Using Cscope and Ctags to navigate Pintos code

Posted on September 18, 2012

This post will guide you how to use cscope and ctags utility to navigate through the Pintos code.

While studying PINTOS source code, its necessary to have some sort of intellisense like other IDEs like Eclipse, Netbeans etc. have. To achieve this, we have development tools like cscope and ctags.

Cscope builds a database of funtions and variables. One can navigate to declarations, function call, function definitions and all using cscope commands.

First install cscope on ubuntu using:

$ sudo apt-get install cscope ctags vim

Go to directory in terminal where you want to use cscope, ctags which in our case will be "$HOME/os-pg/pintos/src" and fire below mentioned command:

$cscope -Rvkq

It creates database files for cscope named as cscope.out.

Cscope can be used directly in two ways:

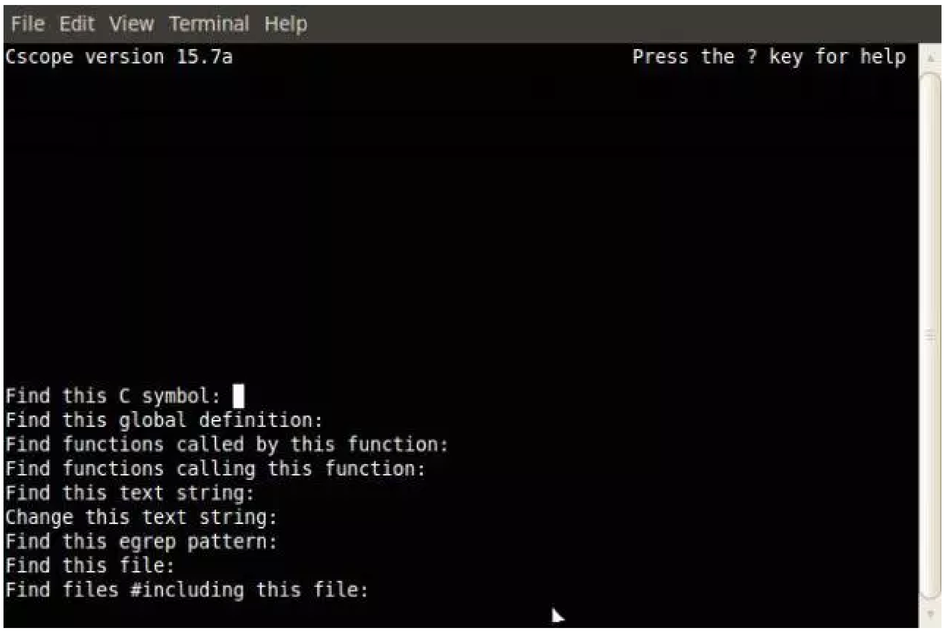
1. Directly from terminal with cscope's default interface.
2. Through vim editor

Explanation:

1. Using Cscope through terminal:

$ cscope -d

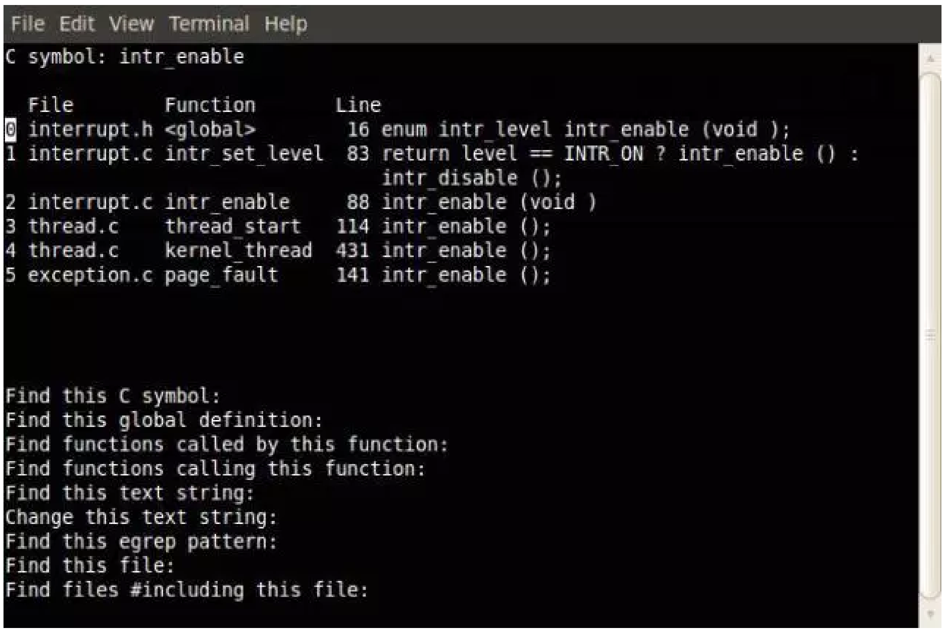
This will open interface as shown in figure :



```
Cscope version 15.7a                                    Press the ? key for help









Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
```

This interface can be used to find symbols, global definitions, functions and included files.

