Subject: Re: Numbers from nowhere?
Posted by Norbert Hahn on Thu, 21 Feb 2008 16:29:03 GMT
View Forum Message <> Reply to Message

Sven Utcke <utcke+news@informatik.uni-hamburg.de> wrote:

> David Fanning <news@dfanning.com> writes:
>
>> elwood writes:
>>
>>> But my question is more pointed: if you assign x=3.3 and you know
>>> apriori that the floating point data type will not have enough bits
>>> to store this number precisely, why does "print" show this number
>>> as 3.3?
>>
>> I presume it is because whatever number *is* stored, when rounded to
>> the 7-8 significant figures a float can accurately represent, comes
>> out to 3.300000.
>
> What number _is_ stored, actually? Assuming we are talking ieee, we
> have one bit for the sign, 8 for the exponent, and 23 for the
> mantissa. So what is 3.3?
>
> 3 = 11 = 1.1 * 2^1
> 0.3 = 0.0100110011001100110011001100110011001100...

Note that the above pattern repeats forever and *must* be truncated
in the real world (of computers) as much as 1/3 is not 0.3333 in
decimal.

[snip]
>
> S | Exp + 127 | Mantissa without leading 1
> 0 | 1000000 | 1010011 00110011 00110011
>
> which, if we recombine it, turns out to be 3.299999952316284179687 5

... because we only store 24 binary mantissa digits. The formatting
routine in IDL most likely works with double precission and hence
appends binary zeroes rather than continuing to repeat the pattern 0011.
This results in a decimal number being slightly less than 3.30000000.

The problem arises from the conversion between binary to decimal numbers.
It is not a problem of IDL but a problem of working with numbers of
finite length.

Sven, thanks for you elaboration!