

# What on Earth is...

# EFI?

For more than twenty years, PCs have relied on a BIOS to boot up and initialise hardware. Is it time for a change? asks **Paul Hudson**. Intel certainly thinks so...

» I rather like my BIOS as it is. Isn't this a case of "fix it till it's broke" that you told us not to do in your Opinion this month?

Well spotted, and that's true to some extent. However, let us run you through a 'true-to-life' example scenario:

**User:** "Hi there. My games are running kinda slow, and a friend told me it was because my AGP Aperture was too low. She said I need to use my B-I-O-S to make the changes? How can I do that?"

**Tech support person:** "OK. First, reboot your computer. You know when your screen is black and lots of funny text appears on the screen? Wait till it beeps, then press the F2 key on your keyboard. A blue screen will pop up, and you need to use the cursor keys with the + and - keys on your keyboard (although it might be page up and page down on your model) to navigate to AGP Aperture and change it to 128MB. Then hit F10 to save your settings and reboot again."

**User (despondent):** ".....uhm, actually, I guess the games aren't running that slow after all. Thanks anyway. 'Bye!'"

In these days of high-res graphics, plug-and-play devices, and, ahem, 'artificial intelligence', many believe that BIOS has seen its prime and is now well and truly over the hill.

While breaking perfectly good systems is clearly wrong, the PC BIOS has been around for a very long time, and hasn't really advanced much since it was invented in 1981. For example, my AMI BIOS beeps once for a DRAM refresh failure, twice for a parity circuit failure, all the way up to beeping 11 times for an external cache error.

A proposed replacement for BIOS is EFI.

» I see. But how could the European Forest Institute replace my BIOS?

EFI, at least in the computing sense, has little to do with oaks, larches, and elms! Instead, it stands for Extensible Firmware Interface, and it defines a wholly new model for interfacing between operating systems and hardware.

You might think of EFI as being a tiny operating system in its own right. It allows users to boot into a high-resolution, high-colour environment designed to be much friendlier – and much more advanced – than today's VGA BIOS.

As it's such a technological leap forward, Intel have taken the opportunity to offload a lot of the

fluff that has accumulated in PC-compatible BIOSs for the past 20 years, which has allowed them to make far-reaching changes on many fronts. For example, computers back in the early 1980s didn't usually come with hard drives, so all the BIOS information had to be squeezed into the ROM. EFI, however, is stored on its own dedicated hard drive partition, so there's a great deal more room available for online documentation, graphics, drivers and anything else your startup requires.

» That sounds pretty straightforward – what's all the fuss about?

Well, the new look is only part of EFI. Perhaps the most drastic change in EFI is the removal of real-mode operation. Modern operating systems run in what is known as *protected mode*. Protected mode systems are able to use virtual memory, multi-task, use extended memory (>640K), and other such critical things.

However, protected mode was only introduced into CPUs with the 286 processor, which was released in 1982 – after the release of the BIOS – which meant that the gremlin we know as real mode was already in widespread use before protected mode started to be used. Furthermore, because DOS was stuck using real mode for most tasks, real mode continued to be in widespread use up until the early 1990s.

Even today, operating systems still may need to make real mode calls to the BIOS, which are incredibly slow because the CPU literally has to be taken out of protected mode to execute the instructions, then switched back into protected mode to continue.

» Crikey! Get back to the easier stuff!

Well, as we said, EFI provides the capability to have a much more flexible interface presented to users; the whole real mode / protected mode thing is transparent to users.

Graphically speaking, EFI BIOSs have a point-and-click, mouse-operated interface that allows users to configure options, select which OS to launch. One sample EFI BIOS even had an MP3 player built-in!

» Cool! So it's like a pretty front-end to a BIOS?

Yes, and no – it's much, much more than that, although certainly many users won't see anything beyond the more attractive user interface.

For example, EFI systems are designed to have basic networking capabilities built into the system itself so that remote diagnostics can be performed. In fact, one of the goals of EFI is to have systems that are 'plug and play' in the most advanced sense. For example, a new server with no operating system installed on it could just be plugged into a server room somewhere without a monitor, and have the base operating system installed over the network using EFI.

» **Ah, but involving networking must mean drivers, and having drivers at the BIOS level must surely be very complicated...?**

Not at all! In fact, EFI was designed to be very flexible with regards to drivers, and utilises a special language known as EFI Byte Code (EBC) so that drivers are cross-platform.

