# Zack's Kernel News

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

*By Zack Brown*

## Perl Dependencies

Rob Landley has not had much luck with his patches to remove Perl as a kernel build dependency. As he pointed out recently, the 2.6.25 Linux kernel added some Perl scripts to the build system. This means that anyone wanting to compile Linux since version 2.6.25, would have to have Perl installed on their system. Before the 2.6.25 kernel, that was never required. Rob's patches replaced the Perl scripts with simple Unix shell scripts. But – as he said in his email – he's posted these same patches over and over, and they've never been picked up.

Andrew Morton wrote back, saying he didn't see why a Perl dependency would be a problem for anyone. He pointed out that any decision to remove Perl dependencies would also necessarily include a commitment to remove all *future* Perl dependencies, which might take a bit of work and diligence. He asked what Rob's overall reasoning was for these patches.

Rob replied, "it's a completely gratuitous build environment dependency, and the kernel has a history of removing those." He also added that his shell scripts were at least as simple, if not simpler, than the Perl scripts they replaced. So, accepting his patches would not only remove the dependency, it would also simplify the kernel.

Rob added that cross-compiling – already a tricky proposition – was also made more error-prone by having a Perl dependency.

There was not much discussion on the mailing list. But clearly, there's resistance to removing the Perl dependency. Rob has submitted these patches at least half a dozen times, and each time they fell on the floor.

My sense is that there must be a significant group of kernel developers who want to use Perl in the build environment, or else something like Rob's patches would be a no-brainer. Clearly, fewer build dependencies are better than more, unless the build dependency is actually important.

## Coding for linux-next

James Hogan posted some patches to port Linux to Imagination Technologies' Meta ATP and HTP processor cores. These are multi-threaded, general-purpose CPUs that are often used in digital radios.

Arnd Bergmann looked over the whole patch and said he couldn't find any show-stoppers and felt that James's work was ready to go into the 3.9 kernel.

The more interesting discussion in this thread turned out to be about kernel development processes. For example, James's initial patch-set was based on the linux-next tree. This probably seemed sensible to him, because he was submitting his code to be included in that tree. But Stephen Rothwell explained that, no, kernel development should be based on Linus Torvalds's official tree. The linux-next tree was essentially just a repository for things on their way into Linus's tree and shouldn't be considered a development target on its own.

James pointed out that his port depended on code that only existed in the linux-next tree and hadn't been merged into Linus's official tree yet. He asked if the right approach would be to snarf those parts of linux-next into his own tree to keep his code working and just feed it back to Stephen that way.

Stephen replied that, yes, that would be appropriate, if the code in linux-next was a real dependency and not just a conflict that James's patch would resolve. Stephen said that if James's patch conflicted with code in linux-next, he should just never mind about the conflict and let either Stephen or Linus resolve it themselves at merge time. But, if James's code legitimately depended on code in linux-next, James could feel comfortable snarfing it into his tree – but Stephen exhorted James to let the relevant maintainers know that he expected their trees not to be "rebased."

Rebasing is when you use Git to merge one branch of development with another, by "replaying" the patches from one branch, directly onto the other, and then discarding the merged branch. When you rebase, you're changing the history stored in your Git repository. If you're pushing your tree to anyone else, this could result in unnecessary merge conflicts. Hence, Stephen's advice to James.

Stephen replied that his linux-next dependencies were actually trivial, and he could easily resubmit his patches without those dependencies, to make life easier. He said he'd reintroduce those changes later.

There was also the question of when someone should consider their code "ready" to

**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**.

submit to the linux-next tree. Stephen answered simply – if it's ready to go into Linus's main tree, it's ready to go into linux-next. The linux-next tree is entirely a staging area for code that's on its way to Linus.

## MS Key Blessings

The Windows 8 "secure boot mode" controversy is encroaching into kernel development. David Howells of Red Hat posted some patches that would allow Linux kernels running in secure boot mode to load new cryptographic keys dynamically that had been signed by Microsoft.

