Ruby master - Bug #11953

ThreadError in 2.3 on code that works on 2.2.4

01/05/2016 07:01 PM - claytonflesher (Clayton Flesher)

Status:	Rejected		
Priority:	Normal		
Assignee:	marcandre (Marc-Andre Lafortune)		
Target version:			
ruby -v:	2.3.0	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

This is my first bug reporting, so I'm sorry if this isn't exactly correct format. I'll update my report accordingly if pointed in the right direction.

I ran into what appears to be a bug while doing a quiz.

I've included the breaking code, the test suite and the description of the quiz as attachments to this bug report.

The following code works fine on Ruby 2.2.4.

```
class Prime
  def self.nth(num)
    error(num)
    primes = build_primes(num)
    primes.last
  end

  private

  def build_primes(num)
    primes = [2]
    index  = 3
    until primes.size == num
      if example?(index, primes)
        primes << index
      end
      index += 1
    end
    primes
  end

  def error(num)
    unless num > 0
      raise ArgumentError
    end
  end

  def example?(index, primes)
    primes.each do |prime|
      if index % prime == 0
        return false
      end
    end
    true
  end
end
```

It raises this error in Ruby 2.3.0

```
ThreadError: deadlock; recursive locking
    /home/clayton/.rubies/ruby-2.3.0/lib/ruby/2.3.0/singleton.rb:140:in `synchronize'
    /home/clayton/.rubies/ruby-2.3.0/lib/ruby/2.3.0/singleton.rb:140:in `instance'
```

```
    /home/clayton/.rubies/ruby-2.3.0/lib/ruby/2.3.0/singleton.rb:142:in `block in instance'
    /home/clayton/.rubies/ruby-2.3.0/lib/ruby/2.3.0/singleton.rb:140:in `synchronize'
    /home/clayton/.rubies/ruby-2.3.0/lib/ruby/2.3.0/singleton.rb:140:in `instance'
    /home/clayton/exercism/ruby/nth-prime/nth_prime.rb:3:in `nth'
    nth_prime_test.rb:28:in `test_first'
```

If this is expected behavior, I'm sorry in advance for wasting your time.

History

#1 - 01/06/2016 03:49 AM - JEG2 (James Gray)

This is the minimal reproduction I could come up with:

```
require "forwardable"
require "singleton"

class Surprise
  include Singleton

  class << self
    extend Forwardable

    def method_added(method)
      (class << self; self; end).def_delegator(:instance, method)
    end
  end

  define_method(:new) do |*|
    fail "Oops"
  end

  def self.do_something
    wrong_level
  end

  private

  def wrong_level
  end
end

Surprise.do_something
```

#2 - 01/06/2016 04:08 AM - JEG2 (James Gray)

My recreation may not be useful though as it has the same error on Ruby 2.2.4. The longer code does not.

#3 - 05/26/2016 09:17 PM - tmornini (Tom Mornini)

James Gray wrote:

> The following code works fine on Ruby 2.2.4
> It raises this error in Ruby 2.3.0
> ThreadError: deadlock; recursive locking

We just saw this same error -- apparently out of nowhere -- on very stable code that utilizes Celluloid while under significant load.

We've never seen this error previously but cannot yet say that it runs fine on 2.2.4 -- though we do have a strong belief that it runs well under 1.9.3.

We'll try it under 2.2.4 and report back.

#4 - 05/27/2016 07:01 AM - nobu (Nobuyoshi Nakada)

- Description updated

With 1.9..trunk, I got same behaviors with both examples, no error with Clayton's and ThreadError with James's.

#5 - 05/27/2016 06:58 PM - tmornini (Tom Mornini)

Tom Mornini wrote:

> We'll try it under 2.2.4 and report back.

We actually tested against 2.2.5, and we're seeing identical behavior to 2.3.1.

I'm going to file an issue against Celluloid on Github and reference this ticket.

