Bonus Advanced Homework
Upstreaming

Released: Friday, January 22, 11:00AM

Milestone 1 (Must start by) Friday, March 18

Due: Before lecture on Friday, April 8

Submission Instructions

This assignment is different than most. You have nearly 10 weeks, however you will have to start early because *you do not control the timetable.*

To receive credit for this assignment you will need to stop by someone's office hours, explain your improvement, and demonstrate your interaction with an open source project.

1 Upstreaming Code

One of the coolest parts of computer science is that with only a small amount of learning and experience, you are qualified to provide real contributions to software projects used by hundreds to millions of people around the world.[1] There is a lot of code to be written in the world and not nearly enough people qualified to write it.

Nearly everyone in the world uses open source software every day. As a computer scientist you will very likely use a lot more open source software than most people. Contributing back to an open source project is a great way to pay back into this ecosystem, a *great* way to gain some real-world software experience, and a nice feather for any résumé.[2]

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 and their developer(s) (if there's even more than one) may not have the time (or patience) to mentor young coders.

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).

[1] Don't believe me? Go check out Google Summer of Code's "Am I Good Enough?". The summer of code program can be an excellent way to spend a summer in lieu of a more traditional internship. Applications open February 29 and close March 25 this year.

[2] Though for many reasons, it should not *replace* your résumé. This post is a long read on the subject (it's also worth reading several of the linked posts). This area is interesting reading for understanding some of how computer science is evolving as a field culturally.

The Assignment:

Pick an active open source project and make a meaningful contribution to it.

> ***active*, adj.** A project used by a reasonable number (say 100+) of people that you don't know.
>
> ***meaningful*, adj.** A non-trivial change. More than a one-word documentation fix. Bigger documentation fixes, such as adding a thorough installation / getting started guide to a project that doesn't have one is a great contribution. Fixing an open issue/bug is great. Adding a feature is great. Anything 10 lines+ of code is likely good. A one character change that fixes a reported bug is also good, however.
>
> Ultimately, this is subjective. However, if you feel you can justify to a course staff that it was meaningful, then it probably was.

Having you contribution accepted, especially your first one, often takes some back-and-forth. The project owner will want you to make small changes. Be prepared for this, it doesn't mean that your overall patch is bad! This does mean, however, that it can take a little time for your patch to be accepted.

Submitting a patch to an open source project is a public process. Usually via something like a pull request if the project uses a collaboration site like GitHub or BitBucket, or by sending an e-mail to a public mailing list. To ensure you start early enough to have a decent chance of your patch being merged, Milestone 1 is the deadline for your first public action. That is, you must start the public process of submitting your contribution(s) by March 18th.

If you have previously contributed to an open source project, consider this assignment a good excuse to go contribute again. Contributions made prior to this semester don't count.

Submission checkoff:

- □ What project you picked, why
- □ What problem/feature you picked, why
- □ How did you implement/fix it?
- □ Show a history of your communication with the upstream maintainers
- □ Show that your patch was accepted – or (judgment call) that you put forth a solid effort thus far and will likely continue to put forth the effort to get this patch in