Zack's Kernel News

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

By Zack Brown

Bye-Bye BKL

After many painful years, Arnd Bergmann posted the patch that many kernel developers have been waiting for: the final removal of the Big Kernel Lock (BKL). He listed a big pile of people who had participated in the effort. Ingo Molnar shouted out, "Yay!" And Thomas Gleixner added, "Thanks a lot to everyone involved!"

Alan Cox said wistfully, "Nice to see it gone – it seemed such a good idea in Linux 1.3." To which Ingo replied:

"No need to feel bad about it – there was simply no other way to do it: the BKL basically represented all the single-CPU assumptions that were built into the kernel from 0.10 up to 1.3 (and at least as much new BKL depending code going forward as well…)."

Ingo also said, "every last such piece of code had to be eliminated. So the BKL represented the status quo – and eliminating the status quo is always hard, as the code that remains became less and less important :-)"

This is definitely a champagne moment in Linux history. A huge amount of work by a huge number of people over a huge quantity of time had to go into removing the BKL; it might not be as flashy as some of the fancy features going into the kernel these days, but look at it this way – if you consider the sheer number of man-hours that has gone into removing the BKL, it represents a massive amount of time that can now be turned to who-knows-what other magnificent projects.

New Static Analysis Tool

Reinhard Tartler announced that he and a bunch of other folks (the Vamos team) had created a new tool called "undertaker," available at *http://vamos.informatik.uni-erlangen.de/trac/undertaker*, that does static code analysis on kernel sources, identifying potential bugs without actually compiling or running the code.

This work was greeted with general praise as folks downloaded and tried it out. One of the interesting goals of the project is that the Vamos team prefers to avoid giving false bug reports, even if that means undertaker will miss out on reporting on actual problems. This way, if undertaker does report something, you can be fairly confident it's identified as an actual bug, and you can start writing your patch accordingly.

Status of Nexus One in the Kernel

An interesting eruption occurred recently surrounding Google's port of Linux to the Nexus One phone device.

Apparently, just about everyone would like to see the code get into the official kernel tree, but the Nexus One code base deviates sufficiently from the official tree that a straightforward merge would be problematic. The Linux developers want to see a clean set of patches that play nice with the rest of the code.

Daniel Walker recently inadvertently unleashed a firestorm when he submitted a pile of patches that was a first attempt at this task. He took Google's code, massaged it until he could get something that would boot (albeit requiring a serial connection to see any output), and then submitted the patches to the Linux folks.

Unfortunately, several things went boom all at once. For starters, Daniel neglected to give proper attribution to the original Google employees who authored the port. Instead, he listed himself as the author, which brought a big heap of hurt down on him from various kernel developers, claiming he was taking credit for the work of others.

As the quote-unquote discussion progressed, it came out that Daniel didn't actually have access to the identities of the original authors because of the way Google's code base was produced. So, Daniel had just been giving credit to Google in the text of the commit messages, while at the same time keeping his own name in the author field to indicate the work he had done massaging all the patches so that they would fit into the official kernel tree in a friendly way.

This misunderstanding led to many bitter words before it was cleared up; partly, this was because Daniel was not being totally clear on the purpose and true significance of the various fields in the commit messages. At one point, Steven Rostedt clarified:

"You stated that the copyright was not yours but Google's. You are not employed by Google, are you? The major problem I have with these patches is that you got code from somewhere else but had no Signed-off-bys from anyone. This is where legal comes in."

Steven went on to say: "How do you know this code was attained legally? Can you take

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**.

sole responsibility that the code was not stolen from non GPL code? The only tag line in a change log that matters is that Signed-off-by. Its the one with (sorta) legal powers. This is saying that you verify that this code was given to you through legal means. Either that you wrote the code yourself and are not under any contract to keep it from becoming GPL, or you took it from someone that gave you their own Signed-off-by that you can trust."

Shortly afterwards, Dima Zavin of Google, who'd been working on the Nexus One code base, said Google was very pleased that Daniel had volunteered to port this code back to the main kernel tree but that they did want proper attribution to be maintained.

This comment helped clarify an additional part of the problem, which was: Instead of attempting to merge the Nexus One Git tree, which would preserve all the history of that development, Daniel just based his own set of patches off of the tip of their code base and submitted just those patches to the Linux folks.

But some Linux developers, such as Russell King, felt that the Nexus One Git repository had become too filled with micro-patches and non-working messiness before it had reached a workable state. And, King felt that importing such a history into the official kernel tree would be introducing just too much of that mess into the kernel history. So, he argued that Daniel had done the right thing in just submitting his own patches.

Although this "micro" nature of Nexus One development made it more difficult to determine authorship, he agreed with Dima that, somehow or other, a proper chain of attribution should be maintained in that case.

But, as Thomas Gleixner pointed out, and as Daniel Walker himself was probably aware when confronting the initial task of creating his patches, the correct attribution in some cases remains an inherently difficult problem to solve. Thomas took Daniel's patches and compared them with the Nexus One code base and discovered a case where Dima Zavin himself had created a single large commit from a number of other places with 16 authors who could not clearly be paired up with the code they wrote.

Thomas added, "The people who make a lot of noise about this, including you, did not even bother to look at the patches and the originating mess, which is in no way suitable to go near mainline in any form."

Dima actually agreed that including the true history of the Nexus One Git tree would be pointless and that he liked what Daniel had done with the code. However, Dima said his own request for attribution was simply: "that if files are directly copied out of the tree with only slight modifications or if they are copied and stripped down for easier consumption, just say Original Authors: if it's reasonable to gather who the primary contributors are. If that is hard due to lots of commits and squashes, even a Cc: to the people who wrote the code would have been enough."

So, ultimately, it seems that in this case the misunderstandings were resolved and that a more appropriate approach was identified.

This is a pretty interesting situation, however, because it's very common for a large project with its own Git tree to end up in this position, facing a difficult merging problem that can only be solved by some rough wrangling. ■■■