

Re: Isn't it time there was a standard align statement?

## Re: Isn't it time there was a standard align statement?

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2007-01/msg01974.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-01/msg01974.html)

---

- *From:* "Malcolm McLean" <[regniztar@xxxxxxxxxxxxxxxx](mailto:regniztar@xxxxxxxxxxxxxxxx)>
  - *Date:* Sun, 14 Jan 2007 16:38:46 -0000
- 

<[artifact.one@xxxxxxxxxxxxxxxx](mailto:artifact.one@xxxxxxxxxxxxxxxx)> wrote in message

It'd be really pleasant (in my opinion) if the next revision of the C language actually allowed some portable control over data alignment.

Compiler-specific mechanisms for this stuff are so varied that it becomes impossible to even abstract the details away behind preprocessor macros.

What I'd like to see:

```
/* per structure alignment */
align(16) struct xyz {
char x;
char y;
int z;
};
```

```
/* per member alignment (obviously padding before the first
member is illegal, so the entire structure would become aligned
in this case */
align(16) struct xyz {
align(16) char x;
char y;
int z;
};
```

```
/* per variable alignment */
align(16) unsigned int x;
```

I don't care about the syntax.

Now, obviously, C is meant to be implemented on everything from self-aware weather-control mainframes, to motorized tie racks, so in the case of the host implementation not supporting the specified alignment, a warning should be emitted and either the closest or

Re: Isn't it time there was a standard align statement?

natural alignment should be given. Warnings can obviously be made fatal with compiler specific switches – and that's no business of the language.

It just seems that this really should be standardized as it clearly is useful for a vast number of programmers who need to get close to the hardware but don't want to stray into assembly code (think AltiVec, SSE).

Sounds like EXACTLY the point of the C language, doesn't it?

I wouldn't mind so much if compiler implementors had come up with a vaguely portable way of doing this, but they haven't even come close. GCC and Intel have won joint first prize for 'most pleasant implementation' though (`__attribute__` or `_declspec()`).

The way to do it is either

1) Have a type called `moststrict_t` which is always aligned properly for any other variable. `malloc()` will only work if such a type exists, so whizzy new hardware that needs types aligned on prime numbers can't be supported anyway.

2) Have a macro `void *align(ptr, type)` which will take a pointer and return the value equal to or above it suitable for storing the type in.

This can be implemented with only the tiniest of tweaks to most compilers, and doesn't add any new syntax to the language.

Note that in practise `moststrict_t` is always double, so any mangement packages will be OK if they align data to double. However it is a nuisance to do this,

.