

Re: [RFC] killing the NR_IRQS arrays.

Re: [RFC] killing the NR_IRQS arrays.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg06133.html>

- *From:* ebiederm@xxxxxxxxxxxxx (Eric W. Biederman)
 - *Date:* Sat, 17 Feb 2007 02:34:39 -0700
-

Russell King <rmk+lkml@xxxxxxxxxxxxxxxxxxxx> writes:

On Fri, Feb 16, 2007 at 08:45:58PM +0100, Arnd Bergmann wrote:

On Friday 16 February 2007 13:10, Eric W. Biederman wrote:

To do this I believe will require a s/unsigned int irq/struct
irq_desc *irq/
throughout the entire kernel. Getting the arch specific code
and the
generic kernel infrastructure fixed and ready for that change
looks
like a pain but pretty doable.

We did something like this a few years back on the s390 architecture, which
happens to be lucky enough not to share any interrupt based drivers with
any of the other architectures.

What you're proposing is looking similar to a proposal I put forward some
4 years ago, but was rejected. Maybe times have changed and there's a
need for it now.

Message attached.

—
Russell King
Linux kernel 2.6 ARM Linux – <http://www.arm.linux.org.uk/>
maintainer of:

From: Russell King <rmk@xxxxxxxxxxxxxxxxxxxx>
Subject: [RFC] IRQ API
To: linux-arch@xxxxxxxxxxxxxxxxxxxx
Cc: Alan Cox <alan@xxxxxxxxxxxxxxxxxxxx>
Date: Sat, 07 Jun 2003 17:05:19 -0700

Hi,

Re: [RFC] killing the NR_IRQS arrays.

I've recently received an updated development system from ARM Ltd, which has caused me to become concerned about whether the existing IRQ infrastructure for the ARM architecture is really up to the job of handling the developments which will occur over the 2.6 lifetime. Essentially we're going to be seeing the emergence of vectored interrupt controllers, and knowing hardware designers, they'll continue the practice of chaining interrupt controllers (which is pretty common on ARM today.) I have some hardware here today which has a vectored interrupt controller chained after two non-vectored controllers. This vectored interrupt controller is on an add-on card, and so has no fixed address space and no fixed IRQ numbering.

Rather than having the job of rewriting this code during 2.6, I'd much prefer to get something sorted, even if it is ARM only before 2.6.

I believe that there are some common problems with the existing API which have been hinted at over the last few days, such as large NR_IRQS. As such, I think it would be a good idea to try to thrash this issue out and get something which everyone is happy with.

Additionally, I've added Alan's "reserve then hook" idea to the API; I seem to remember there is a case in IDE which needs something like this.

Please note that what I am proposing is not to strip out the existing API between now and 2.7; what I am proposing is a structure for 2.7 which can optionally be implemented by architectures and used in architecture specific drivers now if they feel they would benefit from it.

Comments? (other than "wtf are you thinking about this so close to 2.6, are you mad" 8))

Linux Interrupt API

=====

Russell King <rmk@xxxxxxxxxxxxxxxxxxx>

The Linux Interrupt API provides a flexible mechanism to handle and control interrupts within the kernel. The design requirements for this API are:

- must have as little overhead as possible for commodity hardware
- must be easy and obvious to use
- must allow complex multi-level interrupt implementations to exist transparently to device drivers
- must be compatible with the existing API

Essentially, this means that implementation of the existing API must

Re: [RFC] killing the NR_IRQS arrays.

be simple.

The API.

```
=====
struct irq {
/* architecture defined information */
/* must not be dereferenced by drivers */
/* eg, x86's irq_desc_t or sparc64's struct ino_bucket */
};

#define NO_IRQ <architecture-defined-int-constant>
```

When did you need a magic constant NO_IRQ in generic code.
One of the reasons I want to convert the drivers is so we can
kill the NO_IRQ nonsense.

As for struct irq. Instead of struct irq_desc I really don't
care, although the C++ camp hasn't not yet weighed in and mentioned
how that creates a namespace conflict for them.

```
/**
 * irq_get – increment reference count on the IRQ descriptor
 * @irq: interrupt descriptor
 *
 * IRQ descriptor reference counting is mandatory for
 * implementations which provide dynamically allocated IRQ
 * descriptors. statically allocated IRQ descriptor
 * implementations may define these to be no-ops.
 */
struct irq *irq_get(struct irq *irq);
```

```
/**
 * irq_put – decrement reference count on IRQ descriptor
 * @irq: interrupt descriptor
 *
 * Decrement the reference counter in an IRQ descriptor.
 * If the reference counter drops to zero, the IRQ descriptor
 * will be freed.
 *
 * IRQ descriptor reference counting is mandatory for
 * implementations which provide dynamically allocated IRQ
 * descriptors. statically allocated IRQ descriptor
 * implementations may define these to be no-ops.
 */
```

Re: [RFC] killing the NR_IRQS arrays.

Re: [RFC] killing the NR_IRQS arrays.

```
void irq_put(struct irq *irq);
```

We might need this. But I don't think we need reference counting in the traditional sense. For all practical purpose we already have dynamic irq allocation and it hasn't proven necessary. I would prefer to go to lengths to avoid having to expose that kind of an issue to driver code.

Example backwards-compatible IRQ code.

=====

The following example code shows the expected back-compat code required for the x86 architecture. Since x86 uses a fixed table of interrupts, this is relatively straight forward and simple.

Well I just find that comment cute, in the current context :)

I actually don't like the idea of having two version of the infrastructure or for that matter any driver visible changes except the type change at the same time.

That makes the conversion much harder because you have to think about the change. If I have to go in touch a huge percentage of the drivers I don't want to have to think, I don't want something I have to code review. I want something that any first grader can do correct so when I am fatigued after having converted 1000 drivers I can still get it right, and so that there is a chance I don't have to convert drivers.

But otherwise I agree we seem to be largely on the same wavelength.

I also don't like infrastructure changes that never get finished, which is the big danger with providing a backwards compatibility API. Some drivers just never adapt to what is new and better.

I think with a little care I can get 99% of the drivers compile tested by enabling everything in the kernel (allyesconfig?)

I guess when it comes to that I will have to see if I'm crazy or not.

Eric

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>

Re: [RFC] killing the NR_IRQS arrays.