在大型项目上使用Cscope（例如：Linux内核）

如果您需要涉足大型代码库，那么Cscope可能是一个特别有用的工具。通过能够进行快速，有针对性的搜索，而不是手工手动地遍历源文件，您可以节省很多时间（尤其是由于grep开始使用真正的大型代码库会花费一些时间）。

在本教程中，您将学习如何使用大型项目设置Cscope。我们以Linux内核源代码为例，但是其他任何大型项目（包括C ++或Java项目）的基本步骤都相同。

1. **获取源代码。**首先获取源代码。您可以从http://www.kernel.org下载Linux内核源代码。在本教程的其余部分中，我假设您已下载Linux 2.4.18并将其安装到`/home/jru/linux-2.4.18`。

   注意：请确保您有足够的磁盘空间：仅内核tarball就是30 MB，它会扩展为150 MB的源代码，我们将生成的Cscope数据库将吞噬另外20-100 + MB（取决于如何您决定要包含在数据库中的很多内核代码）。如果需要，可以将Cscope数据库放置在与源代码不同的磁盘分区上。

2. **找出要放置Cscope数据库文件的位置。**我假设您将`/ home / jru / cscope`用作存储数据库和关联文件的目录。

3. **生成带有要扫描文件列表的`cscope.files`。**对于某些项目，您可能希望在Cscope数据库的项目目录中包括每个C源文件。在这种情况下，您可以跳过此步骤，而只需在项目的顶级目录中使用“`cscope -R`”来构建您的Cscope数据库。但是，如果您要排除某些代码，并且/或者您的项目包含C ++或Java源代码（默认情况下，Cscope仅解析带有`.c`，`.h`，`.y`或`.l`扩展名的文件），则需要生成一个名为`cscope.files`的文件，其中应包含您希望进行Cscope扫描的所有文件的名称（每行一个文件名）。

   您可能想要使用绝对路径（至少在计划使用编辑器中使用Cscope数据库的情况下），以便可以从创建目录以外的目录使用数据库。我显示的命令将首先从cd转到root，以便find打印出绝对路径。

   对于许多项目，您的find命令可能很简单

   ```
   cd /
   找到/ my / project / dir -name'* .java'> /my/cscope/dir/cscope.files
   ```

   对于Linux内核，这有点棘手，因为我们要排除docs和scripts目录中的所有代码，以及所有芯片的所有架构和汇编代码，除了心爱的Intel x86（我猜是您感兴趣的架构）。此外，在此示例中，我排除了所有内核驱动程序代码（它们使解析的代码量增加了一倍以上，这使Cscope数据库膨胀，并且它们包含许多重复的定义，这通常使搜索变得更加困难。如果您对驱动程序代码，省略下面的相关行，或对其进行修改以仅打印出您感兴趣的驱动程序文件）：

   ```
   LNX = /家庭/jru/linux-2.4.18
   cd /
   找到$ LNX \
        -路径“ $ LNX / arch / *”！-path“ $ LNX / arch / i386 *” -prune -o \
        -路径“ $ LNX / include / asm- *”！-path“ $ LNX / include / asm-i386 *” -prune -o \
        -path“ $ LNX / tmp *”-修剪-o \
        -path“ $ LNX / Documentation *”-修剪-o \
        -path“ $ LNX / scripts *”-修剪-o \
        -path“ $ LNX / drivers *”-修剪-o \
        -名称“ *。[chxsS]” -print> /home/jru/cscope/cscope.files
   ```

虽然find命令的编写可能有些棘手，但对于大型项目，它们比手动编辑文件列表要容易得多，并且您还可以剪切和粘贴其他人的解决方案。

4. **生成Cscope数据库。**现在是时候生成Cscope数据库了：

```
cd / home / jru / cscope # 带有'cscope.files'的目录
cscope -b -q -k
```

该-b标志告诉Cscope生成数据库，而不是启动Cscope的GUI。该-q会附加一个“倒排索引”要创建文件，这使得搜索运行得更快的大型数据库。最后，-k设置Cscope的“内核”模式-它不会在/ usr / include中查找源文件中#include的任何头文件（这在将Cscope与操作系统和/或C一起使用时尤其有用。库源代码，如此处所示）。

在我的900 MHz Pentium III系统（带有标准IDE磁盘）上，解析Linux源的这一子集仅需12秒，并生成3个文件（cscope.out，cscope.in.out和cscope.po.out）。总共占用25兆字节。

5. **使用数据库。**如果您喜欢使用vim或emacs / xemacs，建议您学习如何在这些编辑器之一中运行Cscope，这将使您可以在编辑器中轻松运行搜索。我们有一个针对Vim的教程，emacs用户当然足够聪明，可以从Cscope发行版的cscope / contrib / xcscope /目录中的有用注释中找出所有内容。

否则，您可以使用独立的Cscope基于curses的GUI，该GUI使您可以运行搜索，然后启动您喜欢的编辑器（即，在您的环境中设置为$ EDITOR的默认值，或者默认为'vi'）在确切的行上打开搜索结果。

如果您使用独立的Cscope浏览器，请确保通过调用它

```
cscope -d
```

这告诉Cscope不要重新生成数据库。否则，您将不得不等待Cscope检查修改后的文件，这对于大型项目可能会花费一些时间，即使没有文件更改也是如此。如果您不小心运行了'cscope'，而没有任何标志，则还会导致在不使用快速索引或内核模式的情况下从头开始重新创建数据库，因此您可能需要重新运行上述原始cscope命令才能正确地重新创建数据库。

6. **当源代码更改时，重新生成数据库。**

如果您的项目中有新文件，请重新运行“查找”命令以更新cscope.files（如果您正在使用它）。

然后，只需以与最初生成数据库时相同的方式（在同一目录中）调用cscope（即cscope -b -q -k）。

Jason Duell的教程

返回Cscope主页