
Subject: Testing for NODATA presence in a dataset
Posted by [Jonathan Greenberg](#) on Sat, 21 Dec 2002 22:00:39 GMT
[View Forum Message](#) <> [Reply to Message](#)

I'm having a problem testing for whether an entry in an array is NAN --
doing something like:

```
If (value EQ !VALUES.F_NAN) then begin
    print,'Not a number'
Endif else begin
    print,'Is a number!'
Endelse
```

Will always return 'Is a number', even if I set:
value = !VALUES.F_NAN

What's going wrong with this?

--j

Subject: Re: Testing for NODATA presence in a dataset
Posted by [tam](#) on Fri, 03 Jan 2003 14:10:44 GMT
[View Forum Message](#) <> [Reply to Message](#)

With regard to the discussion of the behavior of NaNs
especially on Windows machines:

I got a very nice, unofficial, response from someone at RSI indicating
that they agreed in principle the behavior should be IEEE 754
compliant, but that behavior is not currently supported
by the Microsoft compilers used in building IDL for Windows.
With these compilers 'proper' behavior would incur an unacceptable
performance penalty. We had a short dialog and in the process found out
that the latest versions of the Visual C++ compiler indicate
that Microsoft has made a change in the way it handles NaNs so
that arithmetic comparisons are IEEE 754 compliant. If this
turns out to be true there a good chance that in the future
the behavior of the Windows and Unix versions of IDL will be consistent.

Of course this is all non-official, but I was very impressed by
the responsiveness of RSI. It seems clear that they are both
aware of this issue and looking to see how it can be resolved
without compromising IDL performance.

Regards,
Tom McGlynn

Subject: Re: Testing for NODATA presence in a dataset
Posted by [George N. White III](#) on Fri, 03 Jan 2003 15:00:21 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Mon, 30 Dec 2002, Tom McGlynn wrote:

```
> David Fanning wrote:
>> Tom McGlynn (tam@lheapop.gsfc.nasa.gov) writes and
>> Bill Thompson confirms:
>>
>>> That doesn't distinguish NaN from the infinities.
>>> The standard trick in any language for looking for NaN's is
>>>
>>> if x ne x then begin
>>>   print,'This is a NaN'
>>> endif else ...
>>
>>
>> Humm, well, consider this little test in IDL 5.5 or 5.6
>> for Windows:
>>
>> IDL> a = [ 1.0, 2.0, !Values.F_NAN, 4.0, !Values.F_NAN ]
>> IDL> print, a
>>   1.00000   2.00000   NaN   4.00000   NaN
>> IDL> print, a(1)
>>   2.00000
>>
>> All well and good so far. Test the algorithm.
>>
>> IDL> if a(1) ne a(1) THEN print, 'NaN' ELSE print, 'Number'
>>   Number
>>
>> Perfect. Working fine. Now test NaN.
>>
>> IDL> print, a(2)
>>   NaN
>> IDL> if a(2) ne a(2) THEN print, 'NaN' ELSE print, 'Number'
>>   Number
>>   % Program caused arithmetic error: Floating illegal operand
>>
>> Oh, oh. What's up with that? And a floating illegal operand to
>> boot. :-(
>>
>> How about the array in general?
>>
>> IDL> print, array ne array
>>   0 0 0 0 0
>>   % Program caused arithmetic error: Floating illegal operand
>>
```

>> Humm. I presume you guys have a reason for thinking
>> like you do. Any insights?
>>
>> Cheers,
>>
>> David
>
> Just to follow up on Bill's message.... I did warn in my first message
> that interpreters had been known to screw up this comparison, but I
> believe the behaviour you see above is clearly non-compliant with
> the IEEE 754 floating point standard. I almost never run IDL
> under Windows, but I'd call this a bug -- though I daresay RSI
> will call it a feature.

Although everyone knows what you mean by the "IEEE 754:1985 floating point standard", I think the current international standard is "IEC 60559:1989, Binary Floating-Point Arithmetic for Microprocessor Systems" (previously designated IEC 559:1989).

> Using IDL 5.2 under Linux I have:
>
> IDL> a=sqrt(-1)
> %Program caused arithmetic error. Floating illegal operand.
> IDL> print, a
> -NaN
> IDL> print a ne a
 print, a ne a
> 1
> IDL> z=[0,0,a,a,0]
> IDL> print, z ne z
> 0 0 1 1 0

I get the same for IDL 5.5 on SGI Irix.

> I believe this to be 'correct' behavior but it appears that
> it is not universally implemented this way within IDL. Of
> course IDL has been implemented on non-IEEE machines (e.g., VAX)
> and so completely consistent behavior may be impossible.

Many compilers and runtimes have options to tweak IEC 60559:1989 behaviour for performance reasons. You have to tell some compilers that you want run-time evaluation of expressions like 'x ne x'. I suppose, now that interpreters are moving to just-in-time compilation, we will need similar options for interpreted languages.

If you need consistent cross-platform support for IEC 60559:1989 formats and also missing-value support, the open source R package is handy (<http://www.ci.tuwien.ac.at/~hornik/R/>).

--

George N. White III <gnw3@acm.org> Bedford Institute of Oceanography
