Opening up Open Source

Derek Buitenhuis

@daemon404

derek@videolan.org

NTTW4

6 December 2019

Budapest, Hungary

Who am I?

- Senior Video Engineer @ Vimeo
  - Don’t have any forks internally, very few in total. We contribute it back.
  - Contribute to many projects via Vimeo, e.g. invested in rav1e development.
- VideoLAN Non-Profit Board Member
- Contributing to FOSS since 2003 (started with X-Chat!)
- Active FFmpeg community member for many years, but less so now.
- Contributions to various FOSS projects like rav1e, L-SMASH, FFMS2, FFV1, x264, etc. as well as many Go packages and some personal projects like BXD and d2vsource.

What is This?

- Should provide a good foundation and understanding of FOSS, why it's important, and how you can get started contributing (and why you should).
- How you can meaningfully interact with open source communities.
- These are all general tips and soft rules rather than hard rules.
  - Ultimately, it's up to the community of the project.
  - Common sense applies.
- I am not an authority, just some guy who likes to send patches and cares about multimedia.

Why Should I Care About Open Source?

- Reverse Question: What legitimate (i.e. not “makes management happy”) reasons can you think of to prefer proprietary software for archival or media pipelines?
  - Probably exactly two: Support, and “because media industry specs are nasty and complex, and only this one special piece of software handles all the undocumented parts”.
  - Proprietary vendors love to spread FUD (fear, uncertainty, and doubt) about FOSS.
    - When it comes to digital media, most of them use some or a lot of FOSS under the hood.
  - Can you actually understand deeply what happens in this software? Is this good enough for future proofing? Will you be able to read or at least understand these files in 40+ years?
    - Example: Putting copies of source code and specs directly on the media in which the archive exists.
  - How many eyes have seen the code? Is it enough to vouch for its quality?

Why Should I Care About Open Source? (Cont'd)

- Have you ever had a problem with some proprietary where you spend days trying to fix it only to talk to someone a year later and find out they suffered the same fate? Open bug trackers matter.
- On support: Many projects have paid support options (VideoLabs, MediaArea) or developer are available for paid work.
- You may not even need paid support (other than to make management happy).
  - Many community members and developers want to make the software better, and polite feature requests or file bugs can be sufficient.
  - Maybe **you** can do it! It's probably less difficult than you think, and most projects welcome new help with open arms, and can mentor you. It all comes down to time.
  - Thought: The cost of an employee's salary vs any given support contract or software license.
- None of this is to say proprietary software doesn't have its place and value.

OK, I Use Open Source. But Why Contribute?

- It’s better for your time: Getting to know the community and its documents or code, for a project that you use every day, or at least often, will save you time and effort in figuring how do do things, how fix things, etc.
- Less burden: Any internal patches or forks may seem fine at the time, but their maintenance burden increases all the time. Someone besides you can have a chance to understand it.
- It benefits your ecosystem as a whole: More standard implementations, less diverging, better interop.
- Influence: It lets you have a say on how the projects you use are developed and work.
- It helps others!
- It’s fun!

Getting Started

- Code is not the only way to contribute to FOSS.
  - Documentation help is **always** welcome, be it technical documentation, user guides, wiki articles, comments, etc.
  - Bug tracker help, forums, replies, user support in general (e.g. #ffmpeg vs #ffmpeg-devel).
  - Supply test files that can reproduce issues or test certain features – it's harder to find these than you think!
- Code:
  - Have a pet issue? Try to fix it and submit!
  - Check the bug tracker, wiki, or IRC/Slack for good starting points.
  - Maybe you just wanna do something for fun!

Before You Contribute

- Read any rules available! This includes: Any code of conduct, IRC or mailing list rules, code style, patch submission, developer documentation, wiki etiquette, documentation rules, CLAs, etc.
- Learn any required tools (usually simply): Mailing lists, git send-email / format-patch, GitHub, etc.
  - Please don’t insist on e.g. using GitHub PRs when a project uses a mailing list for patches.
- Make sure you communicate using the right channel.
  - The correct mailing list (-dev vs -user) or IRC/Slack channel.
  - Don’t email developers directly/privately without knowing they’re OK with this.
- Most of this is common sense, but you would be surprised how many potential new contributors do not. You will start off on a good note if you do!
- Don’t be *That User*™ who complains about everything and how they would do it better, but never does.
- These are just guidelines; every project is different.

