The alignment declaration specifier is in bytes, not bits

devblogs.microsoft.com/oldnewthing/20101202-00

December 2, 2010

Raymond Chen

Explicit object alignment is not something most people worry about when writing code, which means that when you decide to worry about it, you may be a bit rusty on how the declarations work. (After all, if it's something you worried about all the time, then you wouldn't have trouble remembering how to do it!)

I was looking at some customer code, and there was a class who had a data member with an explicit alignment declaration.

```
class Whatever {
    ...
    __declspec(align(32)) LONG m_lSomething; // Must be DWORD-aligned to make writes
atomic
    ...
};
```

I pointed out that the comment didn't match the code. The comment says that the variable needs to be DWORD-aligned (which in Windows-speak means aligned on a 32-bit boundary), but the code aligns it on a 32-*byte* boundary, eight times as generous as required. On the other hand, maybe they really did want the member aligned on a 32-byte boundary (say to put it on its own cache line).

Turns out that in this case, the comment was correct and the code was wrong. To force a variable to align on a DWORD boundary, you want to say `__declspec(align(4))`. Save yourself a bunch of unnecessary padding bytes.

But in fact, in this case, the customer was simply trying too hard. The code was compiled with default alignment, which aligns integer types on their natural boundaries anyway. The compiler was going to align the variable even if you didn't specify anything.

[Raymond is currently away; this message was pre-recorded.]

Raymond Chen

Follow