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.

Input Layer Info

The input layer has received extensive revision over the 2.5 time-frame, leaving many unanswered questions about how to deal with any problems. As input devices are a fairly integral part of most systems, this has made it difficult for some driver writers to catch up to the current state of the input layer.

With the entrance of the 2.6 tree, however, Vojtech Pavlik has written a nice long document, describing various problems that might come up with the input layer, and what developers can do about them. The document is also very useful to ordinary users, who encounter error messages and odd behavior with no obvious explanations or solutions.

Broken Drivers

Some kernel decisions only make sense in the light of less visible aspects of the context in which they were made. The Initio 9100 SCSI device driver, for example, has been marked "Broken" in the 2.6 tree, in spite of the fact that it might seem to run perfectly well.

Why was it marked broken? Simply because there is currently insufficient error checking in the driver. If anything unexpected happens at run-time, the drive could become massively corrupted.

A number of drivers have been marked "Broken", or just outright removed in the Linux 2.6 kernel for various reasons. Some are similar to the above, while others just have to do with having no visible maintainer. Linus Torvalds has become increasingly strict about excising code that is not maintained, even if it seems usable.

His argument is typically that if a feature is so useful, then someone should be willing to maintain it; and if they aren't, then clearly people aren't so interested in the feature. It may not be entirely accurate, but it does get the point across. Cruft, whether it is actual or only apparent, is not welcome in Linux.

Encrypted files

Michael A. Halcrow has decided to design (and possibly implement) his own encrypted filesystem for use under Linux. There already exist several attempts at such a beast, but all have so far come up short. The question of whether to start a new project or participate in the most promising alternative is an open one, but there seems at least no harm in giving it a shot. Michael's first step, unlike many of the alternative designs, has been to ask as many people as possible for the important features they would want to see in an encrypted filesystem.

This questioning has had some interesting results. Among the desired features, seamless encryption seems to be very popular, in which unencrypted files could be passed through to the disk with only minimal processing overhead, and the encryption layer itself could be presented as a module that would layer itself on top of any filesystem that the user would prefer to use such as ext3, ReiserFS, etc.

Also, among the various options for how to organize the encrypted data, it seems that users would prefer to associate a given key with a specific file, as opposed to a block on the disk or anything else; and also to encrypt as much data about the file as possible, including file size, whether the executable bit is set, etc.

Another popular feature that was requested is the ability to extend Nautilus and Konqueror (and presumably other filesystem browsing tools) to seamlessly take advantage of the encryption features. Also, where feasible, one desired feature that was mentioned involves securely "shredding" a file on deletion, so that it can't be recovered from the disk at all (or at least as much as possible). Whether Michael will implement any of this is still an open question, but an encrypted filesystem is an interesting problem that many developers have struggled to address over the years.

SysFS development

The SysFS filesystem is experiencing massive development activity during 2.6, in spite of the fact that this tree is supposed to aim for stability.

It is not actually so unusual for parts of the kernel to be built out during a stable series, but in the case of SysFS, patches are coming in from many places. All want to ensure that their drivers are in sync with what SysFS is supposed to do; while SysFS itself keeps changing.

In February, Benjamin Herrenschmidt proposed a change to SysFS, to add a "devspec" property to provide the full Open Firmware path to devices on PowerPC and PowerPC64 that had Open Firmware support. The original idea was to have PCI entries include an "OF" path, but that turned out to lead to ever-increasing complexity of user-code, and of the SysFS directory tree.

As Linus Torvalds outlined in the linux-kernel mailing list, the PCI layer should not have "magic" knowledge of particular platforms; but, the platform layer could refer back to the PCI layer as necessary. All of these decisions taking place in the 2.6 time frame incite driver developers to submit patches to make the best use of these new SysFS changes. Already some developers (notably Alexander Viro) are complaining that SysFS development should at least start to slow down, if at all possible.

BIOS sanity

One perennial problem in operating system development is the proliferation of BIOSes in the various available hardware. These BIOSes tend to be closed source, and more often than not, quite buggy. Recently, some of these issues surfaced in the 2.6 kernel tree. Tony Lindgren noticed that some BIOSes reported incorrect values for CPU speed and voltage values.

In particular, his Emachines m6805 claimed his 1800MHz CPU was only running at 1600 MHz. In response to this, he created a patch to perform sanity checks, by attempting to confirm the validity of the BIOS' claims with regard to a running system.

Many, many such patches have gone into the kernel before now, and many, many will go into it in the future. It's a pity more chip manufacturers don't publish the source code to their BIOSes, or at least respond to bug reports in a timely fashion. ■

C++

The relationship of C++ code to the kernel is a complex one. Officially, the kernel is written in C, and C++ has no place either in the kernel code proper, or in any of the modules which are written for Linux. In practice, however, it is possible to carefully write a kernel module in C++, if the module is restricted to using only a subset of the C++ specification.

