Zack's Kernel News

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

By Zack Brown

Disposition of Binary-Only Modules

There's an interesting legal question surrounding the Linux kernel: Are third-party, binary-only kernel modules a violation of the GNU General Public License? The question has not been settled in court one way or the other, but I imagine it ultimately will be.

In fact, a vast array of binary-only modules are available for Linux. They are from many companies who want their hardware to function under Linux, but who don't want to give away too much information about the inner workings of that hardware. If a court case determines that binary-only modules violate the GPL, those companies will have to decide whether to abandon support of Linux or to release information that so far they have regarded as secret.

Until the courts rule, the issue can still be addressed in the kernel code itself, and it is. Long ago, something called `EXPORT_SYMBOL_GPL` appeared in the kernel. It provided a way for kernel modules to state their license explicitly in the code itself. If a driver exported that symbol, it meant it had been released under the GPL. Then, kernel developers started making certain parts of the kernel accessible only to drivers that had been released under the GPL.

It's a very touchy subject. Of course, the kernel folks could simply make all interfaces depend on `EXPORT_SYMBOL_GPL`, but they don't. Much of the kernel is still available for binary-only drivers to use, and Linus Torvalds has stated that he's OK with binary-only drivers. Of course, the fact that Linus is OK with it doesn't make it legal. But, it probably has some influence over whether a lawsuit will be filed at all.

Recently, with the release of kernel version 3.1, Christoph Hellwig got a patch accepted that dramatically reduced the ability of non-GPL'd filesystem code to use the kernel's VFS (Virtual Filesystem) interfaces for reading and writing. As Anton Altaparmakov pointed out in his email asking about it, this one change would force all non-GPL'd filesystems to reimplement the entire read/write code paths. He asked if Christoph's patch had intended to alter the status quo or if it had just inadvertently tightened the constraints.

Alan Cox made his position on this issue clear, stating (as I understand it) that he had never given permission for any non-GPL'd drivers to use the code that he had contributed to Linux (which adds up to a truly vast quantity). So, presumably if it came down to it, Alan would probably prefer there be no more binary-only drivers at all.

Andreas Dilger also replied to Anton, pointing out that the actual functionality of Christoph's code did not require the tighter licensing constraints and that those new constraints didn't seem necessary at all, in terms of functionality.

At this point Linus joined in, saying that the change didn't seem to be intentional on the part of Christoph and that it should have been documented in the commit. He suggested to Al Viro (the VFS head honcho) that the license requirements just be put back to their previous state in that part of the code.

Al agreed that this should be done and mentioned that he'd just missed the change when it had been submitted initially. He also added, "folks, for the future, do not use …`_GPL` on VFS exports unless you have a damn good reason to discourage the use in out-of-tree modules in general. Which needs to be clearly documented."

So, a return to the status quo. No lasting change, but it's interesting to see the divisions between kernel developers on this issue. Some feel very strongly that binary-only drivers are a violation of the GPL, and they want to see that part of the GPL better enforced, whereas others – particularly Linus – don't seem to mind the intermeshing of GPL'd and non-GPL'd code as much.

Seeking kernel.org Features

A while ago, kernel.org had a security breach, and someone managed to implant code into a pre-release of the kernel source. This was a very serious penetration. The problem was caught quickly, but the kernel.org administrators had to reimplement the entire site. Project repositories, mailing lists, user accounts were all completely unavailable for a significant amount of time and only returned piece by piece as part of an overall security redesign of the site.

Since then, kernel developers have occasionally asked about when various features would be coming back online. Things like Git repositories had the highest priority, whereas special-case mailing lists were much lower down on the to-do list.

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.**

Back in October 2011, Boaz Harrosh asked specifically about the *git-commits-head* mailing list, an automated announcement list for new kernel releases. At the time, he noted that kernel mirrors could take up to 24 hours to mirror new kernel releases, so the mailing list was really very valuable in certain parts of the world.

Jonathan Corbet, at the time, reassured Boaz that the kernel.org folks understood the importance of the list and would get to it in due time.

Four months later in February 2012, with many of the newly secured kernel.org features up and running, Boaz posted again, asking about the status of the *git-commits-head* mailing list. This time, Jonathan said he hadn't heard anything about it lately and that his attempts to get a status update from the kernel.org folks had not been too successful.

Remounting Oddities

Jerome Marchand noticed that, when using the `mount` command to remount an already mounted drive, the filesystem's options would be set in a very filesystem-specific manner. Some filesystems would keep the options that had already been set, some would revert to the default options, some would ignore any new options that had been set at the time of the remount, and some would do a mixture of those things.

He also noticed discrepancies between the `/etc/mtab` file and the `/proc/mounts` file, which seemed confusing. He asked on the mailing list what the correct behavior of all these filesystems should be. Jan Kara agreed that the situation was certainly a huge mess, but he didn't see any way to fix it, given there would probably be a metric ton of user-land code that depended on each filesystem behaving a certain way.

Karel Zak replied that there was at least one possibility. He said he intended to implement new options for the `mount` command to allow users to specify explicitly the options they did or did not want on remount. So, the existing remounting behavior would continue unchanged, but users could choose to use the new options to get the more consistent behavior.

PohmelFS Rewritten from Scratch

Evgeniy Polyakov announced that PohmelFS – a distributed filesystem – had been completely rewritten and was ready to be included in the main kernel tree. The old version, Evgeniy said, had been based on NFS and had languished for years on the staging tree, without making real progress.

The new version is essentially just a front end for the Elliptics Network. Elliptics provides a generic distributed key/value storage system, so all that was needed was to present a set of filesystem calls on top of that. Evgeniy said the new version supports regular files, symbolic links, hard links, and the rest of the standard set of filesystem features.

Greg Kroah-Hartman was pleased to accept the patch and helped Evgeniy navigate the signed-off-by patch submission process, as well as the best way to remove the old code before adding in the new implementation.

Device Isolation Groups

David Gibson posted some patches to implement "device isolation groups." The idea was that the system administrator might want to protect the rest of the system from the behavior of certain device drivers. Also, in cases where certain device drivers couldn't be protected from each other, they could at least be part of a group of devices that the main system would be protected against.

The implementation, in general, takes the form of a blacklist, with groups of blacklisted drivers being isolated from the rest of the system. But, David is also considering implementing the feature as a whitelist, where anything not explicitly included as part of the system would be considered a danger.

One of the design elements David was working on involved managing the different states that a given driver could pass through. Having drivers appear and disappear on a running system as the system administrator changed their isolation groups might cause problems to user software. David wanted to figure out a way to prevent that type of confusion. ■■■