_maps.vim" 命令（注，此文件无比在可识别的路径中）。

进入有 C 代码的目录，键入 'cscope -R' 命令（'-R' 选项指定包含此目录的子目录，而非仅仅是当前目录）。如果我们没有加入 '-b' 参数（此参数告诉 Cscope 生成数据库以后就退出），您会发现您仍然在 Cscope 的界面（curese-based GUI）中。在这里你就可以做一些 搜索操作（提示：您可以使用方向键改变搜索类型，使用 'tab' 建可以在交替搜索类型和搜索结果）。键入搜索列表左边的数字，Cscope 就会使用 Vim 打开对应的地址（如果您指定了 EDITOR 环境变量，那么调用的编辑器可能不是 Vim）。当您退出 Vim，Cscope 又将您带入到其主界面中，非常的好用。

可惜 Cscope 有一个缺陷：您如果打算编辑一个新的搜索结果，那么您必须退出当前的编辑状态。解决这一办法就是 Vim 对应的插件。使用 CTRL-D 推出 Cscope。

打开 Vim，如果您使用 C 代码中已有的标记（symbol）（例如：'vim -t main'），那么 Vim 将跳转到您所希望的地方。

将光标移动到程序中的各个 C 标签中。连续的快读的键入 "CTRL - \ S"（就是按下 CRTL - 反斜线，然后再按 'S'），然后你会看见在 Vim 底部弹出一个窗口（Window），它显示了程序中所对应的标记。选择相应的一个并键入回车，就会跳到对应的位置。和 ctags 一样，你可以键入 "CTRL-t" 就可以回到搜索前的位置。

助记符：'\'（反斜线）是在按键 '/' 的右边（它用来对应 ctags 搜索）。

继续做一些同样的搜索，不过这次是使用 "CTRL-空格 s"。这次，Cscope 搜索的结果将出现在 Vim 新分割的水平窗口中。[如果您还未使用过 Vim 的多窗口功能：移动使用 'CTRL-W w'（或者使用 'CTRL-W' 加方向键，或者 CTRL-W h/j/k/l 对应 左/上/下/右），关闭窗口使用 'CTRL-W c'（或者使用办法 ':q'），仅仅使用当前窗口使用 'CTRL-W o'，分割两个同样的窗口使用 'CTRL-W s'（或者使用 'CTRL-W v' 垂直分割），在一个新窗口打开文件为 ':spl[it] 文件名']

助记符：各分割条分割了您的 Vim 窗口。

现在使用 "CTRL-空格 CTRL-空格 s"（按住 CTRL 键同时敲两次空格）执行新的搜索命令。如果您的手指还不是足够的灵活，建议您编辑 cscope_maps.vim 文件更改 Vim 中的 timeout 设置，其在文件中的注释中有说明[事实上，我建议关闭 Vim 中那该死的 timeouts 选项]。如果运气足够好运行成功的话，那么就能看见一个垂直分割的搜索结果窗口（注意：这个功能在 5.x 版本中并不适用，垂直分割是 6.x 版本的新功能）。

直到目前为止，我们所介绍的搜索功能都是来自 'cscope_maps.vim' 文件中定义的一些快捷键。这些快捷键能迅速的找到光标所在的标签所对应所指向的位置（有点拗口）。另外一个常用的方式（使用 Vim 内建的 Cscope 支持），键入":cscope find symbol foo"（更简洁的，"cs f s foo"）。执行水平分割的版本，使用 ":scscope"（或者直接使用 ":scs"），不过这仅仅在 6.x 版本下运行。 这样就可以非常容易的指定光标下未指定的标签，命令行的接口允许您指定你想要的任何标签名。我想你肯定有需要它的时候的。

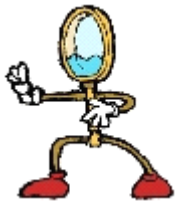
目前介绍的仅仅是对应 's' 的搜索命令，在所有结果中查询使用 'X' 命令。下面我们可以尝试其他的一些 Cscope 搜索命令：'g' 为全局搜索、'c' 为搜索指定的函数名称、'f' 为打开光标对应名称的文件（注意：Cscope 默认在 /usr/include 目录下获取 C 头文件，您使用此规则打开大部分的标准库）。上述这些是本人经常使用的一些功能，不过不仅仅是这些还有其他的（对应的其他功能您可以参阅 cscope_maps.vim 文件或者阅读 Cscope 的参考文档）。

虽然 Cscope 是为 C 语言设计的工具，但其作为富有弹性的工具仍然能够非常好的支持比如 C++ 和 Java 等其他语言。 你可以认为这是一个通用的 'grep' 数据库，同时还具有其他一切额外的功能，比如函数调用和变量定义。

默认情况下 Cscope 只会在当前目录下针对 C、lex 和 yacc（扩展名分别为 .c、.h、.l、.y）程序文件进行解析（如果指定了 -R 参数，则包含其自身的子目录），目前还没有办法让用户自己指定扩展名（是的，我们应该改变这种状况）。作为一个替代的方案，我们可以生成一个名为 'cscope.finds' 的文件列表，并交由 Cscope 读取解析（你可以任何使用通过 'cscope -i foofile' 命令调用）。 在 Unix 系统下你可以非常容易的（而且强大的）使用 'find' 命令生成这个列表：

```
find . -name '*.java' > cscope.files
```

现在运行 'cscope -b' 命令重新生成数据库（ -b 参数表示仅仅重新生成数据库而不调用 Cscope 前端），然后你就可以浏览 Java 程序中的变量等信息了。显然，这说明 Cscope 解析器是非常灵活的。



在大型项目中，您可能需要 -q 选项，并且（或者）使用功能更强大的 'find' 命令。此方面请您查看 使用 Cscope 进行大型项目管理指南 文档。

尝试配置 $CSCOPE_DB 环境变量指定 Cscope 数据库的存储位置，这样做的好处就是能在不同的目录中调用同一数据库。这功能对于那些设置了不同目录的项目尤其有用。注意：在使用此功能前，请进入绝对路径：cd 到 / 目录，并且运行：

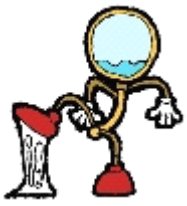
```
find /my/project/dir -name '*.c' -o -name '*.h' > /foo/cscope.files
```

然后再在同样的目录使用 Cscope 调用 cscope.files 文件（或者使用 'cscope -i /foo/cscope.files' 命令），然后再设置 $CSCOPE_DB 环境变量（指向 scscope.out 文件的结果）：

```
cd /foo
cscope -b
CSCOPE_DB=/foo/cscope.out; export CSCOPE_DB
```

（上述的命令是针对 Bourne/Korn/Bash shells 的：我忘记怎么使用 csh-based Shell 来定义全局变量了，我不想在这上面花太多的功夫。）

您现在您本机上的任何目录使用 'vim -t foo' 命令，然后 Vim 会跳转到 'foo' 所定义的正确位置。我曾经尝试编写一些 Shell 脚本（就是定义 CSCOPE_DB 环境变量），用来切换我不同的项目，比如可以简单的使用 'source projectA' 命令。



BUG: Cscope 在 15.4 以前的版本中，有一个非常愚蠢的 BUG。如果数据库名是另外名字而非 'cscope.out' 可能会造成 Vim 假死。解决这个问题的方法就是使用 '-f foo' 参数代替 'foot.out'，这样就好了（最新版本已经解决了该问题）。

就是这些了！ 您可以在 Vim 中使用 ":help cscope" 或者在命令行中键入 "man cscope" 查看更具体的 Cscope 说明。