Zack's Kernel News

Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

ZACK BROWN

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

Defensive Patch Tracking for All

Luis R. Rodriguez pointed out that the "Signed-off-by" tag has become popular with other free software projects.

Originally, the "Signed-off-by" tag for patch submissions was created in response to the SCO lawsuit, which targeted (among others) Linus Torvalds and asked for proof that Linux did not incorporate code derived from Unix System V, which SCO owned. At the time, given the hierarchical nature of Linux development, patches would sift upward through mailing lists, testers and cohorts, official maintainers, lieutenants, and others before finally arriving in Linus's inbox to be applied to the kernel tree. Not much more than good will ensured that the code submitted to Linus was actually owned by the person who originally submitted it.

Ultimately, Linus has an algorithmic approach to development. Kernel development is itself a running process on a very strange system. The copyright challenge represented out-of-memory errors and data throughput bottlenecks and perhaps came close to crashing the system entirely. In other words, it required a god-awful amount of work to refute SCO's claims.

The Signed-off-by tag is Linus's algorithmic answer to that whole issue. Actually the Signed-off-by tag is only the most visible part of what has become an official "Developer's Certificate of Origin" (DCO) process. The DCO is kept in Documentation/SubmittingPatches and has its own version number (currently at version 1.1). According to the DCO-1.1, whenever a patch enters the development system (i.e., someone submits it), everyone who reviews it adds the "Signed-off-by" text identifying themselves as one of the people in the review chain and certifying that they have not added any code that would violate the terms of the GPL.

Thanks to the DCO-1.1, in the future, any piece of code identified as a possible copyright violation in a court case will have a relatively easy path to identify the specific people involved in submitting that code, so they can give an accounting of how they came to submit that code. Thus, the task of clarifying the kernel's origins can be efficiently distributed to the relevant contributors, instead of being heaped on Linus alone. (Other people volunteered to take on much of that burden, but that's a different story.)

Luis, in his recent email, pointed out that the Signed-off-by tag has become popular with other open source projects, but, he said not all of them had a clear understanding of the full specification of the DCO-1.1. Linux used it, so they loved it, but without necessarily grokking how to eke out its best value.

Luis suggested that the DCO-1.1 should be extracted from Documentation/SubmittingPatches and given its own separate standalone project, so other free software projects would have an easy way to find and refer to it.

A few kernel folks considered his suggestion. Alan Cox thought it would be best to leave the document in the kernel tree proper, but that Luis could cut-and-paste it elsewhere, if that would be useful to anyone. He pointed out, "There's a reason that lawyers copy documents into other documents rather than doing late dynamic binding – you want to be sure that what you reference is the *exact* text that is valid for this case."

Jiri Slaby agreed that copying the text to other open source projects would be better than creating a distinct project just for it, or even linking to it in the kernel sources. He suggested that other projects might want to modify the terms of the DCO, and so they'd want to have their own copy of the explanatory text and change it according to their needs. Alan agreed with this. Essentially, any free software project might want to "fork" the DCO, and that should be fine.

At this point, W. Trevor King said that he was having trouble identifying the license under which the DCO-1.1 had been released. He linked to Linus's very interesting 2004 request for discussion [1], and he pointed out that Linus had himself created the initial patches that added the DCO into the Documentation/SubmittingPatches file. Trevor also linked to the Open Source Development Labs' (OSDL) press release announcing the DCO-1.1 [2] and specifically the part where OSDL said, "© 2005 Open Source Development Labs, Inc. The Developer's Certificate of Origin 1.1 is licensed under a Creative Commons Attribution-ShareAlike 2.5 License. If you modify you must use a name or title distinguishable from 'Developer's Certificate of Origin' or 'DCO' or any confusingly similar name."

Trevor asked, "What license is the DCO distributed under and who holds copyright?" There was no answer to this on the linux-kernel mailing list, but in a later post, Trevor announced that he'd created a new Git repository preserving the commit history of the DCO [3]. Because he derived this new repository from the Linux kernel and the Git project, both of which were licensed under the GPL version 2, he released his own repository under the GPL version 2 as well. Trevor said, "If you're using a GPLv2 exact project, you can merge the 'signed-off-by' branch into your project directly."