Humans Gonna Human

- Be polite, genuine, and if necessary, technical.
  - Well thought out and concise prose is a huge benefit for effective communication.
  - Contribute meaningfully, don’t just add noise like ‘+1’ or ‘ping’ every thread.
- Asking questions is generally encouraged, but try to find an answer in e.g. documentation first.
  - More often than not, people are glad to help.
- Do not act entitled; nobody owes your contributions or opinion extra attention over others.
- Minimize the barrier to adoption:
  - Fully test (as much as feasible) your contributions.
  - Provide well written rationale for your changes.
  - Provide any necessary context, background, etc.
- Be open to polite critical feedback on your contributions, technical or otherwise – quality benefits everyone in the long run.

Humans Gonna Human 2: A Wild Jerk Appeared

- Try not to take it personally, at least at first.
  - Everyone is human and has bad days, but consistent poor behavior is a different beast.
  - If it **is** consistent, it is worth raising it up to the community at large to try and rectify it.
    - There should be a proper channel for this, usually.
    - Sadly, not every community has a proper enforceable Code of Conduct.
  - Make sure you're not conflating technical criticism with jerkiness.
- Do **NOT** hit reply immediately and type out a 10 page retort. An upset in-the-moment reply generally begets more of the same, and flame wars benefit nobody.
- Remain polite, explain your stance, and mention they've acted unacceptably.
- If they've noted any legitimate issues (while being a jerk): Address and/or fix them.
- But remember: This is **your** time and energy, and you should spend it where you enjoy. You are not required to have "thick skin" to contribute, or at least, you should not be.

DIY

- Established large projects are not the only way to contribute to open source.
- Why not create your own repo / project?
  - Maybe you have useful scripts or tools that can benefit others.
  - Create a healthier ecosystem in your niche or industry.
- You are the one responsible setting contribution guidelines and the code of conduct.
  - A.K.A. No jerks unless accepted (unless you're a jerk, of course).
- Can make your internal tools more useful and robust via outside contributors.
- Your tool is not too niche to be useful to others! A lot of proprietary software exists to cover these niches because nobody has written anything open.
- Of course this depends heavily on what your job is a "secret sauce" is.
  - I don't see any reason archivists should shy away from sharing for mutual benefit.
- Maybe you just want to do it for fun.

A Good Example: FFV1

- Has an open specification, and multiple tested implementations.
  - No licensing fees, NDAs, or paywalled specifications.
- Has paid support available via MediaArea.
  - Extra specific features like niche ingest color spaces and pixel formats.
- Open and polite mailing list and GitHub repositories, with accessible developers.
  - Open spec meant more can easily implement it, but more importantly, improve upon existing implementations and the spec itself. More eyes means more robust.
- Is “future proof” in that it is both fully specified and implemented with licenses permissible enough to allow embedding in media and understanding by future archivists.
  - Adoption is reflecting this.
- Documented way to contribute and code of conduct.

Another Good Example: rav1e

- rav1e is an open source AV1 encoder written in Rust
- A good mix of paid developers (Mozilla, Vimeo) and hobbyists
- Very inclusive community
  - Very willing to spend time with new contributors to help them learn, be it Rust, difficult encoder concepts like optimal RDO.
  - Big on community outreach (conferences, news, Reddit, events like IBC)
  - Weekly open-to-all voice chat meetings on its roadmap
- More concerned with the long term viability and adoption AV1 ecosystem and community than code dumping hacks to get a kind of good encoder right away.
- Documented developer guidelines and tools to enforce these.

Conclusion

- You should care about Open Source, especially as archivists.
  - More eyes = more robust.
  - Long term viability.
  - Healthy ecosystems matter.
- Contributing isn’t that scary!
  - There are plenty of ways to contribute.
  - Documentation (for developers and for users) is important.
- Everyone is human, even the jerks.

Questions?