

Re: Use of C99 int types

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-04/1876.html>

From: Renate Meijer (kleuske_at_xs4all.nl)

Date: 04/07/05

Date: Thu, 7 Apr 2005 13:28:03 +0200

To: Kyle Moffett <mrmacman_g4@mac.com>

On Apr 6, 2005, at 11:11 PM, Kyle Moffett wrote:

> On Apr 06, 2005, at 07:41, Renate Meijer wrote:
>> On Apr 6, 2005, at 12:11 AM, Kyle Moffett wrote:
>>> Please don't remove Linux-Kernel from the CC, I think this is an
>>> important discussion.
>
> GAAH!!! Read my lips!!! Quit removing Linux-Kernel from the CC!!!
>
>> As I see it, there are a number of issues
>>
>> - Use of double underscores invades compiler namespace (except in
>> those cases
>> where kernel definitions end up as the basis for definitions in
>> /usr/include/*, i.e.
>> those that actually are part of the C-implementation for Linux.
>
> It is these that I'm talking about. This is exactly my point (The
> cases where
> the kernel definitions are part of /usr/include).

Yes. And my point was all the other occurrences. Specifically those in *.c files. Btw. Not everything in the /usr/include/* path is part of the compiler. For instance, the network definitions are *not* part, nor are the syscall interface or any filesystem code.

>> - Some type that does not conflict with compiler namespace to replace
>> the variety
>> of definitions for e.g. 32-bit unsigned integers we have now.
>
> As I said, I don't care about this, so do whatever you want.

Ok. But you are just one developer (as much as I respect that).

>> - Removal of anything prefixed with a double underscore from
>> non-C-implementation

Linux-Kernel: Re: Use of C99 int types

>> *files.*
>
> *ATM, much of the stuff in include/linux and include/asm-* is considered*
> *"C-implementation" because it is used from userspace. If you want to*
> *clean*
> *that up and start moving abi files to include/kernel-abi or somesuch,*
> *feel*
> *free, but that's a lot of work*

Agreed, and i'll probably won't have much time for that the coming five weeks. So don't hold your breath. However, it would be a good thing if someone (or preferably, more than one) would endeavour to do this.

>>> *Personally, I don't care what you feel like requiring for purely*
>>> *in-kernel interfaces, but __{s,u}{8,16,32,64} must stay to avoid*
>>> *namespace collisions with glibc in the kernel include files as used*
>>> *by userspace.*
>>
>> *Aye, but as I have pointed out several times, these types should be*
>> *restricted*
>> *to those files and *only* those files which eventually end up in the*
>> *compilers*
>> *includes. In every other place, they invite exactly the trouble they*
>> *are intended*
>> *to avoid.*
>
> *Precisely.*
>
> *So if you want to make the millions of patches, go right ahead, be my*
> *guest. :-P*

Well... Someone's got to do it, and "millions" seems to be a bit exaggerated...

> *Until somebody steps forward to clean up the huge mess, nothing will*
> *get done.*

I think it's worth the efforts of more than just a single individual. Especially since you seem to agree the current situation isn't exactly ideal.

>> *So in every place except those files which may actually cause a*
>> *namespace conflict or*
>> *a bug because some newer version does not support __foobar, or*
>> *changed the*
>> *semantics. Since using any __foobar type implies relying on the*
>> *compiler internals,*
>> *which may change without prior notice, it is ipso facto undesirable.*
>

Linux–Kernel: Re: Use of C99 int types

- > *Except the kernel wants to be optimized and work and use what features*
- > *are available.*
- > *The kernel uses __foobar stuff provided by the compiler because it has*
- > *gccX.h files*
- > *specifically designed to take compiler interfaces, provide backups*
- > *when they don't*
- > *exist, and use them (and their better checking) when they do.*

I've checked those files, but the use of compiler specific tricks seems a bit more widespread than that.

- >>> *This is kinda arguing semantics, but:*
- >>> *A particular set of software (linux+libc+gcc), running in a*
- >>> *particular*
- >>> *translation environment (userspace) under particular control options*
- >>> *(Signals, nice values, etc), that performs translation of programs*
- >>> *for*
- >>> *(emulating missing instructions), and supports execution of functions*
- >>> *(syscalls) in, a particular execution environment (also userspace).*
- >>
- >> *Ok. And where exactly are linux and libc when compiling code for an*
- >> *Atmel ATmega32 (40 pin DIL) using gcc?*
- >
- > *Where do you get Atmel ATmega32 from?*

My local electronics shop. I brought it up just to make the point the kernel is **not** part of the compilers implementation, since the compiler will work happily without it.