Microsoft would apparently only sign runnable EFI PE binaries, so David's code implemented some data wrapping and file-parsing shenanigans to deal with that under Linux. He asked Linus Torvalds to pull the patch-set.

Linus, however, replied, "Quite frankly, this is f*cking moronic. The whole thing seems to be designed around stupid interfaces, for completely moronic reasons. Why should we do this?"

Matthew Garrett replied that Microsoft was the only signing authority for these keys, and they refused to sign anything but runnable EFI PE binaries; so Linux would have to handle that case.

Linus replied that Red Hat could get in bed with Microsoft if it wanted, but that he saw no reason why any of their key-signing code had to live in the kernel. He said, "We support X.509, which is the standard for signing. Do this in user land on a trusted machine. There is zero excuse for doing it in the kernel." He added, "It's trivial for you guys to have a signing machine that parses the PE binary, verifies the signatures, and signs the resulting keys with your own key."

Matthew wasn't happy about the situation either, but he argued that vendors wanted to ship keys that had been signed by a trusted party and that, for the moment, it seemed that Microsoft was that trusted party.

Linus replied that the whole idea that only Microsoft's keys could be trusted, was completely stupid. And, he added that "getting me to care about some vendor that ships external binary-only modules is going to be hard as hell."

The debate continued for quite a while. A significant portion of it involved technical considerations of various other ways of dealing with the secure boot mode key-signing issue. But all of these considerations boiled down to dealing with the fact that certain hardware included these "secure boot" features and that Microsoft held the keys to that particular kingdom.

At one point Greg Kroah-Hartman said, "I fail to see how Microsoft should be dictating how Linux, or any other operating system, works, especially when they aren't even signing the kernel, they are merely signing a bootloader shim and saying 'do your best for keeping the rest of the system secure please.'"

At around the same time, Linus reentered the discussion with a clearer statement of principle. Ultimately, he said, any Linux security features had to have the central characteristic of protecting the user. And, specifically, he said to Matthew, "The whole and only reason I ever merged module signatures is because it actually allows *users* to do a good job at security. You, on the other hand, seem to have drunk the cool-aid on the whole 'let's control the user' crap."

This seems to be the main distinction Linus draws in this case. He seems to feel that, fundamentally, distribution vendors and other third parties should not be able to control how a Linux system is used by the user. The user should be the ultimate authority; and any key-signing features going into the kernel should focus on providing the user with the security features they want. Security features should never prevent the user from doing something that violates the wishes of that distribution or other third party.

Matthew replied to Linus's argument, saying, "The user Microsoft cares about isn't running Linux. The user is running Windows, and someone's merely using Linux as a vector to launch their backdoored Windows kernel. How do Microsoft protect that user? They blacklist the signature used by that Linux bootloader. If we want to protect the user's ability to boot Linux, we need to protect the Windows users from having Linux used against them."

Linus said that none of that mattered. He told Matthew, "Stop arguing about what MS wants. We do not care. We care about the *user*. You are continually missing the whole point of security, and then you make some idiotic arguments about what MS wants you to do. It's irrelevant. The only thing that matters is what our *users* want us to do, and protecting *their* rights."

A little later, Linus outlined more of what he thought Linux security should be:

*"Instead of pleasing microsoft, try to see how we can add real security:*

- *a distro should sign its own modules AND NOTHING ELSE by default. And it damn well shouldn't allow any other modules to be loaded at all by default, because why the f*ck should it? And what the hell should a microsoft signature have to do with *anything*?*
- *before loading any third-party module, you'd better make sure you ask the user for permission. On the console. Not using keys. Nothing like that. Keys will be compromised. Try to limit the damage, but more importantly, let the user be in control.*
- *encourage things like per-host random keys – with the stupid UEFI checks disabled entirely if required. They are almost certainly going to be *more* secure than depending on some crazy root of trust based on a big company, with key signing authorities that trust anybody with a credit card. Try to teach people about things like that instead. Encourage people to do their own (random) keys, and adding those to their UEFI setups (or not: the whole UEFI thing is more about control than security), and strive to do things like one-time signing with the private key thrown out entirely. IOW try to encourage *that* kind of 'we made sure to ask the user very explicitly with big warnings and create his own key for that particular module' security. Real security, not 'we control the user' security.*

