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.

Real-time real soon

Frederic Rossi has written AEM (Asynchronous Event Mechanism), which aims to provide native support for asynchronous events within the Linux kernel. AEM is intended to address the needs of certain programs for real-time responses to system events. The AEM tool-set exists as one core module and several satellite modules that may be loaded and unloaded at will, to provide a variety of features.

The aem-sched module, for instance, alters the default process scheduler algorithm, to allow event priorities to play a part in the scheduler decision. Originally written for the 2.4 kernel tree, the 2.6 version of AEM falls a little behind the original code. In particular, at the time of writing this article the aem-sched module is not yet ready for use in the 2.6 kernel tree.

In theory, it should be possible for any user to write their own satellite AEM module, to suit their particular needs; though at the moment no documentation exists that describes how to do this. The history of attempts at real-time performance for Linux has been fraught with controversy, although it appears that by now most of those debates have subsided.

Initially it was thought by many that Linux was not suited to real-time software needs, and would never be able to support the fine-grained control needed for some applications, such as software intended to control sophisticated medical machinery, or many multi-media applications.

But through the work of Robert Love and others over the years, the kernel has become much better at being able to guarantee a certain amount of CPU time to those processes that need it. Quite a bit of work still remains however, and a wide array of developers appear to be tackling the issue from many different angles. Much of the work on real-time features has already gone into the 2.6 kernel tree. ■

End of Kcore

The /proc/kcore interface into system RAM may be going away in 2.6, as it appears that not many people use it, and it is very hard to maintain across multiple architectures. Russell King, maintainer of the ARM Linux port, for example, announced in August that he would no longer even try to support /proc/kcore on that architecture, due to the terrible ugliness of the code.

And Linus Torvalds was the first to suggest removing it altogether, since it had a history of breaking after various kernel infrastructure updates. As he pointed out on the linux-kernel mailing list, there had been very few complaints about those breakages, which indicated that almost no one ever used the interface.

In fact, it turns out that some developers do use it to peek into system memory for debugging purposes, though most of those folks say they would rather have a native, fully featured compiler like kgdb built into the kernel, instead of just a big block of RAM to sift through.

Even Alan Cox feels that this would be a good thing; but historically Linus has resisted all attempts to get a kernel debugger included in the official source tree, on the grounds that debuggers are too often used as a way around truly understanding a given problem. By allowing developers to rely too much on tools like kgdb, Linus fears that the quality of the code will actually suffer, because developers will end up submitting patches that they themselves don't fully understand, just because they appear on the surface to fix a bug.

Linus, however, is apparently in a very small minority of developers that believe this. The vast majority, it seems, feel that a kernel debugger provides a very useful tool for tracking down hard-to-uncover problems.

If /proc/kcore is removed from the kernel, it may be even more difficult for developers to perform run-time debugging tests. ■

Kernel options

One of the oldest controversies in Linux history has finally been put to rest. Randy Dunlap has coded up a feature to allow users to derive a kernel's .config file from the kernel binary itself. There have always been users who have balked at their inability to figure out the configuration options a particular kernel was compiled with.

Either they had got the kernel binary from a vendor who did not include the .config in the distribution, or else they'd compiled it themselves but never kept the .config file, and then one day found themselves unable to reproduce the functionality achieved by that kernel. If only they had the .config file, they could figure out why one kernel worked, and the other did not!

Many patches have been written and submitted to Linus Torvalds over the years, but until this past August, none had been accepted. One reason was that such a feature was not considered to be strictly necessary, since users could theoretically keep track of their .config files by hand; and so a feature to do that for them could be considered bloat. Another was that it was actually trivial for a user to append a .config to a kernel binary and retrieve it at need, without causing any ill effects at run-time.

Apparently not everyone thought the feature was completely unnecessary however, and first Dave Jones, and then

Alan Cox accepted Randy's patch into their personal trees for about five months before Linus finally took it into the 2.6-test tree.

The patch saves not only the configuration information, but also information on the compiler and host machine that were used to compile that kernel. This information should be sufficient to rebuild the target kernel precisely.

When he first submitted the patch in March of 2002, the interface was somewhat stripped down, with no interface in the /proc filesystem. By the time of its acceptance, however, it supported two /proc files, as well as a standalone program for retrieving its data from the kernel.

It seems that, rather than Randy's patch addressing some fundamental objection of folks like Alan and Linus, it is more likely that Alan and Linus' attitude toward this kind of feature has changed, making it more acceptable in any form.

Maintaining CramFS

The CramFS filesystem is in need of a new maintainer. Daniel Quinlan is still officially the maintainer of the compressed filesystem, but he would prefer to step down, and is only waiting for a suitable replacement to turn up.

Together with zisofs and SquashFS, these three filesystems provide read-only filesystem compression under the Linux operating system.

SquashFS, a much newer project than either of the other two, appears to have significant advantages, so CramFS is not quite so hotly desired by users as it once was.

CramFS was the original inspiration for the SquashFS project, and SquashFS has managed to improve on a number of CramFS features, including the compression algorithm itself. But the search for a fully-featured compressed filesystem is ongoing.

As of writing this article, a compressed filesystem with read/write capabilities is still unavailable in Linux, and is likely to pose significant problems for any developers who are considering attempting it.

Since data compression relies on abbreviating the patterns found within the full array of the data being compressed, achieving an efficient compression ratio would involve uncompressing and recompressing the entire data-store for each disk-write performed by the user.

While this is generally acknowledged to be a prohibitively slow operation, it is feasible to accept a somewhat lower compression ratio, by applying the compression algorithm only to new data, and thus appending that new data to the existing filesystem store. This method would sacrifice space to gain speed, and appears to be the best chance of true read-write capabilities on a compressed filesystem.

CramFS does not appear to be moving in that direction, however, while SquashFS does seem to be the best hope in this area.

advertisement