- > *I _only_ care about what symbols Linux can use,*

Ok. However, the gcc–crowd may see that in a completely different perspective. For them the linux kernel is just one application and linux (as a platform) just one platform they support.

- > *and as I've mentioned, when running under *Linux*, then it just so*
- > *happens that *Linux* is part of my implementation, therefore the*
- > **Linux* sources,*
- > *which by definition aren't used elsewhere, can assume they are part of*
- > *said*
- > *implementation.*

As i've said, I don't care what you are running, OS is **never** part of the compiler. At most some interfaces are in a fuzzy, roundabout way. And even that is a questionable practice.

- >> *The 'set of software' does*
- >> **not* include any OS. Not Windows, not Linux, not MacOSX, since the*
- >> *whole thing might be directed at a lowly microcontroller, which DOES*

Linux–Kernel: Re: Use of C99 int types

>> *NOT HAVE ANY OPERATING SYSTEM WHATSOEVER.*
>>
>> *Nevertheless, gcc works fine.*
>
> *This is unrelated and off topic.*

Just to make the bloody point GCC is not dependent on Linux in any way and hence the kernel is **not** part of GCC.

> *Heck, you've even consented above that*
> *Linux can use*
>
>>> *Without the kernel userspace wouldn't have anything, because anything*
>>> *syscall–related (which is basically everything) involves the kernel.*
>>
>> *Sure. The same goes for every other program. However, it would be*
>> *pretty*
>> *stupid to say the kernel is an integral part of (say) the Gimp .*
>> *More so, since*
>> *the Gimp and GCC run on completely different architectures aswell.*
>>
>> *By the same token, linux is part of XFree86 despite the fact XFree86*
>> *does not*
>> *require linux to run.*
>
> *But an XFree86 binary compiled on FreeBSD, or a GIMP binary compiled*
> *on FreeBSD,*
> *for the most part, will not run on Linux, because the compiler uses*
> *the _Linux_*
> *environment to build the binary, including the _Linux_ headers and*
> *such.*

No... A binary compiled for one platform will not run on another, usually. This still does not imply the linux kernel is part of gcc.

> *The built binary is nearly useless without Linux, but not vice–versa,*
> *hence even*
> *though the binary is not a derivative work of linux, it requires it to*
> *run.*

Weird. How come i'm running XFree86 and the Gimp on MacOSX? There ain't no linux in sight.

>>> *Heck, the kernel and its ABI is _more_ a part of the implementation*
>>> *than glibc is! I can write an assembly program that doesn't link to*
>>> *or use libc, but without using syscalls I can do nothing whatsoever.*
>>
>> *I can write entire applications using gcc without even thinking of*
>> *using*
>> *any 'syscall' or any other part of linux/bsd/whatever. Still... it's*
>> *gcc.*

Linux-Kernel: Re: Use of C99 int types

>
> *Uhh, what exactly is your application going to do?*

Monitoring water levels and sending out alarm messages (by SMS) when the level either gets too low or too high. Furthermore it controls a "stuw", a device for regulating the waterlevel for which I do not know the english term.

> *So it wants to access memory, it faults to the kernel and gets stuff paged in.*

What kernel? What pages? What OS? There is no OS, just a set of library functions i developed (pretty much) myself.

> *It wants to access a file, it does a syscall.*

What files? I write to EEPROM directly. There is no filesystem. Hell there isn't even a vfprintf. In case you are wondering, it's an application for the Atmel ATmega I mentioned.

> *If it wants to allocate more memory, it calls into the kernel.*

Allocate MORE MEMORY? The 4 kb available is full enough as it is.

> *This is all platform specific, and part of the implementation.*

And for that reason, not part of the implementation. The library (glibc in your case) handles (or rather, should handle) the trickery involved. But glibc isn't part of gcc.

>> *<Wishful Thinking>*
>> *It would be nice if Linux became totally independent of any compiler, or at least that coupling between them would be minimal and that the amount of assembly needed would be minimal.*

>
> *If you feel like fixing the compiler to provide good enough interfaces, or fixing the kernel to abstract all of that out, then fine, but remember that the kernel has to deal _directly_ with the hardware and is generally dependent on direct MMU twiddling, which _can't_ be done from C :-P.*

Agreed. That why i said "minimal" instead of "absent". Personally I think the compilers interfaces are pretty good.