Put simply, the drivers are written in various languages, then compiled down to EBC so that they are cross-platform. These drivers are then interpreted when they are called during system start-up, and, potentially, an IA-64 driver will work just as smoothly on an IA-32 platform. At least, that's the *plan*...

» **Does that mean, then, that EFI is available on several architectures?**

Yes, but as Intel are leading the drive towards EFI, it's only currently available on Intel platforms. So far, that's the Itanium architecture (IA-64), x86 (IA-32), and XScale architecture. However, the specification for EFI is freely available online for anyone to read, and hopefully more hardware manufacturers should be making use of it as time goes by.

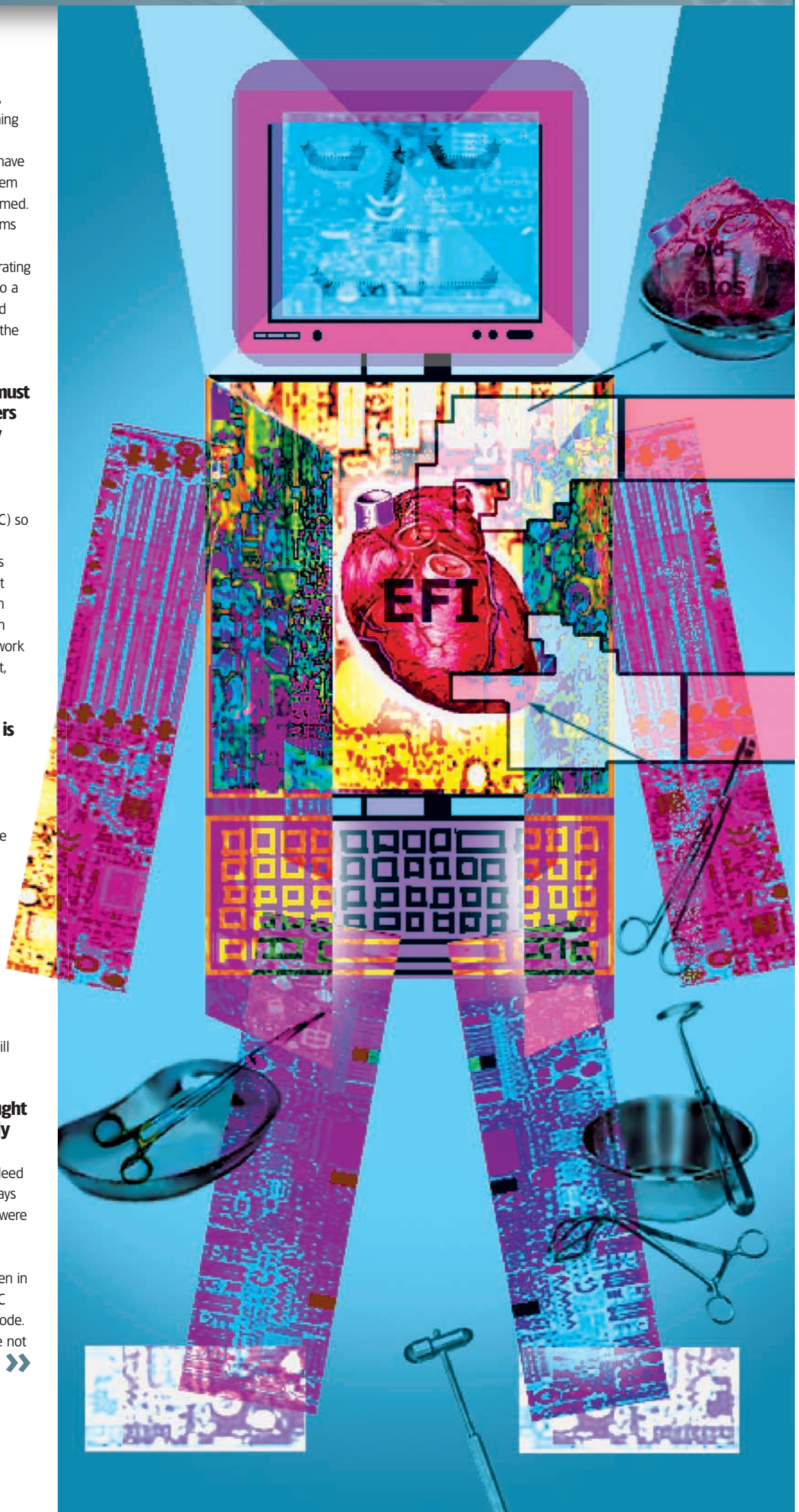
If you didn't know already, XScale is Intel's ARM v5 PDA CPU, and that alone demonstrates how flexible EFI can be.

