ZACK'S KERNEL NEWS

Status of the 2.2 Kernel

Back in August 2007, Xose Vazquez Perez asked about the status of the 2.2 kernel tree and noted that version 2.2.26 had been released way back on February 25, 2004. On the other hand, the latest release candidate for 2.2.27 was from January 12, 2005. Willy Tarreau replied that any new release of the 2.2 kernel tree might lead users to believe that it was usable. However, he pointed out that, by now, a lot of security fixes have not gone into that tree, and it is simply too far out of date to continue to maintain.

Xose accepted this explanation at the time, but recently he followed up, suggesting that the 2.2 kernel be removed from the front page of kernel.org. If it is so out of date that no one should use it or patch it, he argued, it clearly shouldn't be advertised on kernel.org. This seems to make some sense; however, at the time of this writing, the 2.2. kernel is still listed on kernel.org with the rest of the kernel trees.

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching ten thousand messages in a given 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. Our regular monthly column keeps you abreast of the latest discussions and decisions, selected and summarized by Zack. Zack has been publishing a weekly online digest, the Kernel Traffic newsletter for over five years now. Even reading Kernel Traffic alone can be a time consuming task. Linux Magazine now provides you with the quintessence of Linux Kernel activities, straight from the horse's mouth.

Cute Way to Schedule Code Removal

Matthew Wilcox had a nifty idea to save Andrew Morton a little headache. The current list of kernel features that are scheduled for removal is kept in a single file called *feature-removal-schedule.txt*. As part of innocently going about their business, kernel hackers who want to schedule a feature for removal have naturally added their items to the bottom of that file. The problem is that everyone then submits their changes to that file as a patch, so all the patches conflict with each other because they are all attempting to add different text at the same place in the file. As a result, Andrew has apparently been resolving these conflicts by hand, which is annoying for him.

Matthew's idea for helping Andrew is to trick the kernel patching tools into inadvertently doing the right thing. For example, the diff tool produces a patch that contains lines of context around the patches it produces so that the patch tool can apply a patch at the proper location in a file. The diff tool also keeps track of the "before and after" state of the part of the file being modified, but because the changes Matthew is talking about are only adding text, the "before" state is empty.

Given this, his idea is to put a simple separator, like "————————", between entries and, most importantly, at the bottom of the file. By doing this, the diff tool will not only have no "before" state for its patch, but it will also only have this generic separator to provide context for its patches. As Matthew points out, this will cause the patch tool to insert each new entry randomly between any two adjoining entries in the file.

That's a neat trick, and it's nice when a neat trick can save somebody time. Ironically, the git tool would not make the same mistake as diff and patch, but because Andrew doesn't yet use git for this side of his kernel work, this little solution can slip through the cracks and just work.

New General Debugging Code

Thomas Gleixner has proposed a cool new debugging infrastructure for the kernel. His idea is to keep a hashed list of kernel objects and perform sanity checks on them whenever they are touched or memory is freed so that red flags are identified before a bug can cause kernel panic or other bad consequences. These sanity checks wouldn't find all bugs, but when they did throw a red flag, it would almost certainly be because they detected a legitimate bug somewhere.Thomas's plan would be to keep the debug code in the kernel, where it could be enabled easily. The kernel wouldn't run with the debug code enabled by default because that would slow the whole system down.

Initial support for Thomas's work was good, and Greg Kroah-Hartman suggested some additional sanity checks. Andi Kleen also suggested incorporating the features of an old patch by Chris Mason, in which a background thread would allocate memory, mark it, and then check periodically to see whether it had been corrupted. Because the memory would only be used for testing, any code that corrupted it would not necessarily cause an immediate problem for the running system, so detecting the corruption would give the user precious debug information that could be stored in logs before any potential problem.

It's very likely that Thomas's work will be accepted into the kernel at some point, and it will probably continue to be extended by these and other suggestions.

Distributing I2C Maintainership

Jean Delvare put out a call for someone to be his co-maintainer of the I2C subsystem. He'd been having trouble keeping pace with the rapid pace of patch submissions and figured perhaps another set of eyes would help. A couple of weeks later, he announced that Ben Dooks had agreed to take on the role, and he submitted a patch to the MAINTAINERS file including the new listing.

Time for Cogito Users to Switch to git

When Linus Torvalds wrote git, he was aiming for the equivalent of a "system call" layer for revision control. His application provided the very low level features for manipulating changes in a directory that met the needs he'd identified for himself after BitKeeper was no longer available. In fact, he saw git as an improvement over BitKeeper because it removed features he saw as unnecessary and enabled other features that BitKeeper hadn't been able to provide, like sane tagging semantics.

Right from the start, the git program was hard to understand because it didn't provide the kind of full-service features everyone expects from a revision control system. Instead of users just being able to type a single command to synchronize their repository with the one upstream, for example, they had to first "fetch" the changes from that repository and then "merge" them into their local repository with another command. Other less common actions were even more complex to perform. Linus did this to keep the operations flexible and powerful. They were not intended to be used as the front end to a repository. He expected and encouraged other people to script their own user-friendly commands on top of the git "system call" interface.

The first and most popular of the scripted interfaces to git was the Cogito application, and for a while, it seemed as though Cogito would become the main tool ordinary users would use in conjunction with git. It can be difficult to keep track of the status of these sorts of projects, but it now turns out that Cogito is no longer maintained, and git itself will provide both the back-end power layer and the friendly front end for regular users.

The git front end has actually been under development for a while and is called the "porcelain" layer. It provides a command set familiar to most version control users and relies on the lower level commands for its implementation. In the course of helping someone who'd had trouble with a git repository and had described the Cogito commands that had revealed the problem, Linus told him, "First off, you really should lay off the cogito thing, it's pretty much guaranteed that any cogito usage will just be harder and less likely to be correct than just using native git (and almost nobody will be able to help you any more – it's not like it's been maintained for the last year)."

With git providing its own front end and Cogito no longer maintained, it looks like anyone who's been relying on Cogito should switch to using git directly. Anyone relying on any other version control system should also switch to git. It's way cool.

Linux Magazine Exclusive