Advanced Exercise – Week 5

Due: Before February 18, 10:00PM* (ish, see below)

Contributing to an open source project

While version control is useful for your own projects and helps with group class projects, what really makes it awesome it how accessible it makes collaborating and contributing to big (or small) projects.

One thing that's awesome about computer science is that there are many opportunities for early-career students to make contributions – if you aren't familiar, check out some of the programs for CS freshman/sophomores at major companies; or check out alternative programs like Google Summer of Code ("Am I Good Enough?", the summer of code program can be an excellent way to spend a summer in place of a more traditional internship. Applications open March 20 and close April 3 this year)

Slightly less committal than a full-time job, you're also more than qualified to help contribute to open source projects. Sometimes this is contributing code. Many times the most helpful can be improving documentation – try to follow the installation instructions and take notes on what confused you and how might improve them.

For this exercise, the goal is to make a non-trivial contribution to an open source project.

> *non-trivial*, adj. – A loosely defined concept. More than just fixing the spelling of a few words. Rewriting a paragraph of documentation to be clearer is great though. A code fix could be a one character change if it fixes a bug, etc. Ultimately, it'll be up to the discretion of course staff who will know it when they see it, but the goal is to be reasonable. Something non-trivial should take more than two minutes, probably doesn't need to take more than fifteen.

Submission Instructions

Since you're sometimes working on someone else's schedule (i.e. don't push another project to merge your changes just to meet the deadline for your homework!), you need to have a substantial portion of the work done by the deadline.

> *substantial*, adj. – A loosely defined concept. Basically a well-formed commit or series of commits, an explanation to the project maintainer about what you'd like to change/fix and why, and submitted to the project. This should be something publically viewable / auditable (e.g. a pull request, a post to a mailing list, etc).
>
> Again, we'll know it when we see it :).

It's possible that the project will reject your change, sometimes not for a great reason. If you had what we deem to be a well-crafted request, we'll give credit even it was not merged. The goal is to get you some experience with learning how to submit changes to another project.

A good first place to start is by reading over GitHub's guide to contributing to open source projects.

One Option: Contributing to the class website

This class itself is an open source project!

If you haven't already, check out https://c4cs.github.io/reference

The goal is to build up a quick reference for common commands, as well as some examples of how to use them and any gotcha's they may have. The hope is that it's easier for people who are learning (i.e. you) to write documentation that's helpful for other people who are learning as opposed to course staff who don't really remember well what was confusing when we were first learning.

Look through the reference and find something that could use improvement. Or, add a new command that we've talked about in class or that you have other experience with that's not in the reference yet.

To ensure that people aren't working on the same things, before you get started, check to see if there's an open issue for what you want to work on. If no one's doing it yet, make a new issue so that others know what you're working on. If someone on the course staff opened an issue for something you'd like to work on please comment to claim.

Some help to get started

GitHub Pages

Notice how the course homepage is `c4cs.github.io`? GitHub has a really neat feature called GitHub Pages, that will turn a repository into a website – hosted for free. What's really nice about this is that it means it is very easy for many people to collaborate to develop a website, it's just a repository!

In the simplest setup, GitHub will simply serve the files in the repository as static web pages. Writing lots of HTML by hand, however, can be a pain, so GitHub supports a *site generator*, jekyll. Using jekyll, adding an update to the homepage is as simple as adding a text file. The site will also automatically hide updates older than one month.

Getting Going

First, grab a copy of course website repository (https://github.com/c4cs/c4cs.github.io). From there, crack open the `README` file. This is almost always a good place to start. It usually contains background on the project, some information regarding installation (if it's a package), dev environment setup, and anything else that might be pertinent to someone just diving into a new codebase.

Once things are running, it should print out

```
Server address:  http://127.0.0.1:4000/
```

Visit http://127.0.0.1:4000/ in your browser and you should see your own local copy of the course website! Try making some changes to the site and then refresh the page in your browser.

OR

(next page)

Another Option: Any project of your choosing

How to Get Started

Getting started is the hardest part. There is an overwhelming number of projects to choose from. Big projects, such as Mozilla, often have documentation explaining how to contribute, their bug trackers will track and let you query things like [good first bug], and their contributors are often friendly to beginners, with good advice for getting started. The negative with big projects is that they are big. You have to learn a large codebase in order to get started at all.

On the flip side, small projects, such as omnomnorth can be easier to make changes to, but harder to get started with as they do not have mature "getting started" guides.

If possible, I recommend picking something that you use. It's easier to motivate yourself to work on something you use, and there's something pretty cool about using your own software every day. Many things you (should) use are open source. Many libraries, such as the near-universal syntax highlighting library, the near-universal document converter, or a simple library for pretty terminal output have a lot of low-hanging fruit (check out the "Issues" tab).

Some local options

Are you a fan of Chez Betty? There are some open issues there. Speak a foreign language? Help extend Betty's translation support (or help translate any other project, there's huge demand for translations).

eecs.help, our office hours help queue is also open source.

Interested in getting into research? Some of the research labs at Michigan have big projects that could use help. Check out Alex Halderman research group projects: ZMap and Censys. Or stuff from my lab, Lab11, the embedded systems research group. Or Ed Olsen's Robotics group. If you're interested in research with someone, fixing one of their problems is a *great* way to introduce yourself :).