Ruby - Bug #5884

Float::NAN and 0.0/0.0 is represented differently when packed with 'g'

01/12/2012 06:50 AM - hasari (Hiro Asari)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.0.0dev (2011-12-31 trunk 34165) [x86_64-darwin11.2.0]	**Backport:**	

Description

```
$ ruby2.0 -e 'p [Float::NAN].pack("g")'
"\x7F\xC0\x00\x00"

$ ruby2.0 -e 'p [0.0/0.0].pack("g")'
"\xFF\xC0\x00\x00"
```

It would be nice to have Float::NAN and 0.0/0.0 behave identically in this regard.

History

#1 - 01/12/2012 06:54 AM - brixen (Brian Shirai)

While it may make some sense that Float::NAN is the quite NaN (if I understand IEEE 754 correctly), really the only way to get NaN as a value in Ruby prior to Float::NAN was to compute 0.0/0.0, which I would expect to result in signaling NaN. Also, since both NaNs roundtrip through (un)pack('g') as a value on which #nan? returns true, most code should be fine.

Despite all this, I think it would be least confusing if Float::NAN were actually the value computed by 0.0/0.0.

Thanks,
Brian

#2 - 01/12/2012 07:46 AM - naruse (Yui NARUSE)

I can't understand why it is nice and least confusing if 0.0/0.0 equals Float::NAN as a binary.
Could you show a use case or reasonable logic?

#3 - 01/12/2012 08:24 AM - hasari (Hiro Asari)

Besides it being status quo, what is the rationale behind not having Float::NAN and 0.0/0.0 act identically?

As Brian mentioned, until the introduction of Float::NAN, 0.0/0.0 acted as a means of getting IEEE 754 NaN. If the intent for Float::NAN is to act as *the* NaN, then it seems reasonable to expect these objects to behave the same way in all manners possible.

Of course, if the purpose of Float::NAN is *not* to act as an identical object as 0.0/0.0, then this argument will not hold water.

I don't have a personal stake in this matter; I just want a clarification. Thank you.

#4 - 01/12/2012 09:23 AM - matz (Yukihiro Matsumoto)

Hi,

As far as I understand, the concept of "the NaN" itself is against the
definition of NaN in the IEEE 754 that defines NaN as set of floating
point values. Correct me if I am wrong.

```
                    matz.
```

In message "Re: [ruby-core:42069] [ruby-trunk - Bug #5884] Float::NAN and 0.0/0.0 is represented differently when packed with 'g'"
on Thu, 12 Jan 2012 08:24:29 +0900, Hiro Asari asari.ruby@gmail.com writes:
|
|
|Issue #5884 has been updated by Hiro Asari.
|
|
|Besides it being status quo, what is the rationale behind not having Float::NAN and 0.0/0.0 act identically?
|
|As Brian mentioned, until the introduction of Float::NAN, 0.0/0.0 acted as a means of getting IEEE 754 NaN. If the intent for Float::NAN is to act as

the NaN, then it seems reasonable to expect these objects to behave the same way in all manners possible.
|
|Of course, if the purpose of Float::NAN is *not* to act as an identical object as 0.0/0.0, then this argument will not hold water.
|

I don't have a personal stake in this manner; I just want a clarification. Thank you.
Bug #5884: Float::NAN and 0.0/0.0 is represented differently when packed with 'g'
https://bugs.ruby-lang.org/issues/5884

|
|Author: Hiro Asari
|Status: Open
|Priority: Normal
|Assignee:
|Category:
|Target version:
|ruby -v: ruby 2.0.0dev (2011-12-31 trunk 34165) [x86_64-darwin11.2.0]
|
|
|$ ruby2.0 -e 'p [Float::NAN].pack("g")'
|"\x7F\xC0\x00\x00"
|
|$ ruby2.0 -e 'p [0.0/0.0].pack("g")'
|"\xFF\xC0\x00\x00"
|
|It would be nice to have Float::NAN and 0.0/0.0 behave identically in this regard.
|
|
|--
|http://bugs.ruby-lang.org/

#5 - 01/12/2012 09:42 AM - naruse (Yui NARUSE)

Hiro Asari wrote:

> Besides it being status quo, what is the rationale behind not having Float::NAN and 0.0/0.0 act identically?