But, he added, "Because many projects that are not GPLv2 may still want to use the DCO/s-o-b approach, I've included an example CONTRIBUTING file (and CONTRIBUTING.md for GitHub) that are licensed under the very permissive Creative Commons CC0 1.0 Universal. Merge the 'contributing' or 'contributing-github' branch into your project and edit as you see fit."

Luis was ecstatic about this and immediately began incorporating Trevor's work into his own free software projects.

Project Inclusion Criteria

There was a bit of a kvmtool kerfluffle recently when David Rientjes asked if kvmtool would ever be migrated from `linux-next` into the kernel proper.

KVM (Kernel Virtual Machine) is a kernel subsystem that allows users to create and run virtualized systems on a running Linux system. It's great! And kvmtool is the userspace tool that actually boots the KVM guest images so users can use them.

In response to David's question, Stephen Rothwell said that Linus Torvalds didn't want to include kvmtool in the kernel, so Stephen was going to remove it from the `-next` tree; he asked Ingo Molnár to remove it from his `tip/auto-latest` tree as well.

However, Ingo said he planned to keep kvmtool in his tree because he used it for testing and hadn't encountered any problems with it and because Pekka Enberg was planning to submit a new version to Linus in the near future, which might be accepted.

H. Peter Anvin replied, "So why don't we let Linus either accept and reject it for the 3.9 merge, but if rejected, we drop it from `linux-next` until such time as Linus' objections have been addressed?"

Stephen didn't think there was much chance that Pekka's patch would be accepted, and he pointed out that he didn't want Ingo to remove kvmtool from his own tree, only from that part of the tree that resided in `linux-next`. He quoted Linus as saying, "I have yet to see a compelling argument for merging it. It's tons of code, it doesn't match the original 'small simple' model, and I think it would be better off as a separate project."

Ingo replied that the `-next` version of his tree was identical to his working tree, and he didn't want them to diverge, although he did say he'd change it if Linus really insisted. However, he added that kvmtool was improving nicely, had several dozen contributors, and was very useful to kernel development. It aided KVM development and was also used to test experimental kernel features without having to reboot the entire system.

Revealing his frustration with the debate, Ingo continued, "What harm has tools/kvm/ done to you?" And, "*Please* don't try to harm useful code just for the heck of it." He concluded, "Please stop this silliness, IMO it's not constructive at all."

Linus joined the discussion at this point, drawing a hard line. He said that he hadn't seen any good reasons why kvmtool shouldn't just remain a standalone project. He pointed out that Ingo's claims that kvmtool was useful to kernel development were true enough, but that this wasn't a good reason to merge something into the kernel.

He also pointed out that tying kvmtool to the kernel only made it more difficult for users to access. Instead of just going and getting kvmtool, users had

to go and get the entire kernel source tree, with kvmtool inside.

Linus concluded, "let me state it very very clearly: I will not be merging kvmtool. It's not about 'useful code'. It's not about the project keeping to improve. Both of those would seem to be *better* outside the kernel, where there isn't that artificial and actually harmful tie-in. In other words, I don't see *any* advantage to merging kvmtool. I think merging it would be an active mistake, and would just tie two projects together that just shouldn't be tied together."

Pekka replied to this, saying, "you are absolutely correct that living in the kernel tree is suboptimal for the casual user. However, it's a trade-off to make tools/kvm *development* easier especially when you need to touch both kernel and userspace code." He pointed out that "we support KVM on ARMv8 even before the in-kernel code has hit mainline. People implemented vhost drivers in lock-step. Most of the contributors are also kernel developers. And we in fact have a clean codebase that's accessible to anyone who knows the kernel coding style."

Linus replied that these things didn't qualify as justifications for merging the code into the kernel versus keeping kvmtool as a standalone project. The convenience of the developers was simply not a reason to merge code, he said. He added, "The only thing the lock-step does is to generate the kind of dependency that I ABSOLUTELY DETEST, where one version of kvmtools goes along with one version of the kernel. That's a huge disadvantage (and we've actually seen signs of that in the perf tool too, where old versions of the tools have been broken, because the people working on them have been way too much in lock-step with the kernel it is used on)."