The key thing, though, is that it's common source code throughout – the same C code will compile and run on every platform.

» **Wait a minute – C code? I thought BIOSs were written in assembly for maximum speed?**

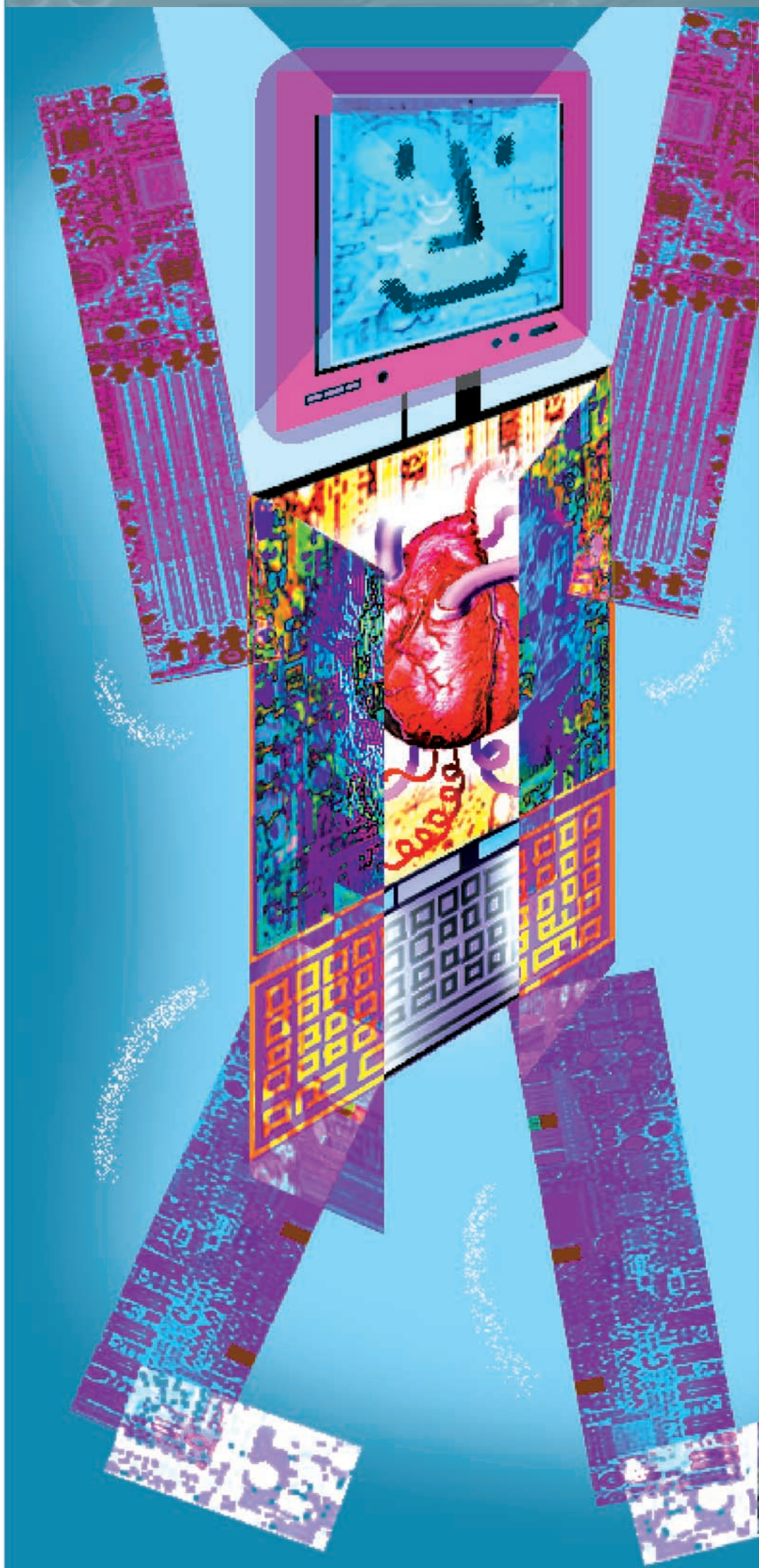
The current generation of boot loaders are indeed written in assembly because back in the old days of computing, processing power and memory were severely limited and every effort was made to minimise program complexity. Furthermore, C compilers have come on leaps and bounds even in the past 10 years, and very often well-written C code will outperform hand-written assembly code.

