范围

名称

cscope-交互式检查C程序

概要

cscope的 [**-bCcdehkLlqRTUuV**] [**-F** symfile **]** [**-f** reffile **]** [**-I** INCDIR **]** [**-i** namefile **]** [**-num图案]** [**-p** Ñ **]** [**-s** DIR **]**

描述

cscope 是一种交互式的，面向屏幕的工具，允许用户浏览C源文件中的指定代码元素。

默认情况下， *cscope* 检查当前目录中的C（.c和.h），lex（.l）和yacc（.y）源文件。 也可以为在命令行上命名的源文件调用*cscope*。无论哪种情况， *cscope都会* 在标准目录中搜索它在当前目录中找不到的#include文件。 *cscope* 默认使用符号交叉引用cscope.out在文件中找到函数，函数调用，宏，变量和预处理器符号。

cscope 首次在要浏览的程序的源文件上使用该符号交叉引用时。在后续调用中， 仅当源文件已更改或源文件列表不同时， *cscope*才会重建交叉引用。重建交叉引用时，将从旧的交叉引用中复制未更改文件的数据，这使得重建比初始构建要快。

选件

以下选项可以任意组合出现：

-b
: 仅构建交叉引用。

-C
: 搜索时忽略字母大小写。

-C
: 在交叉引用文件中仅使用ASCII字符，即不要压缩数据。

-d
: 不要更新交叉引用。

-e
: 禁止在文件之间使用<Ctrl> -e命令提示符。

-F符号文件
: 从symfile读取符号参考线。（符号参考文件是由>和>>创建的，也可以使用<命令读取，如下文``发出后续请求"所述）。

-f引用文件
: 使用reffile作为交叉引用文件名，而不是默认的cscope.out。

-H
: 查看长期使用帮助显示。

-我incdir

在incdir中查找（在INCDIR中查找头文件的标准位置，通常为/ usr / include之前），以查找名称不是以``/"开头且未在命令行或名称文件中指定的任何#include文件下面。（#include文件可以用双引号或尖括号指定。）除了当前目录（首先搜索）和标准列表（最后搜索）之外，还搜索incdir目录。如果出现多次-I，则按照在命令行中出现的顺序搜索目录。

-i名称文件

浏览名称文件中列出名称的所有源文件（文件名之间用空格，制表符或换行符分隔），而不使用默认文件（cscope.files）。如果指定了此选项，则cscope会忽略命令行中出现的所有文件。参数namefile可以设置为``-"以接受来自stdio的文件列表。名称文件中包含空格的文件名必须用“双引号”引起来。在此类带引号的文件名中，任何双引号和反斜杠字符都必须由反斜杠转义。

-k

``内核模式"（Kernel Mode）在构建数据库时会关闭默认的include dir（通常是/ usr / include）的使用，因为内核源代码树通常不使用它。

-L

与-num pattern选项一起使用时，请使用面向行的输出进行一次搜索。

-l

面向行的界面（请参阅下面的``面向行的界面"）。

-num模式

转到输入字段num（从0开始计数）并找到模式。

-P路径

在预构建的交叉引用文件中将相对文件名放在路径之前，因此您不必更改到构建交叉引用文件的目录。该选项仅对-d选项有效。

-pn

显示最后n个文件路径组件，而不显示默认组件（1）。使用0根本不显示文件名。

-q

通过反向索引启用快速符号查找。此选项使cscope除了普通数据库外还创建另外2个文件（默认名称为``cscope.in.out"和``cscope.po.out"）。这允许使用更快的符号搜索算法，从而为大型项目提供明显更快的查找性能。

