

Re: Moving from C++ to VC++

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2005-06/msg00595.html>

- *From:* "Hendrik Schober" <SpamTrap@xxxxxx>
 - *Date:* Thu, 16 Jun 2005 13:54:32 +0200
-

Doug Harrison [MVP] <dsh@xxxxxxxx> wrote:

> On Wed, 15 Jun 2005 16:56:24 +0200, Hendrik Schober wrote:
>> Doug Harrison [MVP] <dsh@xxxxxxxx> wrote:
>>> On Tue, 14 Jun 2005 12:06:19 +0200, Hendrik Schober wrote:
>>>> So which type would you suggest using for
>>>> "longest integer"?
>>>> (May I suggest 'long int'? <g>)
>>>>
>>> So on IA8192, all programs that use long would use 8,192 bit integers?
>>
>> Why not? If it is the longest integer.
>
> Because it imposes insane storage requirements on people who used long back
> in the day when it made at least some sense to use it, i.e. when ints were
> only 16 bits, and long was its guaranteed minimum 32 bits.

I have no idea what you're trying to say.
If I want the longest integer type on the platform, I want the longest integer type on the platform. Of course, that would very rarely help me to save space. Such is the tradeoff.

> Would you really
> like to see fseek take an offset value that's 8,192 bits?

How else would I be able to access files
2^8192 bytes big?

> Is it even
> physically possible to have a non-sparse file that's roughly 10^300 bytes
> long? Seems like an awful waste of space.

"Nobody would ever need more than 640kB." :)

> Suppose int were to remain 32 bits on IA8192, and pointers were 64 bits,
> and size_t were 64 bits, and so forth. (Maybe the fictional IA8192 has
> super-wide registers to speed up certain algorithms, but its address space
> remains mired in reality.) Why would you want to make "long" so insanely

Re: Moving from C++ to VC++

> large?

Now /I/ would like to have 'int' being the native integer type on the machine (64bit, I assume), and long being at least as big.

> Consider these two programs:

>

> 1. Developed for 16 bit ints, 32 bit longs, portably does 32-bit stuff on
> longs.

>

> 2. Developed for 32 bit ints, 32 bit longs, non-portably does 32-bit stuff
> on ints and doesn't use long at all. (Practically speaking, it's portable
> to 32 bits and beyond.)

I see 'short', 'int', and 'long' as "small integer", "fast integer", and "big integer", respectively. If anyone really needed a 32bit integer, they should define one and hunt for that define when porting. Or lobby for an 'int32_t'.

What's wrong about this?

> [...]

>>> The short/int/long

>>> division has been inadequate for a long time. [...]

>>

>> Why?

>

> Because of their lax definition, which fosters writing non-portable code

> while giving the illusion of portability.

I don't agree with you.

(IMO, misuse by itself never belittle the usefulness of things. Bow and arrow were useful despite the fact that they were mis-used to kill humans.)

>>> With all the talk in this thread of moving forward, how about tightening up

>>> rules that made more sense in the day when Ritchie was inventing his

>>> portable assembly language, when hardware was a lot weirder and less

>>> capable?

>>

>> Why? If want the fastest and native (for
>> some definition of "native" anyway) integer,

>> I use 'int'. If it doesn't have to be as big

>> and space is a mild concern, I use 'short'.

>

> Concerning int, my view is that unless told otherwise, it's at least 32

> bits. [...]

Re: Moving from C++ to VC++

And AFIK that's not correct according to the C and C++ standards. They define 'int' to be the native (say "fastest") integer type on the platform.

>> If it has to be as long as possible, I use
>> 'long'. For many things, this is good enough.
>
> Unfortunately, that's not what long means, at least not anymore.

So you agree that what's done now, breaks the traditional meaning of 'long'?

> It was
> always permissible (if uncommon) for long to be (say) 32 bits on a 64 bit
> machine, provided int was no larger.

Yes, it was permissible. It doesn't make sense, though. 'long' really /sounds/ like being the longest integer on the platform.

> However, I don't believe that
> sizeof(long) < sizeof(size_t) was always possible. AFAICT, that's new with
> C99; it at least stretches my imagination to conclude C89 and C++ allow it.
> But that's the way things have gone. At least violations of this assumption
> are easy to detect.

I don't think I have a problem with 'sizeof(long) < sizeof(size_t)'. If I need a 'size_t', I don't use 'long'. If I just need as big an integer as possible, why would I use 'size_t'?

>>> At least we can work within the existing rules to move forward
>>> while taking into consideration the code written for the last couple of
>>> decades of 32 bit machines.
>>
>> 32bit machines are in wide availability since
>> when? Certainly not for what I'd call "decades".
>
> OK, the first true 32 bit mainstream CPUs were the 68020 and 80386, and
> they came out around 1987. OS's took a while longer to catch up, so let's
> call it 15 years. That's pretty close to a "couple of decades", which is
> what I said. In addition, you should factor in the growth in computing
> during that time, not to mention the relative difference in the
> capabilities of 32 bit programs compared to 16 bit programs vs 64 and 32
> bit programs.

And how long do you think 64bit will stick around with us? What growth in computing are we to expect in that timeframe?

Re: Moving from C++ to VC++

Do you really want to suffer from the shyness
to spell out "stupid assumption that breaks"
for all this time?

>> More important, I'd rather take into consideration
>> the lessons learned from going from to 32 and
>> from 32bit, instead of the 32bit itself. 32bits
>> will be gone soon, such transition will come
>> again.
>
> But mindlessly increasing integer sizes is not the way to do it.

Of course not. But noone here spoke of this.
Adhering to what computing standards' intention
is isn't mindless.

I interpret the standards' intention for 'long'
to be a long, if not the longest integer on the
platform. Of course, I might be misinterpreting
the standards, but, AFACS, so far noone here
has challenged this interpretation.

> Things
> like intptr_t, int64_t, int_least64_t, etc make for a far firmer foundation
> that the short/long mess inherited from a portable assembly language
> designed 35 years ago, when hardware was weirder, scarcer, less capable,
> and more expensive. [...]

They have a different meaning that 'long'.
Frankly, I have no idea what 'intptr_t' should
be used for, so I assume it's for storing
pointers in an 'int'. (I assume this because
it correlates with the fact that I have no
idea why one would want to do that.)
'int64_t' means I need exactly 64bit and it
will mean the same on an 8192bit platform.
'int_least64_t' means I need at least 64bit
and it will mean the same on an 16bit platform.
Whereas 'long' simply means I need as long an
integer as possible.

Schobi

—

SpamTrap@xxxxxx is never read
I'm Schobi at suespammers dot org

"Coming back to where you started is not the same as never leaving"
Terry Pratchett

.

• *Follow-Ups:*

- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Tim Roberts

• *References:*

- ◆ **Moving from C++ to VC++**
◇ *From:* NoName
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Severian [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Victor Bazarov
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Carl Daniel [VC++ MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Hendrik Schober
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Carl Daniel [VC++ MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Hendrik Schober
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Hendrik Schober
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Hendrik Schober
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Bo Persson
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Bo Persson
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Hendrik Schober
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Hendrik Schober
- ◆ **Re: Moving from C++ to VC++**
◇ *From:* Doug Harrison [MVP]

Re: Moving from C++ to VC++

- Prev by Date: [*Re: Moving from C++ to VC++*](#)
- Next by Date: [*Re: Pointer to Member Variable has Problem with Arrays*](#)
- Previous by thread: [*Re: Moving from C++ to VC++*](#)
- Next by thread: [*Re: Moving from C++ to VC++*](#)
- Index(es):
 - ◆ [*Date*](#)
 - ◆ [*Thread*](#)