Subject: Re: bitwise operators in IDL?
Posted by John-David T. Smith on Tue, 22 May 2001 21:00:09 GMT
View Forum Message <> Reply to Message

William Thompson wrote:
>
> Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:
>
>> thompson@orpheus.nascom.nasa.gov (William Thompson) writes:
>
>>> "Rick Towler" <rtowler@u.washington.edu> writes:
>>>
>>>> Is there a built in function in IDL for the c++ bitwise operator "&" or is
>>>> this going to be the first DLM i write?
>>>
>>>> Rick Towler
>>>
>>>
>>> AND, OR, and NOT are bitwise operators.
>>>
>>> William Thompson
>
>> Which leads to some interesting confusion sometimes when they are used
>> as logical operators. Consider that:
>
>> 255 AND 'fe'xl is false, and
>> NOT 2 is true
>
> It's not the operators which are confusing here. They are doing exactly what
> they should. Consider the following:

That is does not constitute proof they are not confusing ;)

> So, even in a boolean sense, the operators are working correctly.
>
> What is confusing is that sometimes IDL considers all even numbers to be false,
> while other times only 0 is false. Generally, this depends on whether the
> number is an integer or floating point; integers use even/odd logic, while
> floating point numbers use zero/nonzero logic. For example, the result for the
> statement "if 2 then ..." is completely the opposite of "if 2.0 then ...". To
> mess things up even further, the function KEYWORD_SET() uses zero/nonzero logic
> even if the input is integer, and thus has the potential of changing the
> meaning of a boolean expression. For example, consider the result of
> KEYWORD_SET(NOT 1).

For integers the least significant bit is tested for positivity, hence
the even/oddness.

> The behavior for integers is necessary because of the bitwise nature of the
> operators, while floating point numbers are too complicated to permit such
> bitwise treatment. Thus, these operators are only bitwise for integers.

And long's, ulong's, long64's, ulong64's, BYTE, ...

> It would be nice if IDL had a boolean type that could only take the values
> True and False. Alternatively, one could define system variables !true and
> !false,
>
> DEFSYSV, '!TRUE', -1B
> DEFSYSV, '!FALSE', 0B
>
> and use those when setting variables meant to be boolean. (The -1B is the
> bitwise opposite of 0B.)

It would be really nice if IDL had any logical operators, other than implied in the ambiguous usage of bitwise op's for different types. Specifically, having a "short-circuiting" AND and OR operator set would be exceptionally useful.

How often do you find yourself doing something like:

if ptr_valid(a) AND *a ge 0 then...

only to find that it can't work, because AND always evaluates everything it operates on. Most decent languages offer short circuiting AND's (and OR's etc.), that stop as soon as the true solution is known. Here, if ptr_valid(a) is not true, there would be no need to continue to try to dereference 'a' (which generates an error), and this snippet would be correct.

I guess for now we're stuck with

```
if ptr_valid(a) then begin
  if *a ge 0 then begin...
```

Oh the tedium.

JD