

## Re: Question about bit-fields and portability

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2008-07/msg01658.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2008-07/msg01658.html)

---

- *From:* Peter Nilsson <[airia@xxxxxxxxxxxxx](mailto:airia@xxxxxxxxxxxxx)>
  - *Date:* Sun, 13 Jul 2008 15:24:48 -0700 (PDT)
- 

DiAvOI wrote:

I read here and there that bit-field usage should be avoided because it's not portable. My question is why it's not portable?

They are perfectly portable, they just can't be pe used portably to read binary files produced on a different implementation.

For example let's say I have a struct in my code with bit-fields like the following:

```
struct bitfield_t {
    unsigned int val1 : 1;
    unsigned int val2 : 3;
    unsigned int val3 : 2;
};
```

How can such a bit-field struct be used in a non portable way? (Can you give me an example if possible)

Unless you make non portable assumptions about the representation of int, and the size of the word in which the bitfields are allocated, and the ordering of the bitfield allocation, then you run into problems reading binary files containing a struct bitfield\_t produced by other implementations.

Note that 'other implementations' can mean the same compiler with different settings.

In my understanding non-portable means that if you save a struct like this in a binary form (in a file for example or send it over the network) and then try to restore it on a platform with different endianness or compiler it probably will be interpreted differently. Is

Re: Question about bit-fields and portability

this the only case in which the usage of bit-fields is not portable?

No there are others, such as assuming unsigned short bit fields are supported, or that a plain int bitfield of 1 bit can have two values, 0 and -1.

Will source code which contains bit-fields like `bitfield_t` work (execute) the same on different platforms (Without considering how the bits are saved on each platform)?

Generally, yes, unless you go out of your way to make them work differently. Unfortunately, going of the way seems to be the status quo. ;-)

--

Peter

.