Open world Mindset

30 December, 2023

INSTRUCTOR: ME

The open market

Beginner:

How to learn specific concepts:

- If your goal is to contribute to Bitcoin Core, don't get stuck trying to learn the details of every concept. There are many topics with infinite nuance. The goal is to get a high level understanding and be exposed to different parts of the system. It takes time to internalize the fundamentals of how this distributed consensus system operates.
    - Understanding the bigger picture context is just as crucial as digging into one aspect and familiarizing yourself fully with your proposed changeset.
    - There's also implicit pressure when you interact with others to be knowledgeable about a vast number of topics to contribute to whatever conversation arises.
    - So far there are plenty for helping expose you to a meaningful niche waiting to be worked on through up-for-grabs.
- You first have to piece together a rough map of the area with snippets of understanding, when navigating big projects and then chart out an approach. I first inspect the area I want to visit on a topographic map (topo) and draw a rough path of my intended journey, trying to stay in the gentlest land. Sometimes you hit a cliff and have to backtrack and try an alternate route. Path finding requires a lot of energy and attention. You have to be a navigator and a hiker.
    - Navigating the open source realm (world) reminds me, a lot of off-trail backpacking. বারবার পথ থেকে সরে bag প্যাকিং করতেছি. With backpacking,

How to learn language details:

- I was focused on learning the most relevant parts of the language that would allow me to **read** the Bitcoin Core codebase.
- Hands-on learning works best for me like When I come across a new C++ concept, I read the docs and try to isolate the functionality in an example program.
- I sometimes had questions around punctuation that I didn't know how to google for. I more **often want to confirm** the understanding I've gleaned from docs and tinkering.
    - There are wonderful online communities with strangers willing to help point me in the right direction
    - "cpplang" slack, a ##C++-general irc channel, and #include<C++> community with a discord server.
    - I've also hopped on the #git irc channel for git specific questions.

Find an open issue to tackle (How to):

- "I would love to do this! Thanks for the guide, I'll follow it and let you know my results. 😄"
- "Here's what I have so far: #4532 Please let me know if I did everything correctly!"
- "Thanks for looking into this error. I followed your suggestions. Here's the output."
- "I checked the help docs and didn't find any mentions. I'm not sure how to implement X."
- contribution wasn't accepted, Ask the maintainer for feedback and clarification.
- contribution wasn't accepted, Don't forget to say thanks when a maintainer put effort into helping you.
- Example: I find that just opening a project and trying to "understand it" is generally a fruitless endeavor. Having a concrete goal helps massively, because instead of trying to take in everything, you can focus on tracking down one thing. find some annoying paper cut or limitation, then try to find where it comes from and if you can fix it. usually it's just a small function (well structured projects often do not have big functions), try to fix the issue in that function. It's about working on it despite not understanding a lot, this is the professional attitude. I asked for more direction (relevant questions)**.** I was able to solve the issue after getting all the relevant details

I needed. The maintainers of the project have been well-versed in the project for years. They can help you fairly quickly. The alternative is you spending hours lost in their codebase trying to figure something out that you're not even supposed to know in the first place. (not relevant for the issue)

- I've learnt and discovered so many new things by working on just three basic issues.

PR types / issue tackle types:

- String "TODO" for your level of familiarity with the codebase.
    - I came across some tests that look beginner-friendly.
- Choose a section of the codebase and pay attention to relevant PRs being opened.
    - If you follow along with the PR conversations, you will *definitely* notice follow up to-do's. Also, think critically about how to test the changes. Opening PR's to address follow ups and increase test coverage will be much appreciated.
- Added examples
- Test cases
- Capture PR follow-up ideas
- Features.
- Then I became a mentor in Google summer of code. next year.
- Automate project setup
- Improve tooling and testing
- Split the issue. your issue will likely be reviewed fast if it is more focused and small.

For becoming more familiar:

- Read the onboarding blog posts
- Attend the weekly PR review club. Reviewing PRs is crucial to the project. More relevantly, as you are getting started
- irc meeting every Thursday at 19:00 UTC in #bitcoin-core-dev

Some Facts:

- I've been very surprised to observe how much low-hanging fruit there is to contribute.

- Observing the never-ending list of (both simple, good-to-have and complex) todos has really put in perspective how young the Bitcoin project is and the blatant need for more contributors.
- Issues are like starting a conversation or discussion
- Open source is all about self exploration.
- Pull requests are for starting work on a solution
- IRC, Slack, or other chat channels, Stack Overflow is For lightweight communication.
- In open source, you get to work with people who work in big company's. There are a lot of startup people as well.