

# Advanced Exercises Set 4

## Text Editors

EECS 201 Winter 2020

### Submission Instructions

To receive credit for this assignment you will need to stop by someone's office hours, demo your running code/system, and answer some questions. You may choose the exercises that you wish to do from this set. Each exercise is denoted by its point count. **Extra credit is given for early turn-ins of advanced exercises. These details can be found on the website under the advanced homework grading policy.**

## 1 Some Tools that Support Other Tools (10)

*For part 1, you may use vim or Emacs, whichever you are more comfortable with.*

Integrated Development Environments (IDEs), such as Visual Studio, Xcode, or Eclipse provide many features beyond simple text editing.<sup>1</sup> Some of the most useful features rely on an understanding of C/C++ code, allowing you to jump from a function's use to its definition, inspect the types of complex structures, and assist authoring by suggesting possible completions.

Vim and Emacs both support rudimentary autocompletion out-of-the-box. Try editing an existing file and start typing a function you've used before, then hit `ctrl-n` and `ctrl-p` in vim or `M-/` in Emacs. This autocompletion has no understanding of C, it simply does string matching with any other strings anywhere else in the open files.<sup>2</sup>

There are two utilities, `ctags/etags` (vim/Emacs respectively) and `cscope` that can be integrated with your editor to provide some contextually aware editing assistance. `ctags/etags` works pretty much out of the box. Somewhat annoyingly, `cscope` requires you to generate a list of files for it to process. You may find that the `find` utility, and its filetype filters in particular, are very helpful for generating this file list.

Set up `ctags/etags` and `cscope` for any project of your choosing, and integrate them with your editor of choice.

### Submission checkoff:

- Show off `ctags/etags` usage
  - What did you have to do to generate tags for `ctags/etags`?
  - How often do you need to generate tags (after what changes do you need to regenerate tags)?
  - What did you have to do to integrate `ctags/etags` with your editor?
- Show off `cscope` usage
  - How did you generate a file list for `cscope`?
  - How often do you need to run `cscope`?
  - What did you have to do to integrate `cscope` with your editor?
- Explain the difference between `ctags/etags` and `cscope`
  - Is one strictly better than the other or is it useful in some cases to set up both? Why?

---

<sup>1</sup>Interestingly, "simple text editing" is often what IDEs are worst at out-of-the-box. Hence part 3 of this assignment.

<sup>2</sup>Vim also understands how to parse C `#include` directives and Python `import` statements, and will search those files as well.

## 2 Extending Editors (3)

ctags/etags and cscope are programs that run independently of your text editor. You can also extend your editor with plugins. For vim, plugins are written in the custom language vimscript; Emacs itself is a Lisp interpreter, and plugins are written in Emacs Lisp. Sublime is written Python, as are its plugins. Atom in Javascript, et cetera. While writing a plugin is a great way to learn a lot about the internals of your editor, it can be a steep learning curve. Fortunately, the Internet has an enormous library of really interesting and powerful plugins. As an example, there are plugins that solve the annoyance from part 1, that will automatically regenerate tags whenever needed. Pick (at least) two plugins to install in your editor.<sup>3</sup> You may find it is easier to first install a plugin manager and then install plugins.

### Submission checkoff:

- For each plugin:
  - What does this plugin do?
  - Show off how you use it
  - Why did you install this plugin?

## 3 Everything Old is New Again (2)

While advanced IDEs and new text editors come out remarkably often, people who have learned how to navigate and manipulate text using vim or Emacs are generally not willing to give that up. For this reason, there are plugins (or in some cases it's built-in) to provide vim-like or Emacs-like editing modes for nearly every editor.

Pick your favorite "advanced" GUI editor (suggestions Visual Studio, Xcode, Eclipse, Sublime, Atom, or Visual Studio Code – gvim doesn't count) and set it up to edit text in vim or Emacs mode. You may need to find and install a plugin first.

### Submission checkoff:

- Explain how to set up and use vim/Emacs mode for your editor

---

<sup>3</sup>How to find them, choose? Try Googling "best plugins for (editor)", pick whatever seems cool to you.