

Re: Exposing bool to non-MS, non-C++ callers

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2007-09/msg00211.html>

- *From:* "Bob Altman" <rda@xxxxxxxxxxx>
 - *Date:* Tue, 25 Sep 2007 08:39:11 -0700
-

I guess my original posting could have been a little more complete (at the risk of being longer and harder to digest). I have a standard DLL that is called from FORTRAN and Ada clients (as well as C/C++ and VB.Net). Some of the functions in this DLL accept a structure as an argument. That structure contains some Boolean (true or false) data.

I initially implemented things in a completely "C-friendly" way, by defining the structure as containing "char" (a.k.a. "byte") data, with the specification that the data must be either zero for false or non-zero for true. FORTRAN callers like to plunk -1 (all bits set) into the structure for "true". C callers tend to use 1 for "true". Some callers use built-in conversions or functions to cast their concept of "true" and "false" to a byte data type, and they are often not consistent from operation to operation, so I may get a combination of 1 and -1 values for "true".

In the code that accesses the structure, it's tempting to use the bitwise logical operators (&&, ||, !) on the byte data, but that scheme fails miserably if TRUE isn't consistently represented. If I want to operate directly on the char data in the structure, I need to be super careful to always cast the char data to bool on the fly (I use a macro: #define tobool(x) (x!=0)). The compiler doesn't help catch places where I neglect to do this. It just performs a bitwise logical operation and gives me a numeric result.

So, I thought, why not define the members of the structure as real C++ "bool" values? The callers would still be obligated to place byte data into the structure, but the compiler might be nice enough to create code that consistently interpreted the bitwise logical operators as operating on zero/non-zero data rather than as operating on 8-bit numeric data.

- Bob

""Jeffrey Tan[MSFT]"" <jetan@xxxxxxxxxxxxxxxxxxxxxxxx> wrote in message [news:fYLk3n1\\$HHA.4200@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:fYLk3n1$HHA.4200@xxxxxxxxxxxxxxxxxxxxxxxx)

Hi Bob,

Re: Exposing bool to non-MS, non-C++ callers

This is an interesting question.

In language level, the C++ bool type is defined as one byte. This can be easily confirmed with:

```
printf("Size of bool: %d\n", sizeof(bool));
```

So, once the VC++ compiler sees bool type, it will cut any value greater than a byte into single byte. Let's take the following sample code as an example:

```
void Test(bool test)
{
printf("Size of bool: %d\n", sizeof(bool));
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
int i = 10000000;
Test(i);
}
```

The VC++ compiler will emit the assembly code below:

```
cmp dword ptr [i],0
setne al
push eax
call Test (41122Bh)
add esp,4
```

As you can see, the VC++ compiler will use compare the input value with zero and only set the single byte "al" register for using with the bool type. This obeys the single byte rule.

Let's come back to your specific problem. If the code goes across the DLL boundary, there is another story. The answer for your question is not consistent. Let's explain.

When your DLL project is compiled into a DLL binary file, all C++ type information is lost. There is only binary code in the DLL. Since the x86 platform Windows requires the parameters to be passed in 4 bytes aligned. Even your function parameter is of type "bool", the compiler will reserve a

4 bytes space for this parameter. Now, in the Exe project, if the caller pushes a value larger than 1 byte and less than 4 byte, Windows will accept

it without any problem, no crash or corruption will occur. Sure, this depends on whether the Exe caller will push a value larger than 1 byte. If the caller also declares the parameter as a C++ bool type (assume the caller

is also a C++ language), I think the caller compiler will also use some type of trick (like the assembly code above used by VC++ compiler) to only pass a single byte value to your DLL exported function.

Re: Exposing bool to non-MS, non-C++ callers

Note: if you wrap the bool type in a C++ structure, union or class, the compiler may not align the bool type at 4 bytes by using "#pragma pack(1)".

Yes, I agree that we'd better use Windows BOOL type for export so that we will not confuse with this problem.

Hope this helps.

Best regards,
Jeffrey Tan
Microsoft Online Community Support

=====
Get notification to my posts through email? Please refer to
<http://msdn.microsoft.com/subscriptions/managednewsgroups/default.aspx#notifications>.

Note: The MSDN Managed Newsgroup support offering is for non-urgent issues where an initial response from the community or a Microsoft Support Engineer within 1 business day is acceptable. Please note that each follow up response may take approximately 2 business days as the support professional working with you may need further investigation to reach the most efficient resolution. The offering is not appropriate for situations that require urgent, real-time or phone-based interactions or complex project analysis and dump analysis issues. Issues of this nature are best handled working with a dedicated Microsoft Support Engineer by contacting Microsoft Customer Support Services (CSS) at
<http://msdn.microsoft.com/subscriptions/support/default.aspx>.

=====
This posting is provided "AS IS" with no warranties, and confers no rights.