By dropping assembly source files, Intel are not only making EFI cross-platform, but they are »





# WhatOnEarthEFI



« also adding a large performance boost to the system. Furthermore, because C is a much higher-level language, the overall complexity of the system can be increased: this is where the high-res graphics come in, and also the custom device driver modules.

Intel actually said it best themselves, "The BIOS is the last place on the PC where people have to write in low-level assembler code, and we want to end that".

As to why C was chosen in place of C++, it's anybody's guess. However, C code generally (at least historically) has a speed advantage over C++.

## » Custom modules? You mean anyone can write an EFI device driver to suit their own needs?

Precisely! While a few people still believe that use of assembly is I337, most realise that fewer errors are made in a higher-level language. Furthermore, because EFI replaces nearly all of the BIOS functionality that currently exists, new developers need a great deal less training in the archaic ways of BIOS systems. Instead, they are taught write-once, compile-anywhere techniques that make the whole system more flexible and streamlined.

## » So, customers get a more friendly boot interface and also remote administration. What's in it for manufacturers?

Switching from assembly to C allows greatly reduced time to manufacturing for suppliers. This is partially because C code is much easier to read and write than assembly, but also because the source code for EFI is shared across platforms. Furthermore, having an easier-to-use system reduces manufacturers' support calls, so everyone is a winner.

For peripheral manufacturers, the chore of device driver writing will be greatly simplified because of the EFI Byte Code. Furthermore, the EFI specification defines a collection of device protocols that can be used to unify PC devices once and for all. For example, there's a USB Host Controller Protocol, a Universal Graphics Adaptor Protocol, and a Simple Pointer Protocol – all so that the pre-boot environment is as streamlined as it possibly can be.

## » EFI will be that easy to use? It sounds too good to be true!

Take a fairly common task: flashing your BIOS. Think how you do that. Usually it requires booting up into a simple operating system, coaxing your CD drive into operation, then running a risky ROM patch. With EFI, bootable flash update CDs can be provided that take all the hard work – and all the risk – away.

Furthermore, because the EFI data itself is stored on a hard drive, backups can be made in case of problems.

## » How come no one has thought of this before?

Well, that's not entirely true. The OpenFirmware initiative [www.openfirmware.org](http://www.openfirmware.org) contains a lot of the same functionality as EFI: pluggable, machine-independent drivers, network protocols, and self-test diagnostics.

The key difference is that EFI is backed by some very large companies indeed, which has triggered very quick adoption at the enterprise level. Indeed, EFI is mandatory for Itanium and Itanium 2 systems.

Furthermore, EFI is much more wide-ranging than OpenFirmware. You can think of OpenFirmware as being an evolution of the BIOS, whereas EFI is much more of a *revolution*.

## » What else can it do?

Because EFI is much more advanced than existing BIOS solutions, and also because it can have third-party drivers tacked on with little fuss, it's possible to use EFI to solve operating system problems. For example, let's say a Windows box you're administering crashes (because we all know that Linux doesn't crash, [cough]) and becomes unable to boot. Using EFI and the correct tools, you could correct the problem on the Windows partition without the need to boot up.

Alternatively, thanks to the network connectivity available, you could temporarily give control of your system over to remote engineers to allow them to solve the problem. Intel calls this 'the afterlife'.

A clever feature of EFI, which really demonstrates how much more advanced EFI is when compared to BIOS, is that EFI allows you to run a simple command prompt shell window to perform command-line tasks on your computer *inside* the pre-boot environment. Naturally it's not designed for usage beyond running diagnostic tools, but that may change.

## » Now for the most important question: Does it work on Linux?

Yes! EFI has had excellent support for Linux for over a year now, particularly on the IA-64 kernel port. Furthermore, adoption of EFI will actually help Linux a great deal, because, owing to the cross-platform drivers that EFI uses, Linux could potentially use precisely the same drivers as Windows. Whether that's a good thing or not is entirely subjective!

Intel have really been pushing development of EFI for some time, and have computers working smoothly with both IA-64 Linux and Windows XP. Although there are no plans to fully integrate EFI in the 2.6 kernel, at this rate it won't be long before other Linux ports join the Itanium port.

