Ruby 1.8 - Backport #3273

Float string conversion

05/11/2010 02:12 PM - marcandre (Marc-Andre Lafortune)

Status:	Closed	
Priority:	Normal	
Assignee:		

Description

=begin
For any float f, the two following conditions should hold:
(1) f.to_s.to_f == f (round trips)
(2) f.to_s.chop.to_f != f (minimal)

The second condition is a simplification; if the string representation is in scientific notation, than the character to remove would be the one just before the "e". Also, if the string representation ends with ".0", then it is minimal.

Currently, the first condition fails in Ruby 1.8, and the second fails in Ruby 1.9

$ ruby18dev -ve 'f = 0.21611564636388508; puts f.to_s.to_f == f'
ruby 1.8.8dev (2010-05-11) [i386-darwin10.3.0]
false

$ rubydev -ve 'f = 0.56; puts f.to_s.chop.to_f != f'
ruby 1.9.3dev (2010-05-11 trunk 27730) [x86_64-darwin10.3.0]
false

Note that this implies that Ruby 1.8 and 1.9 do not output the same string representation for either of these two floats.

The conversion algorithm currently checks two precisions. In Ruby 1.9, it tries 16 digits and if that's not enough it then uses 17. In 1.8, it's the same but with 15 and 16.

The fact is that 17 can be necessary (e.g. 0.21611564636388508 is not equal to either 0.2161156463638851 or 0.2161156463638850) and 16 can be too much (e.g. 0.5600000000000001 == 0.56), so three precisions must be checked.

The following patch fixes this issue for trunk (although it can probably be made nicer and/or faster).

Let me know if there are any objections to fixing both the 1.9 and 1.8 lines.

diff --git a/numeric.c b/numeric.c
index f2c8c13..442b069 100644
--- a/numeric.c
+++ b/numeric.c
@@ -569,7 +569,8 @@ flo_to_s(VALUE flt)
else if (isnan(value))
return rb_usascii_str_new2("NaN");

-# define FLOFMT(buf, size, fmt, prec, val) snprintf(buf, size, fmt, prec, val), \
+# define FLOFMT(buf, size, fmt, prec, val) snprintf(buf, size, fmt, prec-1, val), \

- (void)((atof(buf) == val) || snprintf(buf, size, fmt, (prec), val)), \
  (void)((atof(buf) == val) || snprintf(buf, size, fmt, (prec)+1, val))

  FLOFMT(buf, sizeof(buf), "%#.*g", float_dig, value); /* ensure to print decimal point */
  =end

Related issues:

Related to Ruby trunk - Bug #4656: Float#to_s can produce too many digits	**Rejected**	**05/09/2011**

History

#1 - 05/11/2010 08:30 PM - mame (Yusuke Endoh)

=begin
Hi,

2010/5/11 Marc-Andre Lafortune redmine@ruby-lang.org:

> For any float f, the two following conditions should hold:
> (1) f.to_s.to_f == f (round trips)
> (2) f.to_s.chop.to_f != f (minimal)
>
> The second condition is a simplification; if the string representation is in scientific notation, than the character to remove would be the one just before the "e". Also, if the string representation ends with ".0", then it is minimal.

Neat formulation.

> The fact is that 17 can be necessary (e.g. 0.21611564636388508 is not equal to either 0.2161156463638851 or 0.2161156463638850) and 16 can be too much (e.g. 0.5600000000000001 == 0.56), so three precisions must be checked.

Interesting. 18 cannot be necessary, and 15 cannot be too much,
so it is enough to try 15, 16 and 17 digits. Right?

> The following patch fixes this issue for trunk (although it can probably be made nicer and/or faster).

Agreed.

--
Yusuke Endoh mame@tsg.ne.jp

=end

#2 - 05/12/2010 02:51 AM - kstephens (Kurt Stephens)

=begin
Should Float#to_s cache its result since Floats are immutable values and Float#to_s is fairly complex/expensive?
=end

#3 - 05/12/2010 06:45 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:30165] [Bug #3273] Float string conversion"
on Wed, 12 May 2010 02:51:37 +0900, Kurt Stephens redmine@ruby-lang.org writes:

|Should Float#to_s cache its result since Floats are immutable values and the Float#to_s is fairly complex/expensive?

First, Maybe. it should. But it's not a bug at least. Move to
feature-request. Second, we need to prove the effectiveness of the
caching in real-world use-case (is it really a bottleneck?), before
sacrifying space efficiency.

```
                          matz.
```

=end

#4 - 05/12/2010 11:07 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r27745.
Marc-Andre, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end

#5 - 05/12/2010 11:42 PM - RickDeNatale (Rick DeNatale)

=begin
On Tue, May 11, 2010 at 5:44 PM, Yukihiro Matsumoto matz@ruby-lang.org wrote:

> Hi,
>
> In message "Re: [ruby-core:30165] [Bug #3273] Float string conversion"
> on Wed, 12 May 2010 02:51:37 +0900, Kurt Stephens redmine@ruby-lang.org writes:
>
> |Should Float#to_s cache its result since Floats are immutable values and the Float#to_s is fairly complex/expensive?
>
> First, Maybe. it should. But it's not a bug at least. Move to
> feature-request. Second, we need to prove the effectiveness of the
> caching in real-world use-case (is it really a bottleneck?), before
> sacrifying space efficiency.

I'm not even sure how this would work.

Although floats ARE immutable they aren't interned, so it's unlikely that two computed float values which are equal will be the same object, so cacheing to_s for an instance of Float isn't likely to gain much. Caching through some kind of hash would seem to be the only other option, and that may be on the same order of performance as just recomputing the string, not sure just a guess. It would also hamper GC by keeping references, to the strings and float keys.

I'd also guess that this if this IS a problem it's only a problem for a small class of applications, and would be better addressed by those applications.

--
Rick DeNatale

Blog: http://talklikeaduck.denhaven2.com/
Github: http://github.com/rubyredrick
Twitter: RickDeNatale (Rick DeNatale)
WWR: http://www.workingwithrails.com/person/9021-rick-denatale
LinkedIn: http://www.linkedin.com/in/rickdenatale

=end

#6 - 05/30/2010 09:08 PM - marcandre (Marc-Andre Lafortune)

- Category set to core

- Status changed from Closed to Open

- Assignee deleted (marcandre (Marc-Andre Lafortune))

- Target version set to Ruby 1.8.8

=begin

=end

#7 - 06/14/2012 11:30 AM - marcandre (Marc-Andre Lafortune)

- Description updated

- Status changed from Open to Closed