ZACK'S KERNEL NEWS

m68k Revitalized Under git

Geert Uytterhoeven announced a new git repository for the m68k Linux port, replacing a defunct CVS repository. He also posted a patch to add a *make install* compilation target for the m68k build and posted several other m68k updates as well.

MSI HOWTO Rewrite

Matthew Wilcox has rewritten the MSI HOWTO, the first major rewrite since 2004. MSI (or Message Signaled Interrupts) provide PCI devices with an alternative to traditional interrupts that require the existence of a hardware pin on the device. MSI lets the device trigger an interrupt simply by writing data to a specific memory address. Randy Dunlap and Sitsofe Wheeler offered some feedback on the prose, and Matthew updated the doc because of it. Grant Gundler and Michael Ellerman had more technical feedback, which they discussed with Matthew. Ultimately it looks as though this is a welcome rewrite, with all in favor.

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.

Controlling Tracepoints

Steven Rostedt has posted a patch to ftrace, providing a simple interface for users to enable and disable any tracepoint that already exists in the kernel. The new */debug/tracing/available_events* file lists all tracepoints available for tracking, whereas the */debug/tracing/set_event* can be modified to list the events you want to enable or disable.

AMD Performance Counting

Jaswinder Singh Rajput posted some patches to implement performance counting for AMD K7 (and later) processors. The patches count CPU cycles, number of instructions, clock ticks, page faults, context switches, and various other items. Ingo Molnár was very pleased with these patches and tested them, offering some bug reports that Jaswinder was quick to respond to.

Clearer KBuild State When Viewing Help

Cheng Renquan has made some improvements to KBuild. The main user-visible change is to present the currently selected value for any given option visible from that option's help message. Randy Dunlap helped him debug some technical problems but, overall, felt the features were important additions to KBuild.

Improving PCI Detection

As part of a broader swath of PCI patches, Alex Chiang has written some code to create */sys/bus/pci/rescan*. Writing a non-zero value to that file will force a rescan of the PCI buses on the system and rediscover any previously removed PCI devices. Alex also wrote companion code to create */sys/bus/pci/devices/ ... /rescan*, which forces a rescan of a device's parent bus and child buses and rediscovers any devices that had previously been removed from that part of the device tree. He also created */sys/bus/pci/devices/ ... /remove*, which removes the PCI device and any of its children.

nilfs2 Headed For Inclusion

Ryusuke Konishi has submitted nilfs2 for inclusion in the main kernel tree. Up to now, it has lived in Andrew Morton's -mm tree. As Andrew has said, Konishi's efforts have been quite impressive, and he's offered him some advice for further development. Andrew strongly supports the filesystem and plans to send the patches along to Linus Torvalds in the very near future, barring serious objection from other kernel hackers. nilfs2 provides full versioning within the filesystem, including continuous snapshotting and access to older snapshots.

From Og

I had to include the following email in full:

```
Og here.

Greg run off to big hill with
white stuff.

He leave corn bits and paper.

Corn bits have many num-bers.

2 corns.

Num-ber on one corn is 2.6.27.19.

Num-ber on one corn is 2.6.28.7.

Paper have 1 wurd.

Paper say "Update!"

Og no no what paper meens.

Corn only good for wood-chuck. Why
Greg care about corn?

Og con-fus-ed.

Og done.
```

Linus Torvalds replied, "I'm happy to see that we have sunk to a whole new level of professionalism. And imagine that people ever doubted that open-source could ever be mission critical! Ha!"

Official git Repository

Until now, Linus Torvalds has been listed in the MAINTAINERS file as the maintainer of all kernel things not otherwise maintained, but his git repository wasn't listed. Joe Perches recently posted a patch adding *git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git* to that entry.

What Is Staging?

Greg Kroah-Hartman recently offered some clarification on the nature of the "Staging Tree" because its role had changed; also, there seemed to be some confusion about it.

The staging tree, he explained, was the *drivers/staging* directory in the official kernel. Code submitted to that directory in the *linux -next* git repository would be fed directly to Linus during the merge window and would be included in the next official kernel release with little or no objection from him.

Appropriate projects for staging are drivers and filesystems that require no code changes anywhere else in the tree (i.e., they are standalone patches). The only exceptions to this are: firmware can (and should) live in the firmware directory, symbols may be exported from the main kernel code if the relevant subsystem maintainer approves, documentation may live in the *Documentation* directory, although that is frowned upon.

The value of putting code into staging is that it exists in the main kernel tree; therefore, it has the full universe of kernel users available to test it while presenting minimal danger to kernel stability. It answers a need that has been addressed in a variety of ways through the years: How can developers get their code tested by enough users to make it acceptable in the main tree before it actually goes into the main tree?

The restrictions placed on projects going into staging are to make sure that all code preserves kernel stability and is moving in a direct line toward migrating out of staging and into its proper location in the official source tree. Therefore, any project going into staging should be well maintained, either by the person submitting it or by a volunteer who's willing to "babysit" the code. Staging is not a place to "dump code and run away," as Greg puts it. Any code that lives in staging will taint the kernel logs when executed (i.e., it'll print a message saying it ran code from staging).

Bug reports from a tainted kernel will be less likely to find folks willing to debug them. If you're working on code in staging, this puts most of the onus on you to debug it yourself, and if you think the bug wasn't triggered by your code, you should reproduce it on an untainted kernel and submit the report. Then, you'll find plenty of willing hackers to help you.

All of this is by way of isolating staging development from the rest of kernel development. As a developer of staging code, you benefit from an audience of potentially millions of users, but the responsibility of dealing with reports from those users, as well as the behavior of your own code, lies with you.

To me, this seems like a really elegant solution to a problem that has stymied kernel developers for years; no doubt it'll be improved as time goes by.