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**.

Global Hash Table Implementation

What can happen in big software projects is that developers implement the same low-level feature over and over again. Some sort of helper code would be useful, so they create it in their little area of the project and use it happily, never realizing that there are similar helpers scattered all over the codebase.

One such useful gizmo is the hash table. Sasha Levin recently submitted some code for a simple hash table implementation, specifically so it wouldn't need to be reimplemented all over the place by other kernel hackers.

The whole point of a hash table is that it's fast. You put in a key, and you get out the value, quick as a wink, and your code can continue on its merry way. This is especially important when the hash table is in the operating system kernel, because if the kernel is slow, it can make all the user programs slow as well. So, among other optimizations, Sasha's code used macros wherever possible, to shunt as much actual work as possible onto the compiler. Things like the size of the hash key, for example, can be calculated at compile time and not take up any cycles in the running kernel.

One problem with using macros, however, is that expressions can be passed around in an expanded form before actually being calculated; so, something like `3+n/2` could be fed into a macro as a single input. But, instead of having a simple variable name like `bits`, it would be passed as the actual expression `3+n/2`. So, if the macro defines an operation to be done to the input, such as `2/bits`, that would be passed in expanded form as `2/3+n/2` instead of `2/(3+n/2)`, thus potentially producing a subtle operator precedence bug.

Sasha's code addressed this by requiring users to put parentheses around macro variables such as `(bits)`. But parentheses around a single variable tend to look superfluous and could be forgotten by users. In fact, Mathieu Desnoyers pointed out a place where Sasha himself had forgotten the parentheses in the code implementing the hash table.

Matthieu also offered other fixes and suggestions, as did Tejun Heo. It does seem as though, modulo these caveats, that Sasha's hash table implementation will ultimately result in a cleaner kernel for all.

Animation of Kernel Development History

Darrick J. Wong recently posted a YouTube video [1] he'd made of a gource animation of the past 21 years of Linux kernel development.

The GPL3ed gource tool is not new, but it's still lovely to watch kernel development play out over nearly three hours as an animated adventure story. Watching the sudden blooms and contractions of code, as little avatars scurry to and fro with their laser beams of hard labor, is fascinating and mysterious. It's like something I'd expect to see from Hayao Miyazaki.

Whenever I see a visualization tool like gource, I always imagine ways it might help me do my own kernel research. In this case, gource can take a Git tree, focus on a particular branch of development and a particular developer, isolate a particular time frame, and play it at variable-speed resolutions. In fact, looking at the gource man page, it seems to have been designed specifically to be useful to anyone doing research into a given project.

Possible GPL Violation – And Possibly Not

Accusations of GPL violations can get pretty sticky. The accusation itself can end up as a libel suit. This often means that questions about potential violations are handled privately, if possible, and only brought to public attention if the organization refuses to respond, or if it seems quite clear that a violation has indeed occurred.

Andy Grover recently said that Rising Tide Systems, in distributing their Linux-based RTS operating system, included non-GPLed code in their distribution; specifically SCSI-related code that would make their OS more VMware compliant.

The official SCSI subsystem maintainer, Nicholas A. Bellinger, also worked for RTS, and Andy had tried emailing him privately, along with the RTS CEO Marc Fleischmann, but he said he hadn't gotten back any useful responses, so he posted publicly to the list. This resulted in a big brouhaha.

Nicholas did reply on the list and said something very interesting. He said that RTS had contributed certain code to the official Linux kernel and that this code was indeed

released under the GPL. However, he also said that because RTS had produced the code and was therefore the copyright holder, it was free to release the code under a separate, proprietary license if it so chose. And, he said that in fact, this is what RTS had done – they had forked their own codebase and maintained separate GPLed and proprietary branches. The GPLed branch was intended for submission into the official kernel tree, whereas the proprietary branch was intended for their professional RTS OS releases.

