Zack's Kernel News

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

By Zack Brown

Linux 2.4 Status

The 2.4 kernel can still be found running on servers – and perhaps even on some desktop systems. Willy Tarreau has maintained the source tree for years, but a little more than a year and a half ago he announced that if he didn't get any more critical fixes for it after a year, he'd call it done and stop putting out any more releases.

So, Willy was all set to follow through on his plan, but in the middle of August 2011, someone attacked the kernel.org website, and for at least 17 days, a rootkit succeeded in logging activities and passwords of the kernel developers using that site.

As the kernel.org administrators worked on creating a more secure system, a lot of the features relied on by developers and users had to remain down, including Git repository hosting. With the 2.4 repository unavailable, Willy was surprised to find that various users actually had relied on it and wanted it back. Apparently, while they told him they didn't care about actual versioned releases, having an active Git tree was an ideal place to centralize fixes.

So, he's keeping it going! The repository [1] will remain available for the foreseeable future. Willy won't release any more actual versions, which means no more tarballs on kernel.org or special version announcements, but he'll still update the tree and keep it available for interested users, historians, social scientists, and aliens trying to understand our species.

Kernel Insanity at the Highest Levels

Paul Gortmaker reported that the implementation of `IS_ENABLED`, used in the kernel tree to check if a given configuration option has been enabled at compile time, was causing the generated `autoconf.h` file to bloat by 15,000 lines or so. And, as Linus Torvalds pointed out in his reply, the situation was probably slowing down kernel compile times by a significant amount.

Paul's idea was to limit the ways `IS_ENABLED` could be used, but Linus came up with a bizarre reimplementation that had sparks flying out of his eyes when he posted it:

```
#define IS_DEFINED(x) (__stringify(CONFIG_##x)➥
                       [0]=='1')
```

Andrew Morton responded by saying, "Several weeks ago I asked the assemblage if anyone could think of a way of doing this. It seems that I failed to Cc sufficiently perverted parties."

Apparently, when executed, `##x` is replaced by the input to `IS_DEFINED`, then the result of that (e.g., `CONFIG_FCOE`) is evaluated by the compiler before anything else is done. If that config option has been enabled by the user, then the compiler will detect the existence of the symbol and evaluate it as true (i.e., `1`), and the `__stringify` function will turn that into a single character string `1`. After that, the `[0]` character (i.e., the first character of the string) is checked to see if it is equal to `1`. Because that equality is true, the whole kit and kaboodle also evaluates to true, and `IS_DEFINED` returns a value of true, confirming that the config option has indeed been enabled.

If, however, the config option has *not* been enabled, then the compiler will just evaluate it to the text it is (i.e., `CONFIG_FCOE`, or whatever), and `__stringify` will produce the string `CONFIG_FCOE` instead of `1`. Then, when it checks the `[0]` character, it'll find that it equals `C` instead of `1`, and `IS_DEFINED` will therefore return false, which is exactly what one would want because the config option being tested has not been enabled.

Sick.

Linux Licensing Constraints

Many kernel modules are licensed under the GPL version 2 *and* something else, such as the BSD license. These modules define kernel symbols identifying their specific licensing restrictions, but because of the variety of licenses, this results in lots of kernel symbols.

Luis R. Rodriguez recently argued that there was no need for all that clutter. The whole big pile of them could be replaced by a single symbol indicating GPL compatibility. After all, he said, the whole point is to make sure that a given module is released under a compatible license, so the kernel can decide to give access to GPL-specific interfaces.

So, he submitted a patch, replacing all the occurrences of "Dual BSD/GPL," "Dual MPL/GPL," and the rest, with the single "GPL-Compatible" text. This, he said, conveyed what was needed, without all the clutter.

Greg Kroah-Hartman approved of the patch, but there was big blowback from a number of other developers. Al Viro was the first to object. He pointed out that dual-licensed code was being included in the kernel

INFO

[1] Kernel 2.4 repository: *http://git.kernel.org/?p=linux/kernel/git/wtarreau/linux-2.4.git*

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**.

only under the provisions of one of those licenses – the GPL version 2. The other license (BSD, MPL, etc.) referred only to the copy of the code made available by its contributor. Anyone working specifically on kernel code had to adhere to the provisions of the GPL version 2, under which the kernel itself is licensed.

To that extent, Al said, the text "GPL-Compatible" didn't really mean anything, and was in fact less clear than the text "GPLv2."

He also pointed out that because the kernel was explicitly released under the GPL version 2, it was not compatible with other versions of the GPL – specifically, the notorious GPL version 3. So, to specify "GPL-Compatible" could actually mislead users into thinking the code was compatible with other versions of the GPL.

Alan Cox also objected to the patch. He said that the license tag of any piece of code was entirely the domain of the person or organization submitting that code. "Absolutely and utterly. So nobody should for example be touching an Intel `MODULE_LICENSE()` tag without the say so of Intel legal." He explained that it was very important to avoid ambiguity about which license any given piece of code is licensed under. The ambiguity of text like "GPL-Compatible" could lead to a variety of abuses that the current system wasn't susceptible to.

Linus Torvalds also didn't like Luis's patch, saying, "you are actually removing real information, and just making things harder for everybody."

Needless to say, the patch went nowhere. But, it's interesting to see how certain parts of the kernel are extremely brittle. Try to make any change at all – even one that seems like an obviously good thing – and you're confronted by complex history and precedent. In the case of code licensing, this is even more filled with pitfalls because of already legally murky areas, such as whether or not binary-only third-party drivers are legal. Anything that barely stirs the waters inevitably churns up mud.

Kernel Disassembler

Masami Hiramatsu posted a patch implementing an in-kernel disassembler, so that instead of raw opcodes, anyone debugging kernel oopses would see readable assembly language. Or at least readable-er.

Ingo Molnár liked the idea and made a few suggestions for how to make the output even more readable.

H. Peter Anvin also liked the idea in general, but he pointed out that Masami's readable output would take up a lot more space than the raw opcodes. This meant that even more data would be scrolling off the screen than was currently the case, and the current situation was already pretty extreme. Peter suggested that Masami's disassembler be enabled only by an explicit command-line option at boot time. He didn't think users should have to see the more verbose output if they didn't specifically want to during that session.

In spite of Peter's insistence that regular users be protected from Masami's code, both he and Ingo did seem to think the code should go into the kernel and could be very useful to developers debugging their own code.

An amusing little note about Masami's initial email is that it was posted on April Fool's Day, which Peter pointed out at the time. This might have put it in the same category as the suggestion, once upon a time, that the kernel include an entire LISP compiler within it, or a Perl interpreter. Every year, there are lots of posts to the linux-kernel mailing list on April 1, claiming that Linux is going to become a closed-source project, or some other outlandish thing. Masami's in-kernel disassembler idea had a certain similar ring, but it turned out to be a serious suggestion, and one that will probably be adopted into the kernel. ■■■