In addition to this, it is possible that a module may include within itself, a patch to modify Linux's handling of C++ code to be more inclusive. Such patches may or may not ever be accepted into the official kernel tree, but they do allow users to compile and run the driver if they so choose.

The reasons for attempting to write drivers in C++ tend to boil down to personal programming matters. This past January, as an example, quite a bit of effort was expended by developers in order to get a C++ kernel module running under 2.6, just because the module had originally been written in C++, and so it was easier to go through the complicated hoops of supporting it under 2.6, than it was to just rewrite the (quite sizable) driver.

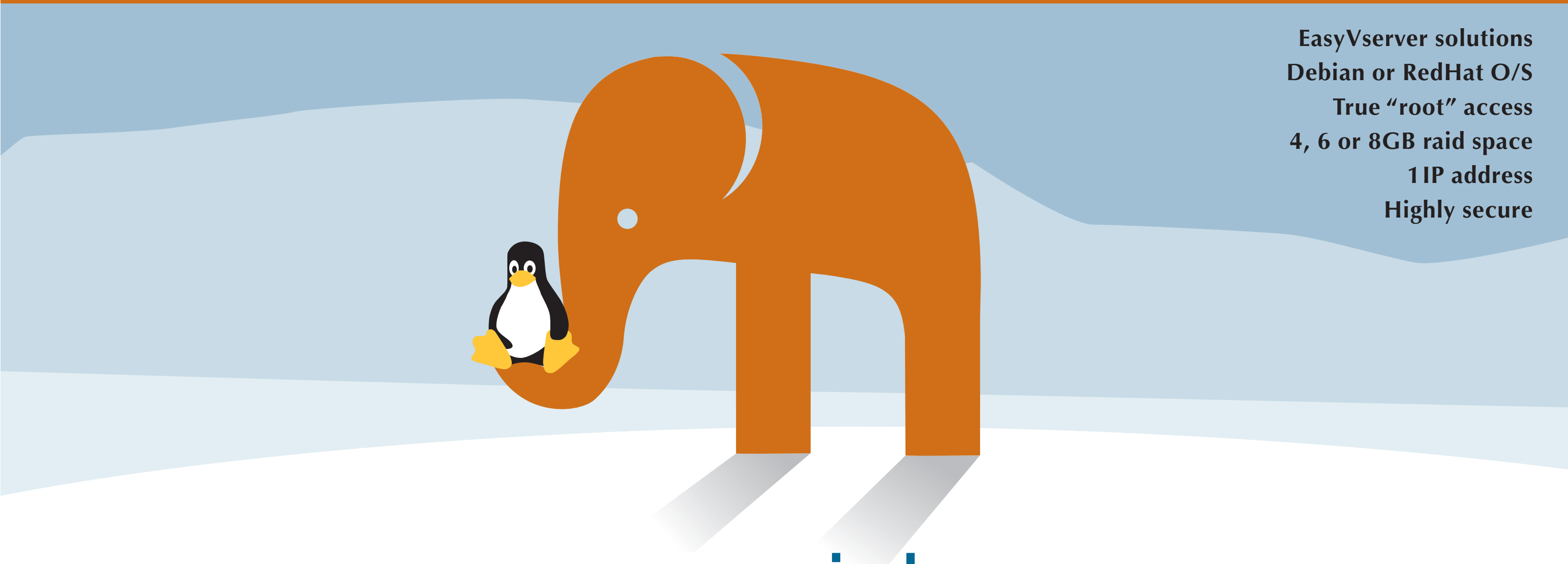
This does leave the question of why C++ was chosen in the first place unanswered. One reason that at least has some legitimacy on its side, is that a driver may originally have been written for another operating system, which supported C++ drivers, and then needed to be ported over to Linux.

In that case, it would be conceivable that the work of supporting C++ would be less than the work of porting the driver to C.

In most cases, however, developers that tend to choose C++ coding for their kernel work, seem to do so because they just prefer C++ programming styles and would like to see more C++ code in the kernel. ■

EASY TO BUY • EASY TO SET UP • EASY TO SEE

EasyVserver solutions
Debian or RedHat O/S
True "root" access
4, 6 or 8GB raid space
1 IP address
Highly secure



p-p-p-pick up
a dedicated linux server

Make the most of Linux technology with the big name in Web registration and hosting packages. Our flexible, scalable, secure EasyVserver solutions start at just £39 per month. Back up by unrivalled support and know-how. If you want the best of Linux come along for the ride.

log on today at: **www.easyspace.com**

Easyspace
your perfect partner for the web

Established in 1997 with over ½ million customers. Accredited ICANN registrar & nominet member. Prices exclude VAT.