So, Nicholas's argument was apparently that, as copyright holders, they could release their code under the GPL and also under a proprietary license, and that including their proprietary code in their own Linux-based OS; therefore, it was not a violation of the GPL.

Andy replied to this, saying that the violation was actually somewhat different from what Nicholas described. The accusation was not that GPLed code was being included in a proprietary version of the kernel. The problem was that a proprietary version of the kernel couldn't legally exist. By including their code in a Linux-based OS, it became a derived work, which, under the terms of the GPL, meant that it too had to be released under the terms of the GPL or not at all.

In reply to this, Nicholas said another very interesting thing. Among other things, he said that RTS only used kernel symbols that were not marked *GPL*. In other words, he implied that third-party code could be embedded into the kernel, but if it didn't used symbols marked *GPL*, then it could be released under a proprietary license.

But, as Alan Cox pointed out, the kernel's symbol tags didn't define the legal barrier between GPLed code and non-GPLed code in the kernel. They were just a guideline for third-party code to follow. And, third-party code was still bound by the terms of the GPL if it were to be included in its own version of the Linux kernel. The existence of symbol tags in the kernel didn't release anyone from compliance with the license.

Alan said, "So either your work is truly not derivative of the kernel (which I find wildly improbable) or you have a problem and since you are aware of the complaints publically I guess probably a triple damages sized problem."

Alan also introduced a new wrinkle, saying, "there are US patent grants for some functions in the kernel that only extend to GPL code so utilising some of the subsystems in the USA may give you other problems even if you can somehow manage to demonstrate your work is not derivative."

At around this point, Bradley M. Kuhn, Executive Director of the Software Freedom Conservancy, joined the discussion. He responded to a point Nicholas had made elsewhere, that the proprietary RTS code had been certified by Black Duck Software. Black Duck, among other things, offers tools that scan code for open source software and report potential violations.

Bradley replied to Nicholas's claim of Black Duck certification, saying, "Often in my work enforcing the GPL, companies have unsuccessfully proposed a Blackduck review as a defense of copyright infringement. I don't think BlackDuck's system does anything to determine whether or not something is a derivative work under copyright law and/or whether a violation of GPL has occurred."

He added, "citing a Blackduck certification is simply off-point in refuting an accusation of any form of copyright infringement. Blackduck's software might be able to tell you if you *have* plagiarized someone's source code that appears in their databases, but they can't possibly tell you that you haven't infringed any copyrights. I'm quite sure Blackduck doesn't give away certification on the latter point."

Nicholas had also said early on in the discussion, and specifically to Andy's employer Red Hat, "We're very disappointed that Red Hat

would not be more professional in its communications about licensing compliance matters, particularly to a company like ours that has been a major contributor to Linux and therefore also to Red Hat's own products. So, while I invite you to talk about this with us directly, I also advise you – respectfully – not to make public accusations that are not true. That is harmful to our reputation – and candidly, it doesn't reflect well on you or your company."

To this, Bradley replied, "While I usually encourage private discussion about GPL violations – at least to start – I've also often found it's nearly impossible to maintain perfect secrecy about alleged GPL violations; openness and public discussions are the standard manner of group communication in the Free Software community. I hope that Rising Tide Systems and its agents are cognizant of this nature of the Free Software community. Furthermore, now that the discussion is public anyway, I hope Rising Tide Systems and its agents will welcome it and avoid any further actions to squelch such discussion."

He added, "I don't think it's reasonable to chastise Andy for raising these questions. While I personally (and Conservancy as an organization) don't usually raise accusations of GPL violations publicly until other methods of private communication are attempted, I believe public discussion is an important component of GPL compliance. Thus, Andy's strategy of discussing it publicly early in the process – while not my preferred strategy – is still a reasonable one. His attempt to raise these serious and legitimate concerns and questions is in no way unprofessional, nor should it be squelched."