*Sure, users will screw that up too. They'll want to load crazy nvidia binary modules etc crap. But make it *their* decision, and under *their* control, instead of trying to tell the world about how this should be blessed by Microsoft.*

*Because it really shouldn't be about MS blessings, it should be about the *user* blessing kernel modules.*

*Quite frankly, *you* are what the key-hating crazies were afraid of. You peddle the 'control, not security' crap-ware. The whole 'MS owns your machine' is *exactly* the wrong way to use keys."*

The debate continued on a number of different technical levels for a long time. But, it seems that Linus has very clear opinions about allowing keys that prevent the user from controlling their own system. My sense is that there's virtually no chance that David's code – or anything resembling it – might get into the kernel.

## Hotplug Cleanup/Recoding

Thomas Gleixner reported that the CPU hotplug implementation had been growing gradually monstrous, with a variety of races, undocumented behaviors, and other insanity. He posted a set of patches to rework the entire thing, creating a predictable state machine that would provide symmetry to the startup and teardown process.

Linus Torvalds had some criticisms of the patch. In particular, he wanted to make sure that Thomas went all the way and didn't hold back. Linus apparently *hated* the current hotplug implementation and felt that a lot of hotplug behavior should not be exposed to the user at all, but just kept tucked away behind the scenes where it could be quietly eradicated.

This was Thomas's goal as well. In particular, he reassured Linus that the hotplug notifier chains would be removed entirely, along with the myriad untraceable user notifier implementations that had grown like lichen around the hotplug code. As he put it, "The current hotplug maze has 100+ states that are completely undocumented. They are asymmetric vs. startup and teardown. They just exist and work somehow, aside from the occasional hard-to-decode hiccup."

Thomas said that this whole mess would be replaced by only two user-visible states: `CPUHP_PREP_<datastructures>`, for setting up and freeing the needed data structures, and `CPUHP_ENABLE_<stuff_on_CPU>`, for enabling and disabling facilities.

The rest of Thomas's "state machine" would be entirely invisible to the user and could be dealt with later. As Rusty Russell put it, "Episode II is where we collapse each into sane states as Thomas clarified. That can be reviewed: I'd hate to try to do it in one go."

## Kernel.org Post-Breach Workflow

The kernel.org security breach from back in August of 2011 is still affecting the workflow of certain kernel developers. After the kernel.org admins locked down all services, they reimplemented only a subset of the features they'd offered before the break-in, and in particular they would only give accounts to users who could verify their identify by a signed cryptographic key.

Recently, Paul Gortmaker noticed a bunch of new files in the Documentation directory of the kernel source tree that were not represented in the 00-INDEX file in that directory. He posted a patch to bring the file up to date.

Rob Landley replied, "I've got a script that makes html navigation pages from the 00-INDEX files and another one that parses that to find dead links in both directions. (Files with no 00-INDEX entry and 00-INDEX entries that don't refer to a file.) I haven't run it in forever because the kernel.org guys took everybody's accounts away, and they won't give me a new .ssh key without a blood test or some such, and even if I did jump through the hoops they made ssh go to a git wrapper you can't rsync through, so I can't update kernel.org/doc/Documentation anymore."

He added, "I have buckets of things I want to do to this directory but no longer have a kernel account. *shrug*."

My personal opinion is, I can see why developers might be unhappy at having to change their workflow – but at the same time, most kernel contributors do simply post patches to the mailing list that then get picked up by the relevant maintainers. The break-in was two years ago. Time to move on. ■■■