Zack's Kernel News

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

By Zack Brown

Simple Flash Filesystem

Dan Luedtke recently announced LanyFS – a filesystem to use with any drive you might carry on a lanyard or keychain – essentially, small flash drives. The goal was to make the filesystem so simple that it would work easily on any operating system. In this case, the lack of features was itself a feature.

Richard Weinberger and Marco Stornelli didn't see the point of such minimalism. To them, it just seemed like reinventing the wheel, because other filesystems already existed with a larger set of features. And, Alan Cox gave a link to an interesting article at *http://lwn.net/Articles/428584/* that discussed the ins and outs of trying to code a flash-oriented filesystem.

Dan pointed out that the filesystem's website, at *http://nonattached.net/lanyfs/*, didn't have complete explanations because the project was part of his Master's thesis, and he wasn't sure how much information he was allowed to publish before submitting the work to his professors.

But, he did say that he hoped the filesystem's minimalism would work well with Arduino projects or other small embedded systems that only wanted to read or play files. The Arduino platform had been a particular motivation for him when his Arduino project ran into trouble with FAT32 files that grew too big for that filesystem. Another motivation had been to interoperate with as many other types of filesystems as possible, without worrying too much about ownership information and other metadata.

There was much skepticism. Marco pointed out that FAT32 was really the standard for the kind of use-case Dan was trying to meet. The FAT32 file size limitation didn't seem sufficient to justify a whole new minimalist filesystem.

But, some people did offer actual feedback about Dan's code. Al Viro pointed out a significant security hole. Because LanyFS allowed infinite recursion, it would be trivial for an attacker to overflow the kernel stack, he said.

Al pointed out a few other technical issues and made an interesting comment at the end of his post about endianness. Apparently, Dan's code flipped its byte endianness in place, instead of taking the more laborious route of having specific variables accept only values with specific endianness. Al recommended being extremely verbose and obvious about endianness issues in order to avoid annoying, difficult debugging issues further down the road.

Theodore Ts'o came back to the issue of whether LanyFS was needed at all. He said, "What I would do if I needed to transfer such a [6MB] file, and I didn't have access to high speed networking, would be to use ext2, and then either use the ext2 FUSE driver with FUSE for Windows or Macintosh – or, I would port the userspace e2tools package to the target OS, and use that to access the ext2 file system. And I'd do that because the software is available today, right now, without having to figure out how to port LanyFS to the operating system."

He added, "I also seriously question the niche of people who want to use a thumb drive to transfer >4GB files. Try it sometime and see what a painful user experience it is …."

Dan did get some support though. Carlos Alberto Lopez Perez pointed out that Microsoft was currently pushing their exFAT filesystem as the preferred way to deal with Dan's use-case. But, as Carlos pointed out, "The problem is that exFAT is full of patents and they require you to purchase a license for use." He said LanyFS might be a great alternative to exFAT, especially because movie files were getting bigger and bigger and would eventually be too big for FAT32. However, Carlos supposed Microsoft would be reluctant to support LanyFS, as it was in competition with their exFAT new hotness.

Raymond Jennings also liked the idea of having an alternative to exFAT, given the patent entanglements that were likely to come up if anyone even thought about writing a Linux port. Alexander Thomas also thought that LanyFS would be a fine alternative to exFAT, and he didn't think much of FAT32 as a filesystem either. However, he too acknowledged that it might be an uphill battle getting major vendors to adopt LanyFS.

The debate continued. At one point, Arnd Bergmann mentioned that he had been cooperating with a vendor to produce a flash filesystem that would be very simple and optimized for most flash media. But, he didn't want to go into detail ahead of the vendor's own announcement. So, at the very least, there is interest from various directions in a LanyFS-type filesystem.

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**.

UEFI Support

Matthew Garrett posted some patches to try to limit the root user's ability to modify the kernel. The idea was to support the Unified Extensible Firmware Interface (UEFI). Ideally, the UEFI would prevent a signed operating system from being able to boot an unsigned operating system. This would give hardware and software vendors the ability to control and limit how their products could be used after purchase.

