Ruby trunk - Bug #12832

Calling Object#method hangs for private method defined on module then made public once it's been used to extend class

10/11/2016 06:45 PM - floehopper (James Mead)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	ruby 2.1.3p242 (2014-09-19 revision 47630) [x86_64-darwin14.0] onwards	**Backport:**	2.1: REQUIRED, 2.2: DONE, 2.3: DONE

Description

```
module Foo
  private

  def foo
    "foo"
  end
end

class Bar
  extend Foo

  class << self
    public :foo
  end
end

Bar.foo # => "foo"
Bar.method(:foo) # => #<Method: Class(Bar)#foo>

module Baz
end

class Bar
  class << self
    prepend Baz
  end
end

Bar.method(:foo) # => hangs!
```

Runs OK

- ruby 2.0.0p648 (2015-12-16 revision 53162) [x86_64-darwin14.5.0]
- ruby 2.1.0p0 (2013-12-25 revision 44422) [x86_64-darwin14.0]
- ruby 2.1.1p76 (2014-02-24 revision 45161) [x86_64-darwin14.0]
- ruby 2.1.2p95 (2014-05-08 revision 45877) [x86_64-darwin14.0]

Hangs on last line

- ruby 2.1.3p242 (2014-09-19 revision 47630) [x86_64-darwin14.0]
- ruby 2.1.5p273 (2014-11-13 revision 48405) [x86_64-darwin14.0]
- ruby 2.1.10p492 (2016-04-01 revision 54464) [x86_64-darwin14.0]
- ruby 2.2.5p319 (2016-04-26 revision 54774) [x86_64-darwin14]
- ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-darwin14]
- ruby 2.4.0preview2 (2016-09-09 trunk 56129) [x86_64-darwin14]

Associated revisions

Revision d977cd4e - 10/25/2016 03:54 AM - nobu (Nobuyoshi Nakada)

proc.c: follow the original class

- proc.c (mnew_internal): follow the original class, not to loop the prepended module. [ruby-core:77591] [Bug #12832]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56489 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56489 - 10/25/2016 03:54 AM - nobu (Nobuyoshi Nakada)

proc.c: follow the original class

- proc.c (mnew_internal): follow the original class, not to loop the prepended module. [ruby-core:77591] [Bug #12832]

Revision 56489 - 10/25/2016 03:54 AM - nobu (Nobuyoshi Nakada)

proc.c: follow the original class

- proc.c (mnew_internal): follow the original class, not to loop the prepended module. [ruby-core:77591] [Bug #12832]

Revision 56489 - 10/25/2016 03:54 AM - nobu (Nobuyoshi Nakada)

proc.c: follow the original class

- proc.c (mnew_internal): follow the original class, not to loop the prepended module. [ruby-core:77591] [Bug #12832]

Revision 56489 - 10/25/2016 03:54 AM - nobu (Nobuyoshi Nakada)

proc.c: follow the original class

- proc.c (mnew_internal): follow the original class, not to loop the prepended module. [ruby-core:77591] [Bug #12832]

Revision 4ed0c2c6 - 11/11/2016 04:02 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 56489: [Backport #12832]

```
    * proc.c (mnew_internal): follow the original class, not to loop
      the prepended module.  [ruby-core:77591] [Bug #12832]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_3@56720 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56720 - 11/11/2016 04:02 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 56489: [Backport #12832]

```
* proc.c (mnew_internal): follow the original class, not to loop
  the prepended module.  [ruby-core:77591] [Bug #12832]
```

Revision 4ff235fe - 11/11/2016 10:52 PM - usa (Usaku NAKAMURA)

merge revision(s) 56489: [Backport #12832]

```
    * proc.c (mnew_internal): follow the original class, not to loop
      the prepended module.  [ruby-core:77591] [Bug #12832]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_2@56731 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56731 - 11/11/2016 10:52 PM - usa (Usaku NAKAMURA)

merge revision(s) 56489: [Backport #12832]

```
* proc.c (mnew_internal): follow the original class, not to loop
  the prepended module.  [ruby-core:77591] [Bug #12832]
```

History

#1 - 10/12/2016 04:50 AM - shyouhei (Shyouhei Urabe)

- Assignee set to ko1 (Koichi Sasada)