Just type in the symbol/function name and hit ‘enter’. For example, if I type ‘intr_enable in symbol find section , and hit ‘enter’:



```
C symbol: intr_enable

  File        Function       Line
0 interrupt.h <global>        16 enum intr_level intr_enable (void );
1 interrupt.c intr_set_level  83 return level == INTR_ON ? intr_enable () :
                                 intr_disable ();
2 interrupt.c intr_enable     88 intr_enable (void )
3 thread.c    thread_start   114 intr_enable ();
4 thread.c    kernel_thread  431 intr_enable ();
5 exception.c page_fault     141 intr_enable ();




Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
```

Press the number on left side to open the file. The file will open in vi editor at that particular location where the searched symbol exists. Its easy to search using this interface. To exit, press ‘ctrl-d’.

2. Using Cscope through vi editor:

For using it through vi editor, you need to install ctags:

$ sudo apt-get install ctags

Few steps to make cscope and ctags interact with vi editor:

$ ctags -R *

2. Add tags to cscope:

Add all the header and c files to a file “cscope.out” using following command again from root directory:

$ find -name *.[ch] > cscope.files

Now we are ready to work. Its best to place the file list in file called cscope.files – which is a default location for cscope. In other cases we would have to manually pass the filelist file to cscope using the -i parameter.

3. Navigate using vim:

Now we are ready to use cscope and ctags with vim, open any file in vim, for eg:

$ vim threads/interrupt.c

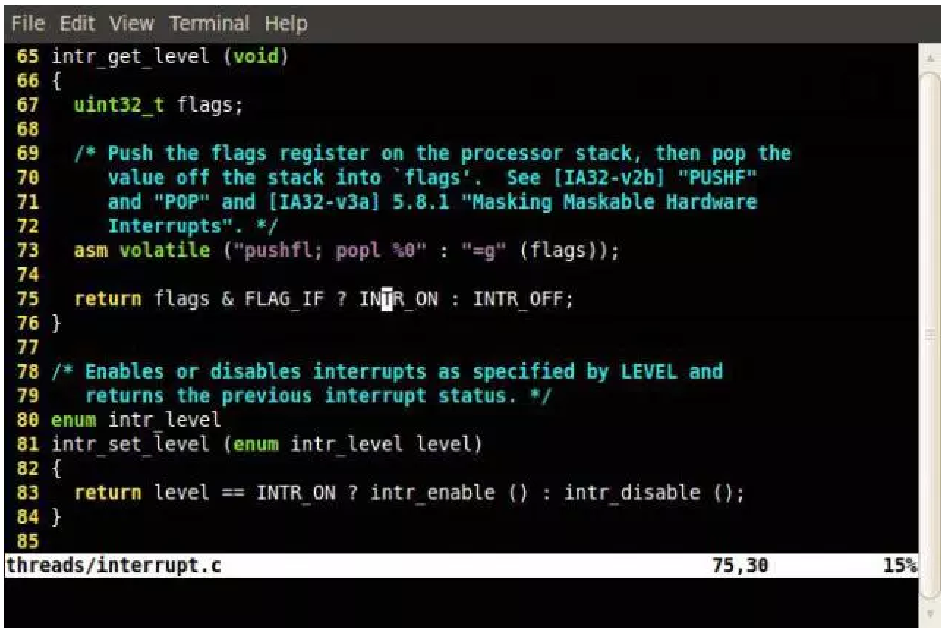
Now we need to inform vim about cscope database.