- I thought it should be quiet NaN
- no reason to choose "\xFF\xC0\x00\x00" there is many NaNs and platform dependent
- the result of 0.0/0.0 may differ on each call

> As Brian mentioned, until the introduction of Float::NAN, 0.0/0.0 acted as a means of getting IEEE 754 NaN. If the intent for Float::NAN is to act as *the* NaN, then it seems reasonable to expect these objects to behave the same way in all manners possible.

Show concreate use case.
I don't think pack/unpack is the use case.
See also the result of Float::NAN == Float::NAN.

Yukihiro Matsumoto wrote:

> As far as I understand, the concept of "the NaN" itself is against the
> definition of NaN in the IEEE 754 that defines NaN as set of floating
> point values. Correct me if I am wrong.

Yes, there is many NaNs.

#6 - 01/12/2012 10:14 AM - hasari (Hiro Asari)

I am not presenting a real use case. That is not my intent. My intent is to clarify the design of Float::NAN, especially in relation to 0.0/0.0. Since there is no way to distinguish the two without it, it seems to me that Array#pack is as good a use case as any. It is useful to the extent that we can distinguish the two, now that it is established that they can (and perhaps *should*) be different.

I see that my assumption that these objects should behave the same was flat out misguided.

It is acknowledged that there are many NaN's. As there is no way to distinguish these multiple NaN values that are passed to Array#pack, then, am I correct to conclude that [Float::NAN].pack("g"), as well as [0.0/0.0].pack("g"), is indeterminate? Also, am I correct to conclude that the value generated on one machine may not be identical to the one that is generated on another? (I understand that this is mostly academic; the two different values make a round trip correctly in and out of their respective packed counterpart, one way or another.)

#7 - 01/12/2012 11:14 AM - naruse (Yui NARUSE)

Yui NARUSE wrote:

> Hiro Asari wrote:
>
> > Besides it being status quo, what is the rationale behind not having Float::NAN and 0.0/0.0 act identically?
>
> - I thought it should be quiet NaN
> - no reason to choose "\xFF\xC0\x00\x00" there is many NaNs and platform dependent
> - the result of 0.0/0.0 may differ on each call

I must add one more reason; the NaN seems not what I defined.
The NaN is provided by the system's math.h.

And additional to say, you should use unpack('G') because it is double.

Hiro Asari wrote:

> I am not presenting a real use case. That is not my intent. My intent is to clarify the design of Float::NAN, especially in relation to 0.0/0.0. Since there is no way to distinguish the two without it, it seems to me that Array#pack is as good a use case as any. It is useful to the extent that we can distinguish the two, now that it is established that they can (and perhaps *should*) be different.

The design of Float::NAN is to provide a NaN.
It may equal to a result of 0.0/0.0 as a binary and maybe not.
The reason why there is no straight way to distinguish the two is people shouldn't need it.

> I see that my assumption that these objects should behave the same was flat out misguided.
>
> It is acknowledged that there are many NaN's. As there is no way to distinguish these multiple NaN values that are passed to Array#pack,

If you know the environment's floating point number system supports IEEE 754 and its endian,
you can know what binary is NaN.

> then, am I correct to conclude that [Float::NAN].pack("g"), as well as [0.0/0.0].pack("g"), is indeterminate?

Yes, its binary expression is out of the spec.
The one accurate thing is the binary is a NaN on the environment.

> Also, am I correct to conclude that the value generated on one machine may not be identical to the one that is generated on another?
>
> (I understand that this is mostly academic; the two different values make a round trip correctly in and out of their respective packed counterpart, one way or another.)

Yes, of course.
There is non IEEE 754 systems.

for example VAX is not an IEEE 754 system and doesn't have NaN. (Ruby 1.9 or later doesn't support it)

And IEEE 754/other specs doesn't say what NaN should be returned.

Anyway if both system is IEEE 754 system, a NaN on a system is also a NaN on another system.

#8 - 01/20/2012 10:23 AM - naruse (Yui NARUSE)

- Status changed from Open to Rejected