Subject: Re: Setting values to NaN
Posted by Michael Wallace on Fri, 25 Jun 2004 16:30:30 GMT
View Forum Message <> Reply to Message

>> OK I spotted it. F_NAN and D_NAN apply to floats.
>>
>> So is there an alternative I can use with integers?
>
>
> How would you define a bit pattern for an invalid integer that's not a
> valid integer?, as opposed to some user-set value (like -99 or or -32767
> or -2147483647L etc.) ?
>
> You need to define some user-set value (like -99 or or -32767 or
> -2147483647L etc.) based on what you knwo about your data.

Just to expand on Paul's answer a little, computers work with floating point numbers very differently than they work with integers. For floating point numbers certain bit patterns are reserved for NaN, positive infinity and negative infinity. If you want to see the gory details, look up the IEEE 754 spec for floating point numbers.

The integer specification does not set aside special bit patterns for NaN or the infinities. Therefore, if you want create a NaN for integers, you have to create an arbitrary definition based on your data. Typically, a good rule of thumb is to use the smallest or largest possible integers as these will most likely be outside your data set. The problem is that since your definition will be arbitrary you need to make sure that if others use your data you indicate that there is a number in the data which needs to be interpreted as NaN.

For example, let's say I have a data set of all positive numbers and I use -1 to indicated NaN. If someone else comes along and doesn't know that -1 represents NaN, they will do their processing using the -1 values and potentially get results that are very far off. I have run into this very thing before. Someone used -999 to represent that no data was present, but didn't document it. It took me the major part of an afternoon to figure out why my code was failing so badly.

-Mike