#6 - 06/07/2016 04:34 PM - noahgibbs (Noah Gibbs)

The error is occurring in Singleton's instance method, which is mutex-protected.

Using the supplied example code, git-bisect points to this Ruby commit as the problem:

```
commit d2487ed47587ec1cd1b456068e0af3ea0b39596d
Author: marcandre <marcandre@b2dd03c8-39d4-4d8f-98ff-823fe69b080e>
Date:   Fri May 22 13:36:39 2015 +0000

    * lib/prime.rb: Remove obsolete Prime.new
      patch by Ajay Kumar. [Fixes GH-891]

    git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@50604 b2dd03c8-39d4-4d8f-98ff-823fe69b080e
```

And that's where Prime switched from using a @the_instance member variable to using Singleton to track its instance.

So it looks like this code never worked with Prime as a Singleton.

#7 - 06/07/2016 04:53 PM - noahgibbs (Noah Gibbs)

Okay, Prime removed its .new() method. Singleton appears to depend on .new(). It's getting the recursive synch because the deprecated new() is calling instance(), so it would recurse infinitely. But instead, it tries to grab its own mutex and dies that way.

#8 - 06/08/2016 05:58 AM - marcandre (Marc-Andre Lafortune)

So, a minimal example is:

```
require 'prime'
class Prime
  def new
  end
end

#  access instance, directly or indirectly via a delegated class method
Prime.prime?(42) # ThreadError: deadlock
```

It's easy to circumvent by calling Prime.instance before defining new, either directly or indirectly (via any delegated class method like Prime.prime?).

Maybe the prime library should instantiate the singleton on load? That would resolve the issue at hand.

I am a bit baffled by what is the intent behind defining an instance method called new though...

#9 - 06/08/2016 08:59 PM - noahgibbs (Noah Gibbs)

Marc-Andre: this is happening in code that doesn't define new *or* call instance() directly -- the supplied sample code doesn't do either of those. Instead, methods called on Prime are forwarded to its instance. If the instance doesn't exist, they try to create it using new. Prime.new doesn't work -- it calls .instance() again, recursing and trying to grab the mutex again.

Just calling Prime.instance() first wouldn't help. You'd have to set the instance variable of Prime (the one Singleton uses) manually in some way to avoid calling Prime.new, which does the wrong thing.

Part of the problem is that Prime.new is an old deprecated interface that users used to use and now shouldn't.

The test code is just calling Prime.nth() and failing.

#10 - 06/08/2016 11:18 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to marcandre (Marc-Andre Lafortune)

Hi.

> this is happening in code that doesn't define new

I'm sorry if I'm missing something, but could you please provide such code?

Lines 12-15 of nth_prime_test.rb defines an instance method called new, and James also defines new in his code.

> Just calling Prime.instance() first wouldn't help

Did you actually try? Inserting Prime.instance in nth_prime_test.rb after the require doesn't produce the error

> Part of the problem is that Prime.new is an old deprecated interface that users used to use and now shouldn't

FWIW, Prime.new was deprecated in Ruby 1.9.0, more than 7 years ago. Are there examples of users still calling it?
Ref: https://github.com/ruby/ruby/blob/fce093432eadc191b3647f116a9c2f6748efda3e/lib/prime.rb#L91

#11 - 06/09/2016 04:02 PM - noahgibbs (Noah Gibbs)

Marc-Andre: sorry, you're right. I somehow missed that this code was defining :new.

Yeah, this is probably not a big problem. Defining new is a weird use case.

#12 - 04/28/2017 01:45 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

#13 - 10/24/2017 06:10 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Assigned to Rejected

Files

nth_prime.rb	528 Bytes	01/05/2016	claytonflesher (Clayton Flesher)
nth_prime_test.rb	973 Bytes	01/05/2016	claytonflesher (Clayton Flesher)
README.md	1.27 KB	01/05/2016	claytonflesher (Clayton Flesher)