Subject: bitwise operators in IDL?
Posted by Rick Towler on Fri, 18 May 2001 21:34:09 GMT
View Forum Message <> Reply to Message

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

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

Subject: Re: bitwise operators in IDL?
Posted by marc schellens[1] on Wed, 23 May 2001 05:14:02 GMT
View Forum Message <> Reply to Message

> 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
To late for short circuitry.
Consider a case when the second function in the if case
has a side effect (e.g. modifying a global variable).
After once defining the language this way, to change it
would mean to introduce incopatibility.

But you can write:
if ptr_valid(a) then if *a eq b then begin
...

which looks a little bit nicer (IMHO).

cheers,
:-) marc

Subject: Re: bitwise operators in IDL?
Posted by John-David T. Smith on Wed, 23 May 2001 15:16:10 GMT
View Forum Message <> Reply to Message

Marc Schellens wrote:
>
>> 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
> To late for short circuitry.
> Consider a case when the second function in the if case
> has a side effect (e.g. modifying a global variable).
> After once defining the language this way, to change it
> would mean to introduce incopatibility.

Not if you introduce another operator set all together for short circuiting. People will use them increasingly, and AND,OR will go back to being used primarily for their bitwise function, as they should be. Sort of like C has "&&" and "&". I can't think of good replacement names (I assume RSI won't allow the sensible && and ||). Ideas? (ANDS? ORS?).

JD

Subject: Re: bitwise operators in IDL?
Posted by Craig Markwardt on Wed, 23 May 2001 18:19:20 GMT

JD Smith <jdsmith@astro.cornell.edu> writes:
> Marc Schellens wrote:
>> To late for short circuitry.
>> Consider a case when the second function in the if case
>> has a side effect (e.g. modifying a global variable).
>> After once defining the language this way, to change it
>> would mean to introduce incopatibility.
>
> Not if you introduce another operator set all together for short
> circuiting. People will use them increasingly, and AND,OR will go back
> to being used primarily for their bitwise function, as they should be.
> Sort of like C has "&&" and "&". I can't think of good replacement
> names (I assume RSI won't allow the sensible && and ||). Ideas? (ANDS?
> ORS?).
>
> JD

I like the idea of short-circuiting logical operators as well. Some time ago I proposed LAND and LOR (for Logical-AND and Logical-OR). Of course what do you do about backwards compatibility? Namely people who have written code with their own LAND or ANDS variables. Imagine this :-)

```
if LAND LAND 1 then ...
```

What would it mean?

Also, short-circuiting operators are not well-defined for vector operands. With vector operands it is possible in an element-by-element comparison for one element to evaluate true, while the other evaluates false. E.g.

```
if [1,1] LAND [0,1] then ...
```

So, both of these issues would at least have to be dealt with, and my guess is that the RSI folks will decide not to deal with it. Still, I think it's worth exploring.

Craig

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------

Subject: Re: bitwise operators in IDL?
Posted by John-David T. Smith on Wed, 23 May 2001 19:46:44 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> JD Smith <jdsmith@astro.cornell.edu> writes:
>> Marc Schellens wrote:
>>> To late for short circuitry.
>>> Consider a case when the second function in the if case
>>> has a side effect (e.g. modifying a global variable).
>>> After once defining the language this way, to change it
>>> would mean to introduce incopatibility.
>>
>> Not if you introduce another operator set all together for short
>> circuiting. People will use them increasingly, and AND,OR will go back
>> to being used primarily for their bitwise function, as they should be.
>> Sort of like C has "&&" and "&". I can't think of good replacement
>> names (I assume RSI won't allow the sensible && and ||). Ideas? (ANDS?
>> ORS?).
>>
>> JD
>
> I like the idea of short-circuiting logical operators as well. Some
> time ago I proposed LAND and LOR (for Logical-AND and Logical-OR). Of
> course what do you do about backwards compatibility? Namely people
> who have written code with their own LAND or ANDS variables. Imagine
> this :-)
>
> if LAND LAND 1 then ...
>
> What would it mean?
>
> Also, short-circuiting operators are not well-defined for vector
> operands. With vector operands it is possible in an
> element-by-element comparison for one element to evaluate true, while
> the other evaluates false. E.g.
>
> if [1,1] LAND [0,1] then ...
>
> So, both of these issues would at least have to be dealt with, and my
> guess is that the RSI folks will decide not to deal with it. Still, I
> think it's worth exploring.

Luckily, this is already illegal:

IDL> if [1,1] AND [0,1] then print, 'yay'
% Expression must be a scalar in this context: <INT Array[2]>.
% Execution halted at: $MAIN$

Only scalars allowed in if's. As long as the new lAND and lOR operators
are only allowed within if statements, no trouble arises.

For your example, proper use of total() is encouraged.

JD

Subject: Re: bitwise operators in IDL?
Posted by Stein Vidar Hagfors H[1] on Thu, 24 May 2001 14:14:16 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:

[...]
> Only scalars allowed in if's. As long as the new lAND and lOR operators
> are only allowed within if statements, no trouble arises.

I've previously suggested the Simula variants "AND THEN" and "OR
ELSE". Which have the benefits of not introducing any new keywords to
confuse with variables, and also reads in a meaningful way..(conveying
the order and sense of use). A bit wordy, but still.

--
---------- ----------
Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

NASA Goddard Space Flight Center, Email: shaugan@esa.nascom.nasa.gov
Mail Code 682.3, Bld. 26, Room G-1, Tel.: 1-301-286-9028/240-354-6066
Greenbelt, Maryland 20771, USA. Fax: 1-301-286-0264
---------- ----------