Linus asked what was standing in the way of kvmtool just being a standalone project. Pekka replied that the kvmtool code was too tightly dependent on the kernel to be taken out. If it were extracted into a separate project, it wouldn't even build correctly. Pekka also said that in terms of their development environment, the mailing list and the kernel development workflow all represented infrastructure that they would have to create afresh if kvmtool became a separate project.

But, Linus replied, "You do realize that none of your arguments touched the 'why should Linus merge the tree' question at all? Everything you said was about how it's more convenient for you and Ingo, not at all about why it should be better for anybody else."

He added, "You haven't bothered to even try making it an external project, so it doesn't compile that way. You're the only one working on it, so being convenient for you is the primary issue. Arguments like that actively make me not want to merge it, because they are only arguments for you continuing to work the way you have, not arguments for why the project would make sense to merge into the main kernel repository."

Linus also said that he didn't mind if Pekka and Ingo continued working the way they had been. They didn't need to disrupt their workflow or development process at all, but he just wouldn't merge the code. "There's no reason why kvmtool couldn't be external the way all the other virtualization projects are," Linus said.

They went back and forth on the same issue for a bit. Pekka's argument was essentially that the developers could work faster and better if the code were in the kernel, and Linus's argument was that code only goes inside the official tree if it belongs there for technical reasons, rather than the convenience of developers.

At one point, Theodore Ts'o joined in the discussion, saying: "I completely agree with Linus here; in fact, the main reason why it's important to keep userspace tools outside of the kernel is that it keeps us careful about interface design. For example, I consider it a *feature* that when we extend the file system data structures for ext4, they have to be made in the two places; once in the kernel, and once in e2fsprogs's version of ext2_fs.h. Yes, it might be more convenient if we pushed all of e2fsprogs into the kernel, so I wouldn't have to edit ext2_fs.h in two places, but when I make changes to ext2_fs.h, I want to be really careful, lest I not break backwards compatibility, and to think very carefully about forward compatibility issues. If there are constantly huge numbers of interface changes in the kernel/userspace interface, then it increases the chances that mistakes will be made. It's better that those mistakes be caught early, and if changes need to be made in two places, it increases the chances that these mistakes will be noticed sooner rather than later."

David Woodhouse agreed, adding: "If you want to use pieces of the kernel infrastructure, then just *take* them. There are loads of projects which use the kernel config tools, for example. There's no need to be *in* the kernel repo. And for code-reuse it's even easy enough to automatically extract parts of kernel code into a separate repository. See the ecos-jffs2 and linux-headers trees, for example, which automatically tracked Linus' tree with a certain transformation to make them sane for just pulling into the relevant target repositories."

The debate continued and got more and more heated. Ingo and Pekka continued to insist that there were real, measurable benefits to including kvmtool in the main tree, while Linus continued to insist that the benefits they identified were all centered around the convenience of the developers and didn't meet his requirement of having a technical justification.

Ultimately, Ingo said in the final post to the thread of discussion: "So, just to bring this to a conclusion, obviously Linus is insisting on it, so I've removed tools/kvm from tip:auto-latest, by going back from the daily merges (where tip:master was == tip:next) to the older complete reintegration merges to linux-next every couple of weeks. This way tools/kvm will still be available in tip:master (merged after full integrations) and there are still the usual daily (or more frequent) delta-merges of tip:master as new bits get ready – with the occasional riskier total reintegration done for linux-next."

Ingo continued, "It's obviously not optimal, but that's the best I could come up with given the constraints." ■■■

INFO

[1] DCO request for discussion: *http://article.gmane.org/gmane.linux.kernel/205867*

[2] OSDL DCO press release: *http://web.archive.org/web/20070306195036/http://osdlab.org/newsroom/press_releases/2004/2004_05_24_dco.html*

[3] DCO Git repository: *https://github.com/wking/signed-off-by*