Zack's Kernel News

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

By Zack Brown

Mercurial Kernel?

Before the security breach that necessitated rebuilding virtually all the services on kernel.org, the Linux Git repository had a Mercurial-based mirror on that site. Joe Perches recently asked whether the kernel.org administrators were planning to put it back up at some point.

Matt Mackall, who had maintained that repository, said it wouldn't be returning to kernel.org and that he was planning to recreate the mirror somewhere else. However, he hadn't been able to get his SHA map file from his kernel.org account after the break-in, so the new Mercurial repository wouldn't be compatible with the old one. He also said that, in any case, it could be a while before the new repository was up and running at its new location.

BKL Gone?

It still feels odd to think that the Big Kernel Lock (BKL) is gone for good. In olden times, people used to say it could never be removed because it was simply everywhere in the kernel and filled too many different needs. Then someone got the brilliant idea of gradually isolating the BKL code without actually trying to get rid of it in one fell swoop. This ultimately led to the creation of a whole bunch of different smaller locks to handle all the various cases that emerged; eventually, the BKL itself did finally wink out of existence.

But even though no part of the kernel actually uses it, remnants of it still remain in the code comments and perhaps in the organization of certain areas of code. Just recently, Davidlohr Bueso posted a patch to eliminate a few lingering BKL-related comments in the USB driver.

Kernel-locking code is interesting, because it affects the granularity of multiprocessing systems. If a lock is too intrusive, other users can have a choppy experience, which can make music and game playing less enjoyable – and it can be problematic for delicate medical monitoring devices. But, the problem with creating smaller, finer grained locking mechanisms is that it adds to the overall complexity of the kernel. It's an interesting balancing act.

Quotas on TmpFS

Besides cleaning up old BKL comments, Davidlohr recently posted a patch to try to eliminate one source of denial-of-service attacks on Linux systems. On most systems, regular users are free to fill up the `/tmp` directory with as much data as they want, in spite of any quota system restricting their home directory. Of course, files in `/tmp` might disappear at any time, but for temporary use, they can come in handy. And, if a user's only goal is to tie up system resources, file integrity on `/tmp` won't be much of a concern.

Davidlohr's patch would create a user quota system that would apply to the `/tmp` directory, and to all tmpFS filesystems. But, in order not to shake things up too much, the default quota would be unlimited, and system administrators could add tighter restrictions if they felt the need.

The idea turned out to be a somewhat controversial. Christoph Hellwig felt there was no need for a new mechanism to handle this feature – it could just as easily be implemented as part of the usual user quota system. But, Lennart Poettering objected that this would require userspace activity to configure the `/tmp` directory each time it was mounted.

A number of folks, including Alan Cox, got involved in the technical discussion. It wasn't always clear when someone was objecting to Davidlohr's overall goal or just to the particular way it was being implemented. Ultimately, the discussion ended inconclusively, but I think, at the very least, everyone agrees that the `/tmp` directory has a denial-of-service issue that's been around for years and that fixing it would be cool.

Contiguous Memory Allocation

Marek Szyprowski at Samsung has been working on writing support for allocating blocks of contiguous memory. He submitted a new patch, using code from Michał Nazarewicz and others. One neat feature, he pointed out, is that the code can generate contiguous regions of RAM by relocating system memory itself after boot-up.

One problem Marek identified with the current version of his code, and that was subsequently verified by Sandeep Patil, occurs when system pages are themselves in the middle of some kind of operation. If they have work still pending, Marek's code might fail to migrate the system memory to an out-of-the-way location. Sandeep was able to re-

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**.

produce this failure in 100% of his test cases. Small embedded systems would typically be the beneficiaries of contiguous memory allocation features. But, contiguous memory could also provide a speed-up for regular systems. In general, however, only specialized systems would benefit from this type of feature.

Security Holes in /proc

Vasiliy Kulikov didn't like that `/proc/interrupts` was world-readable. He said that because it contained the number of emitted interrupts, it allowed a hostile user to see how many characters were in another user's password. He posted a patch to close this off, but Valdis Kletnieks objected, "This whack-a-mole 'turn off permissions on generally useful files because there's an exposure' really has to stop. On probably the vast majority of Linux systems, it's an embedded or a laptop/desktop, and if you have a malicious user running code on it already, the fact they can find out how many characters are in the password is the *least* of your problems."

Valdis suggested that any change of this nature should be done as part of an overall security model. There should be a centralized security system, he said, that would control permissions to sensitive data in a more fine-grained manner than just cutting off access entirely to everyone.

But Vasiliy pointed out that, one way or another, security holes had to be plugged. If private information was leaking out to where regular users could see it, then that was a bug that needed to be fixed. In the absence of Valdis's suggested "overall security model," these smaller fixes were still needed.

H. Peter Anvin suggested creating a new mount option for procFS, that would allow some people to read `/proc` files and not others. He pointed out that by just blanketing security constraints onto all files that might conceivably pose a security risk, the kernel folks were just forcing users to become root more and more often in order to accomplish regular system tasks. Having more root users, he said, was not a good solution to the problem.

Valdis liked Peter's idea, and the discussion continued until Linus Torvalds came in, with:

*I want *one* global policy that the kernel would actually know about: is the user physically at the machine right now.*

Sadly, I don't think the kernel has any good way to figure that out automatically.

Because quite frankly, a lot of the /proc files should be "root or desktop user." If you control the hardware, you should damn well be able to see the interrupt counts in order to do bug reports etc. without having to sudo or similar.

Torvalds went on to say, "The person in front of the hardware really *is* fundamentally special. Right now all the distros do magic things with the audio device because they know the person in front of the machine is special. But all those things are ad hoc per device, and never cover things like random /proc files etc."

This is a very interesting overall statement about security. According to Linus, a person who has physical access to the machine is virtually the same as the person who has root access. The assumption seems to be that if you have physical access, you can do whatever you want to the system anyway (juggle it, throw it against the wall, etc.), so there's no real way for the software to guard against your actions.

After Linus's comment, a lot of big names like Alan Cox, Greg Kroah-Hartman, and Theodore Y. Ts'o jumped on board with their own ideas about how to address the security issues; nevertheless, the discussion ended inconclusively. ■■■