Zack's Kernel News

INFO

The Kernel Mailing List comprises the core of Linux development activities. Traffic volumes are immense 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 that take on this impossible task is Zack Brown.

Our regular monthly column keeps you up to date on the latest discussions and decisions, selected and summarized by Zack. Zack has been publishing a weekly digest, the Kernel Traffic Mailing List for several years now, reading just the digest is a time consuming task.

Linux Magazine now provides you with the quintessence of Linux Kernel activities straight from the horse's mouth.

Cryptoloop On The Chopping Block

Since kernel 2.6.3, Andrew Morton has been considering removing cryptoloop from the 2.6 kernel. Cryptoloop arrived from Andries Brouwer in July 2003, during the 2.5 development series, to allow encrypted filesystems to be mounted over loopback. Even at the time Andrew was not entirely on board with it, and now in August of this year James Morris got a good reception to his patch removing cryptoloop entirely.

One problem with removing cryptoloop or any 2.6 feature, is that the kernel is currently in the stable cycle, and even a small user-visible change is bound to meet stiff resistance. But Andrew's approach to the stable series has some nuances that have perhaps existed in an informal form during earlier stable series as well. Among these is his decision to allow the distribution vendors to be the primary source of stable kernels. The mainline kernel, even during the stable series, will aim toward a richness of features and a high level of efficiency. Stability will not be abandoned, but it will not be as pure a goal as in the past.

Andrew has said that his new approach will have to adapt to changing situations as kernel development continues, and as kernel developers and users voice their opinions on the issues. The goal, apparently, is to refine the development process over time, including the now traditional division between the stable series of the kernel and development series.

The reliance on distribution kernels is a new idea in Linux kernel development. Historically, developers tended to feel that the official sources should be sufficient for use on any system. Undoubtedly, many kernel versions will continue to be useful in that way, but the precise relationship between the kernel developers and distribution developers seems to be undergoing a change toward greater decentralization.

Within this context, cryptoloop is one feature that, with its bugs, security problems, and lack of a maintainer, will be taken out of the kernel with less reticence than might otherwise have been used. But, in deference to the traditional concerns for series stability, Andrew seems to be leaning toward removing it in the 2.7 development series instead. ■

Software Suspend Reunification

Back in September of 2003, Patrick Mochel and Pavel Machek were in disagreement about the work being done on the software suspend feature. Software suspend is a mechanism to save the state of a running system and return to it after a reboot. In the course of their disagreement, Patrick went so far as to fork the swsusp code away from Pavel, and create his own pmdisk project.

Forking an open source project, although perfectly legal, has strong elements of culture surrounding it. There are social protocols regarding how or when a project may be forked. One legitimate reason to fork a project is when the maintainer is unresponsive, or else unreasonably rejects a developer's contributions. In September, Patrick forked the code because he believed Pavel was rejecting reasonable patches, and making it impossible to contribute to swsusp.

The projects continued developing and announcing their progress on the linux-kernel mailing list, which only fostered the acrimony; but gradually Patrick did start to slow down his development. Finally, this July, after some private discussions with Pavel, they decided to merge their work and continue with Pavel as the project leader. In the reunification announcement, Patrick apologized for the fork and made up with Pavel. The two have been working together in respectful friendship ever since.

And so this saga joins the ranks of forks that have reintegrated themselves. This is a comparative rarity; the vast majority of forks either take over from the original project, or continue indefinitely as a competing project. Very rarely do both projects continue to make valuable contributions that are then brought together again. Examples of such reunifications include such central projects as the GNU C compiler and glibc. ■

Reiser4 Going Into 2.6

Reiser4 has been accepted into the 2.6.8.1-mm2 kernel. This filesystem, touted by its creator as the fastest in Linux, has been pushing for official inclusion for a long time. Its acceptance into Andrew Morton's -mm series puts it just one short hop away from acceptance into the official tree.

The Reiser folks are very excited about this release, as Reiser4 represents a lot of work and a significant departure from ReiserFS version 3.

Reiser4 is not directly compatible with its predecessor, so to upgrade you must first tar up your directory, create the new filesystem, and then untar the directory. Also, only the core functionality has been included in this release of the kernel; more exotic features of Reiser4, like the facility for accessing multiple small files with a single system call have been deferred until the core code has been accepted. The core code is still sufficient for typical use ■

Scheduler Troubles

A group of developers have been growing increasingly frustrated with the state of the process scheduler in the 2.6 kernel, in particular Ingo Molnar and Arjan van de Ven. These developers found it difficult to do detailed audio work because of lags and jumps caused by the scheduler. The desire for real-time Linux performance has existed among developers for over a decade, and although many programmers have spent many hours working on this problem since it first arose, no one doubts that there is much work remaining. Ingo and Arjan went so far as to create a patch that effectively added many scheduling points throughout the kernel, thus reducing the amount of time between process rescheduling.

The problem with this approach is that it tends to disregard the scheduler work that has already been done, and that continues to be done. Specifically, Robert Love and Andrew both feel that the most likely cause of scheduler troubles are fixable bugs that do not require lots of new infrastructure in the form of scheduling points sprinkled throughout the code.

So far that idea has not gained much traction among the kernel maintainers (Linus, Andrew, etc.), possibly because the scheduler is such a core piece of functionality. Offering many competing implementations in an official kernel might make it more difficult to confirm that a given kernel series is heading toward stability. Still, it is fascinating to watch the struggle to improve the solutions to this very thorny problem. ■

Kernel Space And User Space

Another mechanism for communicating between user space and kernel space has reared its head, and developers are now debating how to keep this new mechanism under control so it doesn't become another ProcFS or I/O Control interface, with entirely unruly behavior locked into the kernel through the need for backward compatibility.

Robert Love, Arjan van de Ven and Kay Sievers have implemented a system of asynchronous notification, or 'kernel events' layer, which wraps around netlink to allow kernel events to be broadcast to user-space. The initial proof-of-concept event implemented by the trio focused on processor temperature detection and reporting. In theory, any number of such events could be identified and reported on using this kernel event notification mechanism.

The problem is how to keep the events layer from overlapping with SysFS – or how to integrate the events layer completely into SysFS if that is what needs to be done. No one is quite sure about the proper organization right now, though clearly the feature is very powerful and potentially quite useful. But if it becomes another /proc directory, with erratic conventions glomping on arbitrarily, in five years time – or less – developers may find themselves wishing they'd never laid eyes on this code. Fortunately the lessons of /proc, ioctl(), and /dev have apparently been well learned by kernel developers, and the discussion of how to avoid that future nightmare has already begun. ■