Alan Cox didn't think it would be possible, at the kernel level, to prevent the root user from regaining control. He said, "an untrusted application can at GUI level fake a system crash, reboot cycle and phish any basic credentials such as passwords for the windows partition."

Matthew thought that hostile software trying to phish credentials could be defeated by a Secure Attention Key (SAK). A SAK is a trusted key combination that initiates a known login process. If an untrusted application tried to make the user think the system had rebooted, the SAK would expose the subterfuge by invoking the kernel's native login process, instead of the fake one presented by the hostile software. In response, Matthew's suggestion was simply to implement SAK support in the Linux kernel.

Pavel Machek was dubious about that idea. First of all, he said, SAK had to display identically on all systems to be effective. So, it would have to include the penguin logo and a "This is not Windows" message, all in the kernel code.

The discussion ended there, but it's clear that UEFI support will be part of the kernel in one form or another. It's kind of surreal to hear Linux developers discuss ways of taking away the ability of the user to control their own system. However, as Linus Torvalds once said about Digital Rights Management (DRM), it's just a feature. It can be enabled or not; it can be used for good or bad purposes, and it's impossible to guard against the bad without also preventing the good.

Don't Make ABI Changes... Or Else

Linus Torvalds went on a tear over Application Binary Interface (ABI) changes. He *really* hates those things. In this particular case, Thomas Gleixner had posted what he thought was a simple fix, getting rid of a null pointer issue in the itimer code, but because the change would be to the ABI, Linus replied, "That's not how ABIs work. If it has become something people rely on, it now *is* part of the ABI, and no amount of 'violates the spec' matters what-so-ever. 'The spec' is paper – and worthless. What people actually *do* is all that matters."

Michael Kerrisk put his head in the lion's mouth suggesting that, with enough lead time to prepare users, an ABI change should be OK. He added that if a change were to be made in this particular case, it should be to make Linux match up with other existing systems like FreeBSD and NetBSD. Linus replied:

"YOU SHOULD NOT MAKE ABI CHANGES.

*I don't understand why this seems to be so hard for people to understand. There are exactly *zero* reasons to change the ABI for its own sake, and this whole thread is a wonderful example of how F*CKING STUPID it was to even consider it. There are real and valid reasons to change the ABI, but for every single one of them, there is some external issue:*

– security. We've had cases where we had an ABI that simply exposed too much information.

– implementation issues. Sometimes, we've done something really really badly, and some subtle ABI issue may simply not work. This is basically never about normal system calls used by normal applications, though – it's about things like the whole iptables flaps etc.

*– actual real applications breaking. We've had cases where we simply did things wrong, and portable applications broke. Then we can *try* to fix it, and see if something else breaks from that.*

And quite frankly, for all but the security case, even then we're often better off at least having a compatibility layer for the old cases, even if it was bad and wrong (example: the very original linux 'select()' timeout behavior, where Linux did the documented thing, but nobody else did. Or the various versions of 'stat()' we've had. Or the inotify/dnotify/fsnotify things).

Occasionally some compat model may not be worth it (if the interface is too specialized and there really is just one or two system apps that use it), but that's very very rare to the point where it shouldn't even be considered an issue.

*Quite frankly, our most common ABI change is that we don't even realize that something changed. And then people may or may not notice it. And we've had cases where the same system call returned *different* things for different subsystems, and we tried to make it at least internally consistent.*

But the 'premeditated ABI change just for the reason of an ABI change'? It's bullshit. And it's bullshit whether it shows up in feature-removal or not. (The whole feature-removal file is BS, for that matter, but that's a different issue).

SO STOP DOING ABI CHANGES. WE DON'T DO THEM.

The absolute worst thing a kernel can do is 'change the user-level interfaces'. It has to be done occasionally (see above), and sometimes we do it by mistake, but anybody who does it on purpose 'just because' should not be involved in kernel development (or library development for that matter)."