-R

递归源文件的子目录。

-s目录

在目录中查找其他源文件。如果在命令行上提供了源文件，则将忽略此选项。

-T

仅使用前八个字符来匹配C符号。如果最小长度大于8个字符，则包含除点号（。）以外的特殊字符的正则表达式将不与任何符号匹配。

-U

检查文件时间戳。即使没有文件更改，此选项也将更新数据库上的时间戳。

-u

无条件构建交叉引用文件（假定所有文件都已更改）。

-V

在屏幕的第一行上打印cscope的版本号。

-I，-c，-k，-p，-q和-T选项也可以位于cscope.files文件中。

请求初始搜索

交叉引用准备好后，cscope将显示以下菜单：

找到这个C符号：

找到此函数定义：
查找此函数调用的函数：

查找调用此函数的函数：
查找以下文本字符串：
更改此文本字符串：
找到以下egrep模式：
查找此文件：
查找包含该文件的文件＃：

反复按<Up>或<Down>键移动到所需的输入字段，键入要搜索的文本，然后按<Return>键。

发出后续请求

如果搜索成功，则可以使用以下任何单个字符命令：

0-9a-zA-Z
　　编辑给定行号引用的文件。
<**空格**>
　　显示下一组匹配的行。
<**标签**>
　　在菜单和匹配行列表之间切换
<**上**>
　　移至上一个菜单项（如果光标在菜单中）或移至上一个匹配行（如果光标在匹配行列表中）。
<**下**>
　　移至下一个菜单项（如果光标在菜单中）或移至下一个匹配的行（如果光标在匹配的行列表中）。
+
　　显示下一组匹配的行。
--
　　显示上一组匹配行。
^ **e**
　　按顺序编辑显示的文件。
>
　　将显示的行列表写入文件。
>>
　　将显示的行列表追加到文件中。
<
　　像-F选项一样，以符号引用格式（由>或>>创建）从文件中读取行。
^
　　通过shell命令过滤所有行，并显示结果行，替换已存在的行。
|
　　将所有行通过管道传递给shell命令，并在不更改它们的情况下显示它们。

在任何时候，这些单字符命令都可以使用：

<**返回**>
　　移至下一个输入字段。
^ **n**
　　移至下一个输入字段。
^ **p**
　　移至上一个输入字段。
^ **y**

搜索最后键入的文本。

^ b

移至上一个输入字段并搜索模式。

^ f

移至下一个输入字段并搜索模式。

^ c

搜索时切换忽略/使用字母大小写。（忽略字母大小写时，搜索``FILE"将匹配``File"和``file"。）

^ r

重建交叉引用。

!

启动一个交互式外壳程序（键入^ d返回到cscope）。

^ l

重画屏幕。

?

提供有关cscope命令的帮助信息。

^ d

退出cscope。

注意：如果要搜索的文本的第一个字符与 上述命令之一匹配，请先键入（反斜杠）将其转义。

用新文本替换旧文本

键入要更改的文本后，cscope将提示您输入新文本，然后显示包含旧文本的行。使用以下单字符命令选择要更改的行：

0-9a-zA-Z

标记或取消标记要更改的行。

*

标记或取消标记所有要更改的显示行。

<**空格**>

显示下一组线。

+

显示下一组线。

--

显示前一组线。

一个

标记或取消标记所有要更改的行。

^ d

更改标记的行并退出。

<Esc>

退出而不更改标记的行。

!

启动一个交互式外壳程序（键入^ d返回到cscope）。

^ l

重画屏幕。

?

提供有关cscope命令的帮助信息。

特殊键

如果您的终端具有可以在vi中使用的箭头键，则可以使用它们在输入字段中移动。向上箭头键对于移至上一个输入字段很有用，而不是重复使用<Tab>键。如果具有<CLEAR>，<NEXT>或<PREV>键，

它们将分别用作^ l，+和-命令。

面向行的界面

使用-l选项，您可以使用cscope，在这种情况下面向屏幕的界面将无用，例如，来自另一个面向屏幕的程序。

当准备好输入行时，cscope会显示>>提示，该输入行应立即以字段编号（从0开始计数）开始，然后是搜索模式，例如，``lmain"查找主函数的定义。

如果只想进行一次搜索，请使用-L和-num模式选项代替-l选项，并且不会显示>>提示符。

对于-l，cscope输出参考线的数量cscope：2行

对于找到的每个引用，cscope输出由文件名，函数名，行号和行文本组成的行，并用空格分隔，例如，main.c main 161 main（argc，argv）

请注意，与面向屏幕的界面不同，不会调用编辑器来显示单个引用。

搜索时，可以使用c命令切换忽略/使用字母大小写。（忽略字母大小写时，搜索``FILE"将匹配``File"和``file"。）

您可以使用r命令重建数据库。

当检测到文件结尾或输入行的第一个字符为``^ d"或``q"时，cscope将退出。

环境变量

CSCOPE_EDITOR
: 覆盖EDITOR和VIEWER变量。如果您希望对cscope使用与由EDITOR / VIEWER变量指定的编辑器不同的编辑器，请使用此功能。

