Subject: Re: strange GT and LT behavior
Posted by Dick Jackson on Mon, 20 Oct 2014 16:35:46 GMT
View Forum Message <> Reply to Message

On Monday, 20 October 2014 08:41:52 UTC-7, superchromix wrote:
> I've encountered a bizarre situation where IDL thinks that 0 is less than a negative number. Can anyone rationalize this? Is it really not ok to compare the value of an unsigned integer with a signed integer? Shouldn't the compiler handle this?

Hi Mark,

Forgive me for boiling down your test case:

```
IDL> 0ULL LT -100L
   1
```

I think what's happening is that, to compare a 64-bit (unsigned) type to a 32-bit (signed) type, the 32-bit value is converted to the "higher precedence" type, even though it will no longer be able to represent a negative number

From Help on "Language > Operators > Relational Operators"
=====
Each operand is promoted to the data type of the operand with the greatest precedence or potential precision. (See Data Type and Structure of Expressions for details.)
=====

Here's what was happening

```
IDL> 0ULL LT ULong64(-100L)
   1
IDL> help, -100L
<Expression>    LONG      =         -100
IDL> help, ULong64(-100L)
<Expression>    ULONG64   =   18446744073709551516
```

If we're pushing the limits here, this is possibly even more troublesome:

=====
Note: Signed and unsigned integers of a given width have the same precedence. In an expression involving a combination of such types, the result is given the type of the leftmost operand.
=====

This leads to the following curiosity, where it seems that, with the same "level" of precision (64 bits, but one signed and one unsigned), a < b and b < a:

```
IDL> 0ULL LT -100LL
   1
IDL> -100LL LT 0ULL
   1
```

I suppose the lesson here is, if there's a chance of comparing positive and negative values, be sure to convert both expressions to a signed type, or a float type.

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada - www.d-jackson.com