Subject: Re: - unsigned variables
Posted by Longtime Lurker on Fri, 24 Sep 2004 00:44:34 GMT
View Forum Message <> Reply to Message

Holger Fleckenstein wrote:

> A strange behavior in IDL occured to me.
>
> In C++ if I do:
> unsigned short x=1;
> printf("%d",-x);
> I get
> -1
> like I would expect.

Whereas:
printf("%u",-x);
4294967295

printf("%hu", -x);
65535

like I would expect

%d simply tells C to output the bit pattern stored in the argument(s) as if
it were a *signed* integer

Similarly
printf("%f", -x);
gives
2.102785
on my little-endian machine. The bit pattern is identical in all four cases
all that changes is the way it is interpreted.

>
> In IDL however:
> x=1U
> print, -x
> gives
> 65535
> So it basically treats it like I had done:
> print, uint(-1)

But -x is a UINT just like x
help, -x
<Expression> UINT = 65535

So IDL is doing exactly what you ask it to

Consider
print, -x, format = '(F)'
  65535.0000000000000000

>
> Does anybody have an explanation for this?
> Is this, because of a typecast before executing the print?
> (Can creat bugs, which are hard to localize.)

IDL outputs the true value of -x taking its type into account - any formatting is applied to this value. C[++] outputs the value of the bits interpreted according to the rules of the supplied conversion specifier.

The C behaviour has caught me out with code like

long long x = 1, y = 1;
printf("%d %d\n", x, y);
1 0

When I should have had

printf("%lld %lld\n", x, y);
1 1

IDL's behaviour seems preferable to me...

Paul

Subject: Re: - unsigned variables
Posted by JD Smith on Fri, 24 Sep 2004 00:53:18 GMT
View Forum Message <> Reply to Message

On Thu, 23 Sep 2004 15:56:16 -0700, Holger Fleckenstein wrote:

> A strange behavior in IDL occured to me.
>
> In C++ if I do:
> unsigned short x=1;
> printf("%d",-x);
> I get
> -1
> like I would expect.
>
> In IDL however:
> x=1U
> print, -x

> gives
> 65535
> So it basically treats it like I had done:
> print, uint(-1)
>
> Does anybody have an explanation for this? Is this, because of a typecast
> before executing the print? (Can creat bugs, which are hard to localize.)

Nope, it's because your print format is treating it as a signed long
integer. Try:

unsigned short x=1;
printf("%hu",-x);

which gives:

 65535

The difference is, IDL *knows* your integer is an unsigned short. C
doesn't know or care, and so is happy to print it however you like.
You can always change IDL's mind by explicitly casting it:

IDL> print,fix(-1U)
      -1

Note that short -1 and 65535 are actually represented by the exact
same bit pattern, namely:

1111111111111111

JD