Linux-Kernel: Re: Use of C99 int types

>> *It would be nice if linux defined and documented its own platform
>> specific types
>> somewhere in the arch directory, using a consistent (across
>> platforms) naming scheme
>> and used those types consistently throughout the kernel,
>> drivers, daemons and other
>> associated code.*
>
> *Got a patch?*

Not yet.

>> *</Wishfull Thinking>*
>>
>> *<Nightmare>*
>> *Your scenario above. Never-ending streams of compatibility issues and
>> gcc drifting
>> further and further from the ISO-C standard and more and more
>> developers depending
>> on non-standard interfaces, linux growing ever more dependent on
>> support fro features
>> ABC and XYZ being implemented consistently cross platform, so that if
>> I want to use
>> gcc to compile for an AVR, i'm stuck with a shitload of linux issues,
>> kept "for backward
>> compatibility".*
>> *</Nightmare>*
>
> *Linux has a bunch of gcc headers that configure it based on the
> compiler version.
> Change the compiler and we'll add another header, which, though messy,
> provides
> us some safety. It would be _nice_ however, if the compiler had a
> gcc/types.h file
> that just provided it all for us, so we don't need to hardcode it all
> based on
> the architecture specified. Directly defined __gcc_u32 types (Or
> whatever the GCC
> people like) would be even better.*

Why?

>>>> *Nope. The syscall interface is employed by the library, no more,
>>>> no less. The C standard does not include *any* platform specific
>>>> stuff.*
>>>>
>>> *Which is why it reserves __ for use by the implementation so it can
>>> play wherever it wants.*
>>>
>> *The C-implementation, which still does not include the kernel. At
>> most*

Re: Use of C99 int types

Linux-Kernel: Re: Use of C99 int types

>> a few header files, which are used as a basis for standard types by
>> the C
>> implementation, but no more. Any double underscore in a .c file is a
>> blatant
>> error. Most used in .h files are, too.
>
> So how do I get <linux/fb.h> to work? There aren't "just a few",
> there are
> __u32, etc in _everything_ with an ioctl or syscall interface,
> basically
> anything with an ABI.

>> Fine. I assume it does. But #include <linux/fb.h> does not make the
>> framebuffer (nor linux, for that matter) part of the
>> c-implementation. From
>> the two files mentioned above, only stdlib.h is.
>
> Ok, so how do I fix linux/fb.h to _not_ use __u32?

The way other libraries handle it. That ain't magic. The thing that strikes me in the file you mention is that both versions (`__u32` *and* `u32`) are used in parts of the header that are (judging by `#ifdef __KERNEL__`) internal to the kernel.

This implies that merely stripping the double underscore may suffice. After all, that is code that will never be part of the C-Implementation.

>>> I want it to get the correct types, I don't want it to clash with or
>>> require the
>>> libc types (My old sources might redefine some stdint.h names, and I
>>> don't want it
>>> to clash with my user-defined types.
>>
>> Redefining stdint types is (for this reason) a Bad Idea.
>
> So how do I use them in <linux/*.h>?

Either `stdint.h` becomes part of the kernel, which implies your redefined versions in old source are up to some maintainance, or we rely on the `asm/types.h` versions and use them consistently. In either case, use of double underscores where none are required, is not good. Nor is redefining stuff that's defined by gcc, or worse ISO.

>>>> Anything you like. 'kernel_' or simply 'k_' would be appropriate.
>>>> As long as you do not invade compiler namespace. It is separated
>>>> and uglyfied for a purpose.
>>>
>>> But the _entire_ non _namespace is reserved for anything user
>>> programs want to do with it.

Linux-Kernel: Re: Use of C99 int types

>>
>> *The above prefix was an alternative to using a double underscore*
>> *prefix. Using *no**
>> *prefix, should not conflict with the compiler, excepting, of course,*
>> *the types required by*
>> *the standard.*
>
> *But these are also used by c programs.*

Used, yes. Defined, no. The stuff the compiler headers define **without** double underscores is the stuff that is exported for users to use. The stuff that **has** double underscores isn't.

—

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>