

Re: Data alignment problems with sizeof and new

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2005-11/msg00035.html>

- *From:* "Doug Harrison [MVP]" <dsh@xxxxxxxx>
 - *Date:* Tue, 01 Nov 2005 19:27:42 -0600
-

On Tue, 1 Nov 2005 13:12:03 -0800, astdsoftware
<astdsoftware@xxxxxxxxxxxxxxxx> wrote:

>I have a strange problem with a project involving classes with data alignment
>padding. It appears that, in this particular project, the sizeof and new
>operators are not taking the alignment padding into account when calculating
>the size of an object. The struct member alignment is set to 8 (/Zp8), and
>there are no #pragma pack directives. The test class below shows how, given a
>class with the first member being a dword size followed by doubles, the first
>double member is padded out to an offset of 8, increasing the memory
>footprint by 4 bytes. Using the sizeof() operator outside the scope of the
>class, the size returned is the sum of all of the members, but if I use the
>sizeof operator inside the constructor, it calculates the right size.

>

>The real problem is that the new operator only allocates enough memory for
>the object without any padding, and when the constructor starts initializing
>its data members, the last one writes beyond the memory block allocated for
>the class. If I move this code to a different project, it works just fine,
>but I can't find any project options or anything which might explain this
>behavior.

>

>Of course, the problem goes away if I change the alignment to /Zp4, simply
>because the padding goes away, but that only works if I don't use anything
>smaller than 4 bytes in size.

>

>Any clues would be greatly appreciated!

>

>

```
>// TestClass.h
>#pragma once
>class TestClass
>{
>public:
>TestClass(void);
>~TestClass(void);
>unsigned int m_dwData; // Offset 0
>double m_Data1; // Offset 8
>double m_Data2; // Offset 16 (0x10)
>double m_Data3; // Offset 24 (0x18)
```

Re: Data alignment problems with sizeof and new

```
>double m_Data4; // Offset 32 (0x24)
>};
>-----
>// TestClass.cpp
>
>#include "StdAfx.h"
>#include ".\testclass.h"
>TestClass::TestClass(void)
>: m_dwData(0)
>, m_Data1(0)
>, m_Data2(0)
>, m_Data3(0)
>, m_Data4(0)
>{
>int mysize = sizeof(*this); // Evaluates to 28h (Correct Value)
>}
>TestClass::~TestClass(void)
>{
>}
>-----
>// Main.cpp
>
>#include ".\testclass.h"
>TestClass* pTest;
>pTest = new TestClass;
>int tsize = sizeof(*pTest); // Evaluates to 24h (Incorrect Value)
>-----
```

Here's a repost of an message I wrote back in May 2005 that may explain your problem:

<http://groups.google.com/group/microsoft.public.vc.stl/msg/37bb8ca26dbd5c01>

There's a bug in <winsock2.h> such that it changes the packing from the default 8 to 4 when WIN32 isn't #defined. You're #including c.h in two different ways; in a.cpp, it follows <winsock2.h>, while b.cpp doesn't #include <winsock2.h>. I expect this is causing various classes to have different layouts in these files. You can see the underlying problem if you add the following line to the top of c.h and following all its #include directives:

```
#pragma pack(show)
```

Then when I compile using your command line, I get:

```
C>cl -I. -nologo -MDd -EHsc -W3 -DWINVER=0x400 a.cpp b.cpp
a.cpp
c:\temp\c.h(4) : warning C4810: value of pragma pack(show) == 4
c:\temp\c.h(7) : warning C4810: value of pragma pack(show) == 4
```

Re: Data alignment problems with sizeof and new

Re: Data alignment problems with sizeof and new

```
c:\temp\c.h(9) : warning C4810: value of pragma pack(show) == 4
c:\temp\c.h(11) : warning C4810: value of pragma pack(show) == 4
b.cpp
c:\temp\c.h(4) : warning C4810: value of pragma pack(show) == 8
c:\temp\c.h(7) : warning C4810: value of pragma pack(show) == 8
c:\temp\c.h(9) : warning C4810: value of pragma pack(show) == 8
c:\temp\c.h(11) : warning C4810: value of pragma pack(show) == 8
```

I think that in general, if you're going to use the Windows SDK, you should #include <windows.h> before any other Windows header, which would avoid this problem without you having to remember to specify /DWIN32 for these command lines you create yourself.

And BTW, you should avoid the /Zp option like the plague, for the reasons given here:

<http://groups.google.com/group/microsoft.public.vc.mfc/msg/8a94f9c97b8dbf09>

Use #pragma pack instead, either directly or through the <pshpackN.h> and <poppack.h> headers, to control the packing of individual structs.

--

Doug Harrison
Visual C++ MVP

.

-
- *Follow-Ups:*
 - ◆ *Re: Data alignment problems with sizeof and new*
◇ From: astdsoftware
 - *References:*
 - ◆ *Data alignment problems with sizeof and new*
◇ From: astdsoftware
 - Prev by Date: *Re: /Za option*
 - Next by Date: *SIZE T versus size t*
 - Previous by thread: *Re: Data alignment problems with sizeof and new*
 - Next by thread: *Re: Data alignment problems with sizeof and new*
 - Index(es):
 - ◆ *Date*
 - ◆ *Thread*