

# Re: pointer q

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2006-05/msg02627.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2006-05/msg02627.html)

---

- *From:* Flash Gordon <spam@xxxxxxxxxxxxxxxxxxxxxx>
  - *Date:* 16 May 2006 21:11:02 +0200
- 

Joe Smith wrote:

"Flash Gordon" <spam@xxxxxxxxxxxxxxxxxxxxxx>

Joe Smith wrote:

"Flash Gordon" <spam@xxxxxxxxxxxxxxxxxxxxxx>

Joe Smith wrote:

[snip]

```
*sp = (short)*1p; /*  
ill-advised, but Let's  
Pretend */
```

This is very different. If you were trying to fix the code further up so it would compile what you wanted was:

```
short *sp = (short*)1p; /* ill-advised, but  
Let's Pretend */
```

How thick is my head ^ ?

We all make mistakes.

See below.

Am I correct to think that a type of masking occurs here? joe

No, not masking. More a case of only reading part of the data with my "fix". Or with your code writing to an uninitialised pointer and causing half of your socks to

## Re: pointer q

migrate to Alaska.

Am I not reading, with correction, the number that a smaller type thinks a larger type was? My socks are well accounted for, but Scott Knuds not only flew out of my hard drive, but is eating my lunch.

Your version of the code:

```
#include <stdio.h>
```

```
int main ( void )  
{long l = 420000;  
long *lp = &l;
```

```
short *sp;  
*sp = (short)*lp; /* ill-advised, but Let's Pretend */
```

Writes through an uninitialised pointer which is undefined behaviour.

My partial correction:

```
#include <stdio.h>
```

```
int main ( void )  
{long l = 420000;  
long *lp = &l;
```

```
short *sp = (short)*lp; /* ill-advised, but Let's Pretend */
```

It's a little hard to discern what is a mistake when the premise is that what we're doing is ill-advised or illegal. I do think the above line is miscast.

Arrrgggghhh!

I meant

```
short *sp = (short*)lp; /* ill-advised, but Let's Pretend */
```

```
/* Use of *sp */
```

Tries to read a short from where a long was stored. On normal 2s complement systems the part of the long that is read will depend on the endianness of the system.

The standard also says:

| An object shall have its stored value accessed only by an lvalue

<snip>

I.e. don't do what the above code does.

## Re: pointer q

I think if a fellow were serious, ie, trying not to do illegal or ill-advised things, and he wanted to "pick apart" something that the compiler has stored as a long at an address accessible to the source, he would go after it with the last of those options, unsigned chars, and make the necessary conversions to make zeroes and ones.

Yes, using unsigned char would be the correct way to pick it apart, and indeed there are sometimes good reasons for doing this.. You still have to watch out for endianness and be aware that different implementations have different length longs.

<snip>

There's something driving me bats right now. How does one complete this table:

```
long * | short | long ***
```

```
-----  
type is |||
```

```
-----  
type-specifier is |||
```

You should really use a fixed width font when doing things like tables. It may have looked formatted on your machine, but here it looks a right mess, and I'm not going to try all my fonts on the off chance one uses the same spacings as whatever font you used.

The type specifiers in the above are short and long. The types are long\*, short and long\*\*\*.

You might find a copy of the latest draft of the C standard useful, see [http://clc-wiki.net/wiki/C\\_standard](http://clc-wiki.net/wiki/C_standard) Section 6.7.2 is called "Type specifiers" and obviously tells you what they are!

—  
Flash Gordon, living in interesting times.  
Web site – <http://home.flash-gordon.me.uk/>  
comp.lang.c posting guidelines and intro:  
[http://clc-wiki.net/wiki/Intro\\_to\\_clc](http://clc-wiki.net/wiki/Intro_to_clc)

Inviato da X-Privat.Org – Registrazione gratuita <http://www.x-privat.org/join.php>