Other folks, like James Bottomley and Theodore Ts'o, came down on the other side of the issue, urging Andy and others to avoid making accusations that could only really be resolved by lawyers. Theodore added, "The bottom line is that copyright licensing can get *complicated* and so before you start flinging about accusations, one would be wise to be 100% sure of the facts. You need to make sure that they have distributed lines of code which came from the *Linux* kernel, and not just from code which they may have originally contributed to the Linux kernel."

At around this point, Lawrence Rosen, a lawyer advising RTS in this matter, joined the discussion. He said that RTS "distributes versions of its scsi target code that support features and functions not officially in Linux (or at least, not yet). That commercial RTS business includes the licensing of those derivative works of its own code to its own customers."

He also added, "I hope that we can tone down the arguments about whether the use of Linux APIs and headers automatically turns a program into a derivative work of Linux. I think that argument has been largely debunked in the U.S. in the recent decision in Oracle v. Google, and in Europe in SAS v. World Programming. Does anyone here question whether the original work that RTS contributed to Linux (and that *is* under the GPL) is an original work of authorship of RTS despite the fact that it links to other GPL code using headers and APIs?"

Dave Airlie said, "The whole in-kernel version under the GPL isn't the problem here and has nothing to do with the violation. Its the one they distribute separately with their RTS OS kernel, and if they distribute it in one combined work, then a GPL violation is possibly indicated."

By this point, Andy, who started the whole discussion, decided to continue it in private. At the time the discussion ended, most participants seemed to agree that no violation of the GPL had been clearly established and that more information was needed. Bradley did say he disagreed with Lawrence's interpretation of the Oracle v. Google and Europe in SAS v. World Programming cases. But that discussion did not continue.

There's a tremendous amount at stake in the legal interpretation of the GNU General Public License, version 2. The whole nature of "copyleft," and the idea of requiring derived works to be released under the same open source license, is one that many contributors feel is crucial to the success of free software.

At the same time, there's a lot of uncertainty about how the GPL should be interpreted, and even whether the long tradition of proprietary device drivers is actually a violation of the license. There are questions about whether the ways a copyright holder chooses to enforce their license affects the way the license should be interpreted by the courts. If the license holder chooses not to pursue violations of a given provision of the license, does that establish a precedent that protects those users from future legal action?

And, amid those debates are fears that saying too much in public might result in lawsuits for libel; so, people typically have become reluctant to express their opinions. The above discussion is one of the few in recent years that has had a substantial amount of discussion associated with it. Most such debates begin and end with someone saying something along the lines of, "we're not lawyers, and it's up to the lawyers to figure this stuff out."

The GPL version 2 is a fascinating and readable license [2]. If you haven't read it lately, you should.

KVM Maintainership

Avi Kivity recently decided to stop maintaining KVM (the Kernel Virtual Machine) after six and a half years. He added his name to the CREDITS file, removed his entry from the MAINTAINERS file, and left Marcelo Tosatti, momentarily, as the sole KVM maintainer.

Shortly thereafter, he invited Gleb Natapov to take on the role of co-maintainer, and Gleb sent in a patch adding himself to the MAINTAINERS file. Avi accepted the patch and pushed it out to Linus Torvalds.

KVM has ultimately revolutionized the way many people use Linux. The ability to create and destroy instances of Linux and other operating systems under a running Linux system has offered new ways to test products, new ways to implement security, and new ways to just poke around and explore systems without having to shell out big bucks for new hardware. There's a long history of system emulation under Linux, including things like Dosemu and Wine, not to mention VMware. But KVM offers a fully integrated, first-class feature of the Linux kernel.

Thanks for all the great work, Avi! And, Gleb, I hope you have lots of fun figuring out cool new directions for KVM. ■■■

INFO

[1] Gource animation of Linux kernel development: *http://www.youtube.com/watch?v=pOSqctHH9vY*

[2] GPL version 2: *http://www.gnu.org/licenses/gpl-2.0.html*