CSCOPE_LINEFLAG
: 编辑器的行号标志的格式。默认情况下，cscope通过等效于``editor + N file"的方式调用编辑器，其中``N"是编辑器应跳转到的行号。emacs和vi都使用此格式。如果您的编辑器需要其他内容，请在此变量中指定它，并以``%s"作为行号的占位符。例如：如果您的编辑器需要作为``编辑器-＃103文件"来调用以转到第103行，请将此变量设置为``-＃%s"。

CSCOPE_LINEFLAG_AFTER_FILE
: 如果需要在要编辑的文件名之后使用行号选项调用编辑器，则将此变量设置为``是"。要继续上述CSCOPE_LINEFLAG中的示例，请执行以下操作：如果您的编辑器需要查看``编辑器文件-＃number"，请设置此环境变量。大多数标准编辑器（vi，emacs）的用户都不需要设置此变量。

编辑
: 首选编辑器，默认为vi。

家
: 主目录，该目录在登录时自动设置。

包含
: 用冒号分隔的目录列表以搜索#include文件。

贝壳
: 首选外壳，默认为sh。

来源
: 用冒号分隔的目录列表以搜索其他源文件。

术语
终端类型，必须是屏幕终端。
术语表
终端信息目录的完整路径名。如果您的终端不在标准terminfo目录中，请参阅curses和terminfo了解如何进行自己的终端描述。
TMPDIR
临时文件目录，默认为/ var / tmp。
查看器
首选的文件显示程序（例如less），它会覆盖EDITOR（请参见上文）。
虚拟路径
用冒号分隔的目录列表，每个目录下面都有相同的目录结构。如果设置了VPATH，则cscope在指定的目录中搜索源文件；否则，将在源目录中搜索源文件。如果未设置，则cscope仅在当前目录中搜索。

档案

cscope.files
包含-I，-p，-q和-T选项以及源文件列表的默认文件（由-i选项覆盖）。
cscope.out
符号交叉引用文件（由-f选项覆盖），如果无法在当前目录中创建，则将其放在主目录中。
cscope.in.out
cscope.po.out
包含用于快速符号搜索的反向索引的默认文件（-q选项）。如果使用-f选项来重命名交叉引用文件（因此它不是cscope.out），则将
-in和.po 添加到随-f提供的名称中来创建这些反向索引文件 的名称。例如，如果您指定-f xyz，则这些文件将被命名为xyz.in和xyz.po。
印迪尔
#include文件的标准目录（通常为/ usr / include）。

告示

cscope可以 识别以下形式的函数定义：

fname空白（args）白色arg_decs白色{
哪里：
fname 是函数名称
空白
为零或多个空格或制表符，不包括换行符
args
是不包含``"或换行符的任何字符串
白色
为零或多个空格，制表符或换行符
arg_decs
是零个或多个参数声明（arg_decs可能包含注释和空格）

函数声明不必从行的开头开始。返回类型可以在函数名称之前；cscope仍会识别该声明。偏离此格式的函数定义将不会被cscope识别。

菜单选项``查找此函数调用的函数"的搜索输出的``函数"列：输入字段将仅显示该行中调用的第一个函数，即此函数

```
e ( )
{
      return ( f ( ) + g ( ) ) ;
}
```

显示将是

```
该函数调用的函数：e
文件功能行
ac f 3 return ( f ( ) + g ( ) ) ;
```

有时，由于#if语句中的花括号，可能无法识别函数定义或调用。类似地，变量的使用可能被错误地识别为定义。

一个 **类型定义** 前面的预处理语句名会被错误地认为是一个全球性的定义，例如，

```
LDFILE *
＃如果AR16WR
```

预处理程序语句还可能阻止识别全局定义，例如，

```
字符标志
#ifdef ALLOCATE_STORAGE
    = -1
#endif
;
```

函数内部的函数声明被错误地识别为函数调用，例如，

```
f ( )
{
      void g ( ) ;
}
```

被错误地识别为对g的调用。

cscope 通过查找class关键字识别C ++类，但不识别结构也是一个类，因此它不识别结构中的内联成员函数定义。它还不希望*typedef中*的class关键字 ，因此它错误地将X识别为

```
typedef类X * Y;
```

它还无法识别运算符功能定义

```
布尔功能:: operator ==（const Feature＆other）
{
```

```
  ...
}
```

它也不用函数指针参数识别函数定义

```
ParseTable :: Recognize（int startState，char * pattern，
  int finishState，void（* FinalAction）（char *））
{
  ...
}
```

指数

该文档是由man2html使用手册页创建的。 时间：2002年3月13日格林尼治标准时间04:41:42