The Maintainer's Paradox

One of the fundamental principles of Open Source, I learned years ago from Eric Raymond's seminal work "The Cathedral and the Bazaar". [Cathedral] It is what he designated as "Linus' Law".

In the paper, it was expressed as a statement about Open Source code quality: "With enough eyeballs, all bugs are shallow." Put in different terms: If more people are looking at a problem, then there is a higher probability that someone will have the needed skill to identify the problem easily.

This seems obvious, and it applies not just to identifying bugs, but also to proposing new ideas for a project. Especially in the area of ideas, it is not just numbers that count, but diversity. Diversity is important. Diversity of usage. Diversity of experience. Diversity of thought. Diversity is the lifeblood of Open Source projects. However, it also has its downsides. Different ideas take more time to understand, to deal with, to shape and to integrate.

In the Open Source training I do at Sony, I represent Linus' Law with light bulbs. [Light Bulbs] If you have 5 or 10 light bulbs, then you will have some good ideas. But if you have thousands of light bulbs, then fairly regularly you will have some great ideas that can be incorporated into a project. It's a matter of probabilities. The role of a maintainer is to identify those great ideas, cultivate them, and ultimately incorporate them into the project.

Over the last few years. I have become the maintainer of the Fuego test system. [Fuego] And I have learned a few things. Over my 25-year career in Open Source, I had always been a user and a contributor, and I have sometimes groused about how this or that maintainer had not done things I had wanted them to.

[Architect] But being the maintainer of an Open Source project has given me a new perspective. I only have a limited percentage of my time devoted by Sony to working on Fuego. Compared to the Linux kernel, Fuego has a low bandwidth of patches that get contributed. But even so I sometimes find myself overwhelmed. Sometimes when I see a patch contribution, my first reaction is "oh no – I don't have time to look at another patch set today".

(As an aside, I need to be careful here, because there are Fuego contributors in the audience, and I don't want them to think I don't want their contributions.)

Despite my initial negative reaction, I find that many times when I look at a patch set, I see ideas or approaches that I hadn't considered before. I learn more ways to do things. I become a more experienced and skilled engineer.

The maintainer's paradox is this: An Open Source maintainer actually makes more work for themselves every time they take a patch from a contributor. [Puppy] Each patch is like a puppy that must be cared for indefinitely. Sometimes puppies are cute, welcome gifts. But they need care and feeding. The job of a maintainer expands automatically, inexorably, and it's harder than I thought.

So maintainers have a built-in incentive to ignore or reject patches. But at the same time, we WANT new patches coming in. We're happy to see new contributors, fresh ideas, and especially fixes to longstanding problems.

The interesting thing about being a maintainer is that it consists of more than just technical work. There's a balancing act, to keep contributors engaged and feeling good about their contribution, while also correcting it, asking for changes, or outright rejecting it. Due to time constraints, answers can sometimes be short and unclear. [Figure group] There's a social element to interacting with people on the mailing list. Despite the aspirations to have code review and patch acceptance be based purely on meritocracy, there are social aspects of interactions and communication problems. There are misunderstandings, frustrated responses, and sometimes flippant words that hurt feelings.

[warning signs] Recently, at another conference, there was a talk delivered by an experienced kernel maintainer expressing observations of negative behavior within the Linux kernel community. I don't agree with everything that the person said, and I don't believe the problem as he stated it is as bad as he has characterized it. But there were a lot of details omitted from his presentation so there may be incidents I don't know about. One can argue about whether the Linux community is more like the environment on the left or the right. In any event, I believe there is always room for improvement.

In the Technical Advisory Board of the Linux Foundation, we have been recently having discussions about ways to improve positive communication within the Linux community. Sometime in the next few months I hope we can publish and discuss, and refine together, some new guidelines and tips to help both contributors and maintainers avoid negative communication.

In the meantime, I'm going to suggest a few things you can do to make things better. The first two are for when you see what you consider to be inappropriate communication in the Linux community.

1) Call out the behavior, privately or publicly

This can be difficult when you are the target of negative comments. It is helpful to do this as an impartial 3rd party. Doing this in public opens the possibility to discuss acceptable standards of communication in the community.

2) Route around the offender

For the past few years, there has been a push within the Linux community to utilize maintainer groups. This means that subsystems have more than one maintainer, who can approve patches. Use a different member of the maintainer group, or send your patch to Andrew Morton, if you are having a hard time working with a particular maintainer.

There are also things you can do to make your maintainer's life easier, to relieve their stress, and to make them more willing to take your patches:

3) Listen carefully to, actively clarify, and act on their feedback

A lot of frustration comes from misunderstandings on the mailing list, due to curt answers or missed instructions.

4) Assist the maintainer, by reviewing patches or providing feedback to others on their patches or ideas.

This is something that relieves me, as the Fuego maintainer. Each response I compose takes time, and I really appreciate it when someone else answers the question for me. I particularly like it when someone answers it correctly.

5) The most powerful thing you can do is to become a maintainer yourself. Join one of the maintainer groups in the Linux kernel.

Now, having said all that, let me just add one caveat. Not everyone is cut out to be a kernel maintainer. The kernel is now used in billions of devices, in thousands of different industries and applications. Keeping it running smoothly, without regressions in performance or quality, is very difficult. There are areas of the kernel where it is really, really difficult to contribute something beneficial. That's not to say it's impossible, but it can be very hard. My last bit of advice is this, and I think it's the most important:

[at work sign]

6) Find someplace to contribute and just do it.

Every open source project has areas to contribute to, which require different levels of technical expertise. Start with a bugfix, or a device driver in staging, before trying to tackle changes to the kernel memory system or the scheduler. But that's just code. There are other ways to contribute. Help with documentation. Reproduce and triage bugs. The possibilities are endless. I have a whole list of things on the Fuego project that you can help with, if you are interested in contributing. That's how you get started.

[rainbow] Working on Linux has been one of the most rewarding things I have done in my professional life. Let's keep working together to improve our communication and our environment, our software and our projects, and our contribution to society at large.

Thanks.