Subject: Re: odd behaviour from array_equal() with NaN, Inf values
Posted by Craig Markwardt on Mon, 23 Jan 2012 05:46:12 GMT
View Forum Message <> Reply to Message

On Jan 22, 11:42 pm, wallabadah <write.to.wpow...@gmail.com> wrote:
> I've come across the following behaviour while tracking down odd behaviour from NaN and Inf values. I'm trying to use array_equal() to check that arrays contain the same content - but it bombs when the arrays contain NaN. I've tracked it down to the process of copying an array containing NaN values to a new variable. For example:
>
> IDL> a = findgen(5)
> IDL> a[1] = !values.f_nan
> IDL> print, array_equal(a, a)
> 1
> IDL> b = a
> IDL> print, array_equal(a, b)
> 0
> IDL> print, array_equal(b, b)
> 1
>
> Repeating the process with !values.f_infinity behaves as expected:
>
> IDL> a = findgen(5)
> IDL> a[1] = !values.f_infinity
> IDL> print, array_equal(a, a)
> 1
> IDL> b = a
> IDL> print, array_equal(a, b)
> 1
> IDL> print, array_equal(b, b)
> 1
>
> Is this a bug in make_array() or some artifact of how values are copied to new variables?? Is it reproducible on different platforms (I'm on Mac OS X, IDL 8.1).

Greetings--

I think it's a subtle bug in ARRAY_EQUAL(). ARRAY_EQUAL(A,A) *should*
return 0 for your case.

It is a property of NAN that it can never equal itself so - perhaps
surprisingly -
(A[1] EQ A[1]) is false. Try it yourself!

I suspect that the IDL folks decided to put an special case
optimization in to the code which always returned true when the same
variable is passed in both parameter positions. Unfortunately, this
fails when one more more elements is NAN.