

Re: void * pointer convert problem.

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2005-12/msg03628.html

- *From:* Keith Thompson <kst-u@xxxxxxx>
 - *Date:* Fri, 30 Dec 2005 21:09:53 GMT
-

Christopher Benson-Manica <ataru@xxxxxxxxxxxxxxxxxxxxxxxx> writes:

```
> nelu <tandauioan@xxxxxxx> wrote:
> (IANALL; apply salt granules as appropriate.)
>
>> struct s1 {
>> int id;
>> char *name;
>> int year;
>> };
>
>> struct s2 {
>> int id;
>> char *name;
>> int year;
>> time_t tstamp;
>> };
>
>> ...
>> void *p;
>> struct s2 b;
>> ...
>> p=&b;
>> ((struct s1 *)p)->year=2000;
>> ...
>
>> the last assignment may not be portable?
>
> That's my reading of 6.7.2.1 (12) of n869:
>
> "Within a structure object, the non-bit-field members and the units in
> which bit-fields reside have addresses that increase in the order in
> which they are declared. A pointer to a structure object, suitably
> converted, points to its initial member (or if that member is a bit-field,
> then to the unit in which it resides), and vice versa. There may be
> unnamed padding within a structure object, but not at its beginning."
>
> There is nothing that I can see which forbids an implementation from
> putting the year members at different offsets (due to the above
> mentioned unnamed padding), making the code you provided nonportable.
```

Re: void * pointer convert problem.

> Of course, non DS9K machines are unlikely to behave in such a
> manner...

But see also C99 6.5.2.3p5:

One special guarantee is made in order to simplify the use of unions: if a union contains several structures that share a common initial sequence (see below), and if the union object currently contains one of these structures, it is permitted to inspect the common initial part of any of them anywhere that a declaration of the complete type of the union is visible. Two structures share a `_common initial sequence_` if corresponding members have compatible types (and, for bit-fields, the same widths) for a sequence of one or more initial members.

Strictly speaking, this guarantee applies only if the two structures are members of the same union. If a compiler can prove to itself that no such union exists in the program, it can theoretically use different layouts for the common initial subsequences of two different structure types. But it's difficult to imagine any non-DS9K implementation going to the extra effort for no benefit I can think of. The most straightforward way to comply with 6.5.2.3p5 is simply to use layout rules that always guarantee the same layout for any common initial subsequence.

If you're really concerned about conformance, you can declare (and not bother to use) a union just to avoid the possibility, but I don't think it's worth the effort.

--

Keith Thompson (The_Other_Keith) kst-u@xxxxxxx <<http://www.ghoti.net/~kst>>
San Diego Supercomputer Center <*> <<http://users.sdsc.edu/~kst>>
We must do something. This is something. Therefore, we must do this.

.

• *Follow-Ups:*

- ◆ **Re: void * pointer convert problem.**
◇ From: Christopher Benson-Manica

• *References:*

- ◆ **void * pointer convert problem.**
◇ From: Eric J.Hu
- ◆ **Re: void * pointer convert problem.**
◇ From: Christopher Benson-Manica
- ◆ **Re: void * pointer convert problem.**
◇ From: Eric J.Hu
- ◆ **Re: void * pointer convert problem.**
◇ From: tmp123
- ◆ **Re: void * pointer convert problem.**

Re: void * pointer convert problem.

Re: void * pointer convert problem.

◇ *From:* Christopher Benson–Manica

◆ ***Re: void * pointer convert problem.***

◇ *From:* nelu

◆ ***Re: void * pointer convert problem.***

◇ *From:* Christopher Benson–Manica

- Prev by Date: ***Re: String reversing problem***
- Next by Date: ***Re: o/p problem***
- Previous by thread: ***Re: void * pointer convert problem.***
- Next by thread: ***Re: void * pointer convert problem.***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***