Inside vim go in command mode (Esc) and type,

:cscope add cscope.out

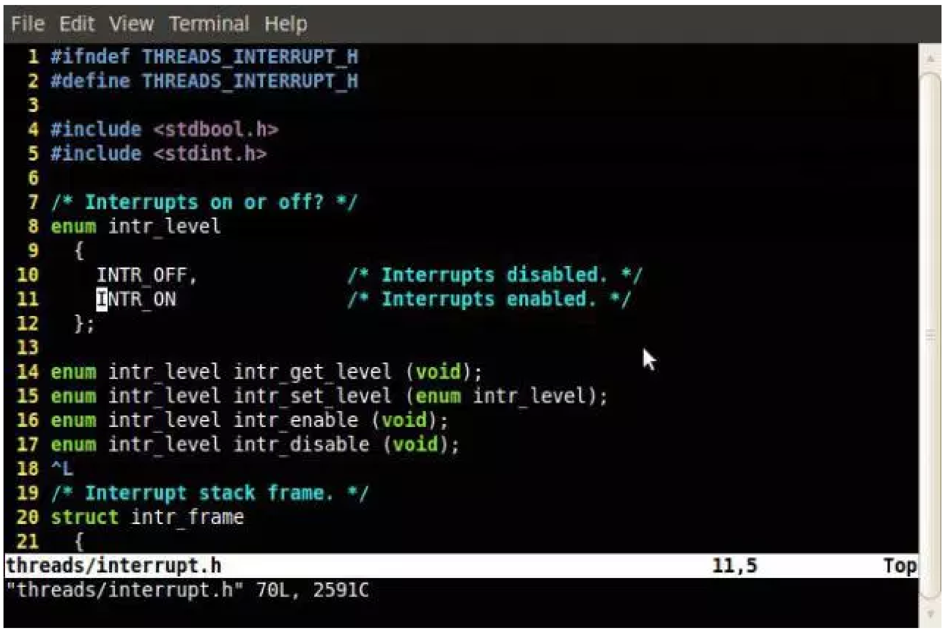
The file cscope.out was created in the pintos/src directory of PINTOS, using ‘cscope -Rvkq’ command.

If I am at line 75 and want to know what the INTR_ON is? Then I can point cursor to it and use the shortcut provided by ctags (ctrl +]) and it will take me to the definition of that variable in interrupt.h



```
65 intr_get_level (void)
66 {
67   uint32_t flags;
68
69   /* Push the flags register on the processor stack, then pop the
70      value off the stack into `flags'.  See [IA32-v2b] "PUSHF"
71      and "POP" and [IA32-v3a] 5.8.1 "Masking Maskable Hardware
72      Interrupts". */
73   asm volatile ("pushfl; popl %0" : "=g" (flags));
74
75   return flags & FLAG_IF ? INTR_ON : INTR_OFF;
76 }
77
78 /* Enables or disables interrupts as specified by LEVEL and
79    returns the previous interrupt status. */
80 enum intr_level
81 intr_set_level (enum intr_level level)
82 {
83   return level == INTR_ON ? intr_enable () : intr_disable ();
84 }
85
threads/interrupt.c                                   75,30          15%
```

After I press ‘ctrl+]’ :



```
#ifndef THREADS_INTERRUPT_H
#define THREADS_INTERRUPT_H

#include <stdbool.h>
#include <stdint.h>

/* Interrupts on or off? */
enum intr_level
  {
    INTR_OFF,             /* Interrupts disabled. */
    INTR_ON               /* Interrupts enabled. */
  };

enum intr_level intr_get_level (void);
enum intr_level intr_set_level (enum intr_level);
enum intr_level intr_enable (void);
enum intr_level intr_disable (void);
^L
/* Interrupt stack frame. */
struct intr_frame
  {
threads/interrupt.h                                    11,5           Top
"threads/interrupt.h" 70L, 2591C
```

Thus ctags + cscope helps us to navigate through the source code faster.

Function definitions: One can navigate to function definition from function call location by following the same procedure as for variables with ‘ctrl+]’.

CTRL +] to jump to function or data_type or variable declaration. We can use CTRL + t to go back. For using cscope through vim, goto command mode and type :cs and the cscope interface will pop up.

Thanks Ashay Raut for compiling this tutorial.

Happy Coding!!

– Rasesh

Share this:

Tweet Share 1

Like

Be the first to like this.

About Rasesh Mori

I am Rasesh Mori, Software Development Engineer at Amazon since July 2013 after completing masters in Computer Science and Engineering from IIIT, Hyderabad.

View all posts by Rasesh Mori →

This entry was posted in Pintos and tagged assignment, Cscope, Ctags, global definitions, Pintos. Bookmark the permalink.

Pintos IIITH (OS – PG)

Create a free website or blog at WordPress.com.