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
