Subject: Re: bitwise operators in IDL?
Posted by thompson on Mon, 21 May 2001 14:41:57 GMT
View Forum Message <> Reply to Message

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:

```
 IDL> if 255 then print,'true' else print,'false'
 true
 IDL> if 'fe'xl then print,'true' else print,'false'
 false
 IDL> if 255 and 'fe'xl then print,'true' else print,'false'
 false

 IDL> if 2 then print,'true' else print,'false'
 false
 IDL> if not 2 then print,'true' else print,'false'
 true
```

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).

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.

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.)

William Thompson