## » I keep hearing 'Itanium' and 'Intel'. Is EFI Intel-only?

Well, as we said, that's how it is currently. EFI was designed to promote a uniform pre-boot

## Open Firmware explained

### Yesterday's machine-independent BIOS

Open Firmware is the name given to the IEEE 1275 Standard for Boot Firmware: Core Requirements and Practices, and is a specification for a machine-independent BIOS designed to find and configure compliant hardware. Since its launch, IEEE 1275 has been officially withdrawn by the IEEE, and has no real official support today.

However, there is – or perhaps *was* – a great deal of community support behind a group of developers who formed the Open Firmware Working Group with the aim of continuing development of the standard and creating practical implementations for mainstream use.

I say that there perhaps *only was* support for Open Firmware, because the support has slowly died out as time has gone by. There *are* still people out there who fervently believe it's the way forward, but, for example, the 'What's New?' section of the Open Firmware website is dated 1999 – hardly up-to-date! Open Firmware was a great idea, with backing from Sun and Apple, but it's quite possible that it has had its day now that EFI is approaching a critical mass popularity.

While it's possible that Open Firmware and EFI may duke it out in the near future, it's likely to be a short fight – especially with EFI's continued heavyweight backing from Intel.

environment for Intel platforms, but the wide-ranging scope of the system allows EFI to be ported wherever it is needed. Given that Itanium and XScale processors both support EFI and are pretty much polar opposites of each other, anything is possible!

Remember that Itanium was a joint initiative between Intel and Hewlett-Packard. Some of the lead engineers on EFI work for HP, so by no means is EFI limited to one company.

## » Will it run as fast as existing BIOSs?

In many cases, EFI will perform much, much better than current BIOSs. However, once you're booted into your OS, this shouldn't be noticeable anyway as Linux will still do most of the hard work itself.

With regard to the speed that new features and functionality can be programmed for EFI, the worst case scenario predicted by Intel is that EFI will be three times faster.

## » So, is that "it" for BIOS, then?

Not necessarily. In fact, to support backwards compatibility for those 'just in case' scenarios, EFI boards may include a legacy BIOS subset locked deep away from the OS. This option would provide the best mix of new functionality and backwards compatibility.

## » Where can I learn more?

Intel have a large amount of documentation on their website, at [www.intel.com/technology/efi](http://www.intel.com/technology/efi), and you can also find the full EFI specification there as well. Also on the same site, Intel have released a full sample implementation of EFI that is compatible with IA-32 and Itanium.

If you're interested in developing for EFI, Intel produce an open-source EFI Application Toolkit that includes a complete TCP/IP implementation as an EFI protocol, a remote debugger, and more.

Before you charge off to read the EFI specification, one small note: EFI 1.10 (the current

specification) is a weighty 1084 pages at the time of writing this article. Concise it *isn't*, but we'd recommend prospective developers get a copy.

## » I'm a treeware fan: what's available about EFI that I can get from my local bookstore?

*IA-64 Linux Kernel: Design and Implementation* by David Mosberger, Stéphane Eranian, and Bruce Perens (Prentice Hall PTR ISBN 0-13-061014-3) is an excellent book on IA-64 and also kernel internals. Not only does it make very easy reading, but it also is *the* authoritative reference on how the Itanium processor family interacts with Linux. If you own a copy already, turn to page 426 to start reading about EFI. If you don't already own it, what are you waiting for?

## » So, can you sum up EFI in one sentence for me?

Put simply, EFI is the result of adding accepted best practices to the problem of booting and managing the low-level aspects of a computer. That's EFI in a nutshell. **LXF**

## BIOS and you

### Growing old gracefully...

BIOS has been around for an awfully long time, at least as far as other components of computers go, so why has it taken so long to change it?

Well, the problem is that there is an exceptionally large amount of legacy code out there that relies on BIOS being there, particularly code that relies on real mode CPU operation. As a result, it's taking a long time for such old code to die out and therefore no longer require supporting in a BIOS replacement. Furthermore, some parts of computers are just so ingrained into our collective consciousness, that to change them would require a big mental step as well as a big technological step – look how long it's taking for the floppy disk to fade into obscurity!

While BIOS will certainly be used for the next few years at the least, momentum is building up to replace it, and I sincerely doubt that it will be missed.