Ruby master - Bug #14729

Float("long_invalid_string") fails to throw an exception

05/01/2018 02:37 PM - samiam (Sam Napolitano)

**Status:** Closed  
**Priority:** Normal  
**Assignee:**  
**Target version:**

ruby -v:  
ruby 2.6.0dev (2018-04-29 trunk 63298) [x86_64-darwin16]  
**Backport:** 2.3: REQUIRED, 2.4: DONE, 2.5: DONE

**Description**

When Float() is used to convert a string into a float, invalid characters in the string throw an error.

But when a really long string is passed to Float(), invalid characters exceeding the size of the internal C buffer are ignored and no error is thrown.

This behavior is inconsistent; underscores are verified throughout the entire string so why not look for other invalid characters?

I have a weak patch but would prefer to see what the developers think of this bug before I post it. Should Float() accept any size string or limit it?

**Code details:**

The code in question is object.c:rb_cstr_to_dbl_raise().

https://bugs.ruby-lang.org/projects/ruby-trunk/repository/entry/object.c#L3232

Specifically the buffer limit is usually 70-1 digits. For reference, \(2^{64}\) is 20 digits so this may be an academic exercise.

https://bugs.ruby-lang.org/projects/ruby-trunk/repository/entry/object.c#L3271

As an aside, I believe the last check on errno in the function is unnecessary. Errno should be examined immediately after a system call, which it is, so it's unclear why it's checked again at the end of the function.

https://bugs.ruby-lang.org/projects/ruby-trunk/repository/entry/object.c#L3307

The following code demonstrates the issue with some additional comments.

```ruby
#!/usr/bin/env ruby
require 'test/unit'
require 'test/unit/assertions'
include Test::Unit::Assertions

class TestFloat < Test::Unit::TestCase

  def test_strtod_ok
    assert_raise(ArgumentError) { Float('1' * (BUF_SIZE-1) + 'a') }
  end

  def test_strtod_no_error
    assert_equal(1.111111111111112e+68, Float('1' * BUF_SIZE + 'a is ignored'))
  end

end
```

03/20/2020 1/4
# case 3: entire string is scanned for underscores
# Result: when '_' is found in string, prev char is checked and MUST be ISDIGIT
# or error is thrown by rb_cstr_to_dbl_raise not strtod.
def test_underscores_checked_whole_string
  assert_raise(ArgumentError){
    Float('1' * BUF_SIZE + '234_56a_890')
  }
end

# case 4: the bug - should ruby scan entire string and detect invalid chars
# just like it does for invalid underscores so this test should pass?
# Result: no exception raised (currently)
def test_check_whole_string_for_invalid_chars
  assert_raise(ArgumentError){
    Float('1' * BUF_SIZE + 'a')
  }
end
end

Related issues:
Associated revisions
Revision 955849c1 - 05/04/2018 06:12 AM - nobu (Nobuyoshi Nakada)
object.c: raise on long invalid float string
  object.c (rb_cstr_to_db_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@63334 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 63334 - 05/04/2018 06:12 AM - nobu (Nobuyoshi Nakada)
object.c: raise on long invalid float string
  object.c (rb_cstr_to_db_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]

Revision 63334 - 05/04/2018 06:12 AM - nobu (Nobuyoshi Nakada)
object.c: raise on long invalid float string
  object.c (rb_cstr_to_db_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]

Revision a88ab94e - 01/20/2019 09:17 AM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 63334: [Backport #14729]
  object.c: raise on long invalid float string
  * object.c (rb_cstr_to_db_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_5@66880 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66880 - 01/20/2019 09:17 AM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 63334: [Backport #14729]
  object.c: raise on long invalid float string
  * object.c (rb_cstr_to_db_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]

Revision 4c2f9198 - 01/31/2019 10:47 AM - usa (Usaku NAKAMURA)
merge revision(s) 63334: [Backport #14729]
  object.c: raise on long invalid float string
  * object.c (rb_cstr_to_db_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@66962 b2dd03c8-39d4-4d8f-98ff-823fe69b080e
merge revision(s) 63334: [Backport #14729]

object.c: raise on long invalid float string

* object.c (rb_cstr_to_dbl_raise): check long invalid float string more precisely when truncating insignificant part.
[ruby-core:86800] [Bug #14729]

History

#1 - 05/01/2018 03:25 PM - shevegen (Robert A. Heiler)

Should Float() accept any size string or limit it?

I think the method Float() should behave in the same way as .to_f on a String in this regards. So if .to_f is fine to be used on any sized string, the Float() method should be fine on that string as well. In this context I assume that the string has only valid numbers such as 1 2 3 etc.. - obviously Float() will raise an error for non such characters used whereas .to_f is fine with any character in a given string.

#2 - 05/01/2018 03:27 PM - shevegen (Robert A. Heiler)

This behavior is inconsistent; underscores are verified throughout the entire string so why not look for other invalid characters?

I think that particular reasoning is incorrect in regards to underscores, as underscores are valid in both variants aka simply ignored, as far as I can see it:

"123_456".to_f  #=> 123456.0
Float("123_456")  #=> 123456.0

I have not checked on the size restriction yet though.

#3 - 05/02/2018 01:32 PM - nobu (Nobuyoshi Nakada)

- Related to Bug #14731: Float() ignores exponent in long string added

#4 - 05/02/2018 03:16 PM - samiam (Sam Napolitano)

shevegen (Robert A. Heiler) wrote:

I think that particular reasoning is incorrect in regards to underscores, as underscores are valid in both variants aka simply ignored, as far as I can see it:

"123_456".to_f  #=> 123456.0
Float("123_456")  #=> 123456.0

I have not checked on the size restriction yet though.

Underscores are treated differently with to_f and Float - they are not just stripped out of the string.

to_f never throws an exception. It just stops processing at invalid chars.

Float() throws an exception on invalid underscores regardless of string length.

irb(main):0151:> "1_000__".to_f
=> 1000.0
irb(main):0152:> Float("1_000__")
ArgumentError: invalid value for Float(): "1_000__"

#5 - 05/04/2018 06:12 AM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed

Applied in changeset trunk|r63334.

object.c: raise on long invalid float string

- object.c (rb_cstr_to_dbl_raise): check long invalid float string more precisely when truncating insignificant part. [ruby-core:86800] [Bug #14729]

#6 - 05/04/2018 10:19 AM - nobu (Nobuyoshi Nakada)
- Backport changed from 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN to 2.3: REQUIRED, 2.4: REQUIRED, 2.5: REQUIRED

#7 - 01/20/2019 09:18 AM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.3: REQUIRED, 2.4: REQUIRED, 2.5: REQUIRED to 2.3: REQUIRED, 2.4: REQUIRED, 2.5: REQUIRED, 2.5: DONE

ruby_2_5 r66880 merged revision(s) 63334.

#8 - 01/31/2019 10:48 AM - usa (Usaku NAKAMURA)
- Backport changed from 2.3: REQUIRED, 2.4: REQUIRED, 2.5: DONE to 2.3: REQUIRED, 2.4: DONE, 2.5: DONE

ruby_2_4 r66962 merged revision(s) 63334.