Ctags Tutorial

Ctags is a tool that makes it easy to navigate large source code projects. It provides some of the features that you may be used to using in Eclipse or other IDEs, such as the ability to jump from the current source file to definitions of functions and structures in other files. Ctags will make it much easier to find the Linux kernel files that you have to modify for your CSE 451 projects. Ctags also supports many languages besides C, so you may find it useful for future projects.

Ctags should already be installed on CSE instructional servers such as `forkbomb` and `attu`. Ctags is first run on its own to generate a "tags" file, then it is invoked from within another Linux text editor such as Emacs or Vim.

These steps assume you want to use Ctags on the Linux kernel, but should generalize to other projects.

Important

If you are not on `forkbomb` or `attu`, make sure that the system you are using has "Exuberant Ctags" installed, rather than the original "Ctags," by running `ctags --version`.

Tip

Like all Linux programs, Ctags has a man page (`man 1 ctags`). All of the information in this tutorial, and lots more advanced information, can be found there.

Ctags with Emacs

Tip

If you are unfamiliar with Emacs, you should go through the Emacs tutorial. Run `emacs`, then press `C-h t` (Ctrl+h, then t) to begin the interactive tutorial. It shouldn't take too long, and it's worth your time. You can also check out the online tour. If you get stuck, press `C-g` to cancel pending commands, and exit Emacs by pressing `C-x C-c`.

Disclaimer

The author of this tutorial is not an Emacs expert; if you notice potential flaws or improvements, please contact him.

1. `cd` to the root directory of your Linux kernel code:

```
cd /cse451/user/project1/linux-2.6.13.2/
```

2. Run Etags (Ctags for Emacs) over the kernel to generate the `TAGS` file. `etags` will overflow its stack if called recursively over the entire kernel, so we use the `find` command to find all of the `.c`, `.h`, and `.S` (assembly) files in the kernel, then tell `etags` to append the tags in those files to the `TAGS` file. For Linux 2.6.13, this should only take a minute or so:

```
find . -type f -iname "*.[chS]" | xargs etags -a
```

Note

You may see messages like "Warning: cannot open source file '...' : Permission denied" while etags is building the tags file. These warnings can be ignored.

3. Open any Linux source file in Emacs and use the following basic commands:

Keyboard command	**Action**
`M-. <RET>`	Jump to the tag underneath the cursor
`M-. <tag> <RET>`	Search for a particular tag
`C-u M-.`	Find the next definition for the last tag
`M-*`	Pop back to where you previously invoked "M-."

Important

The first time you run an Etags command within Emacs, you may have to specify the location of your `TAGS` file (i.e. `/cse451/.../linux-2.6.13.2/TAGS`). Say yes when prompted to load the really big tags file.

The first command is probably the one you will use most often: it jumps to the definition of the tag (function name, structure name, variable name, or pretty much anything). The second command can be used to search for any tag in the `TAGS` file, regardless of the file you are currently viewing. Sometimes Etags will find multiple definitions for a given tag; when this is the case, use the third command to jump through the possible definitions until you find the one that you want. Finally, use the fourth command to jump back up in the tag "stack."

You'll probably find that for some tags (common structures, for example), Etags finds hundreds or thousands of uses in the code, and jumping through them (with the third command above) to try to find the original definition is useless. In this case, you can run the following two commands to list all of the uses of a given `<tag>`:

```
M-x tags-apropos <RET>
Tags apropos (regexp): <tag> <RET>
```

This will display a list of the tag definitions in another buffer. Switch to the new buffer (`C-x o`), scroll through the list of definitions to the one that you want, then press Enter to open the file. When you're done, instead of jumping back up in the tag stack, close the new buffer (`C-x k`). To switch back to your original buffer and expand it, use `C-x o` to switch to it, then `C-x 1` to expand.

Note

Even the list of all definitions given by `tags-apropos` may be too large to find the definition that you're looking for. This is mostly a problem for structs (`struct inode`, for instance) that are used frequently in the kernel. You should still find Etags useful for jumping to function definitions and less-commonly-used structs. Ctags for Vim appears to do a better job of separating "definitions" from "uses" in its tags file, so this is less of a problem for Vim; for Emacs, there may be other ways to mitigate this problem (see this page, for example). Alternatively, you may wish to use cscope to find function and structure definitions, or just use the third step of the Vim instructions below.

Ctags with Vim

Note

These commands were tested with Vim (7.2), but will likely work with Vi or other Vi emulators as well.

1. `cd` to the root directory of your Linux kernel code:

   ```
   cd /cse451/user/project1/linux-2.6.13.2/
   ```

2. Run Ctags recursively over the entire kernel to generate the `tags` file. For Linux 2.6.13, this should only take a minute or so:

   ```
   ctags -R *
   ```

   Note

   You may see messages like "Warning: cannot open source file '...' : Permission denied" while `ctags` is building the tags file. These warnings can be ignored.

3. To search for a specific tag and open Vim to its definition, run the following command in your shell:

   ```
   vim -t <tag>
   ```

4. Or, open any Linux source file in Vim and use the following basic commands:

Keyboard command	Action
`Ctrl-]`	Jump to the tag underneath the cursor
`:ts <tag> <RET>`	Search for a particular tag
`:tn`	Go to the next definition for the last tag
`:tp`	Go to the previous definition for the last tag
`:ts`	List all of the definitions of the last tag
`Ctrl-t`	Jump back up in the tag stack

The first command is probably the one you will use most often: it jumps to the definition of the tag (function name, structure name, variable name, or pretty much anything) under the cursor. The second command can be used to search for any tag, regardless of the file that is currently opened. If there are multiple definitions/uses for a particular tag, the `tn` and `tp` commands can be used to scroll through them, and the `ts` command can be used to "search" a list for the definition you want (useful when there are dozens or hundreds of definitions for some commonly-used struct). Finally, the last command is used to jump back up in the tag stack to the location you initiated the previous tag search from.

Author: Peter Hornyack (pjh@cs)