- Status changed from Open to Assigned

I can reproduce this. Seems like a infinite loop inside ofrb_callable_method_entry_without_refinements. I think this is a showstopper bug.

```
zsh % lldb ./miniruby target.rb
(lldb) target create "./miniruby"
Current executable set to './miniruby' (x86_64).
(lldb) settings set -- target.run-args  "target.rb"
```

```
(lldb) run
Process 23728 launched: './miniruby' (x86_64)
Process 23728 stopped
* thread #1: tid = 0xea3fd, 0x00000001001f2535 miniruby`rb_callable_method_entry_without_refinements + 24 at v
m_method.c:848, stop reason = signal SIGSTOP
    frame #0: 0x00000001001f2535 miniruby`rb_callable_method_entry_without_refinements + 24 at vm_method.c:848
   845      const rb_method_entry_t *me = method_entry_get(klass, id, defined_class_ptr);
   846
   847      if (me) {
-> 848          if (me->def->type == VM_METHOD_TYPE_REFINED) {
   849              if (with_refinement) {
   850                  const rb_cref_t *cref = rb_vm_cref();
   851                  VALUE refinements = cref ? CREF_REFINEMENTS(cref) : Qnil;
(lldb) bt
* thread #1: tid = 0xea3fd, 0x00000001001f2535 miniruby`rb_callable_method_entry_without_refinements + 24 at v
m_method.c:848, stop reason = signal SIGSTOP
  * frame #0: 0x00000001001f2535 miniruby`rb_callable_method_entry_without_refinements + 24 at vm_method.c:848
    frame #1: 0x00000001001f251d miniruby`rb_callable_method_entry_without_refinements(klass=<unavailable>, id
=26065) + 13
    frame #2: 0x000000010011d820 miniruby`mnew_internal(me=0x000000010185ec40, klass=4320521560, obj=432052160
0, id=26065, mclass=4320688000, scope=0, error=1) + 96 at proc.c:1375
    frame #3: 0x00000001001210f3 miniruby`rb_obj_method [inlined] mnew_from_me(scope=<unavailable>, mclass=<un
available>, id=<unavailable>, obj=<unavailable>, klass=<unavailable>, me=<unavailable>) + 26 at proc.c:1403
    frame #4: 0x00000001001210d9 miniruby`rb_obj_method + 24
    frame #5: 0x00000001001210c1 miniruby`rb_obj_method + 84
    frame #6: 0x000000010012106d miniruby`rb_obj_method(obj=4320521600, vid=<unavailable>) + 13
    frame #7: 0x00000001001f0e56 miniruby`vm_call_cfunc + 184 at vm_insnhelper.c:1752
    frame #8: 0x00000001001f0d9e miniruby`vm_call_cfunc(th=0x0000000100604310, reg_cfp=0x00000001007fffa0, cal
ling=<unavailable>, ci=<unavailable>, cc=<unavailable>) + 46
    frame #9: 0x00000001001ff3fe miniruby`vm_call_method_each_type(th=0x0000000100604310, cfp=0x00000001007fff
a0, calling=0x00007fff5fbfd900, ci=<unavailable>, cc=<unavailable>) + 142 at vm_insnhelper.c:2138
    frame #10: 0x00000001001ff95b miniruby`vm_call_method(th=0x0000000100604310, cfp=0x00000001007fffa0, calli
ng=<unavailable>, ci=<unavailable>, cc=<unavailable>) + 235 at vm_insnhelper.c:2288
    frame #11: 0x00000001001f8639 miniruby`vm_exec_core(th=0x0000000100604310, initial=<unavailable>) + 6217 a
t insns.def:1066
    frame #12: 0x00000001001fdda3 miniruby`vm_exec(th=0x0000000100604310) + 131 at vm.c:1711
    frame #13: 0x0000000100082bc0 miniruby`ruby_exec_internal(n=0x000000010185f9d8) + 176 at eval.c:244
    frame #14: 0x000000010008686f miniruby`ruby_run_node [inlined] ruby_exec_node(n=<unavailable>) + 47 at eva
l.c:308
    frame #15: 0x000000010008685b miniruby`ruby_run_node(n=<unavailable>) + 27
    frame #16: 0x0000000010021788e miniruby`main(argc=<unavailable>, argv=<unavailable>) + 78 at main.c:36
    frame #17: 0x00007fff8dc4f5ad libdyld.dylib`start + 1
    frame #18: 0x00007fff8dc4f5ad libdyld.dylib`start + 1
(lldb)
```

#2 - 10/12/2016 09:24 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN to 2.1: REQUIRED, 2.2: REQUIRED, 2.3: REQUIRED

#3 - 10/12/2016 02:22 PM - floehopper (James Mead)

In case it helps, it seems as if Ruby v2.3 is "more" broken than earlier versions. If I remove the Bar.method(:foo) lookup *before* the Baz module is prepended, the final lookup still hangs in Ruby v2.3, but not in v2.2. See below:

```
module Foo
  private

  def foo
    "foo"
  end
end

class Bar
  extend Foo

  class << self
    public :foo
  end
end

module Baz
end

class Bar
  class << self
```

```
    prepend Baz
  end
end

Bar.method(:foo) # => hangs in Ruby v2.3.1, but not v2.2.5
```

#4 - 10/24/2016 05:24 PM - chrisroos (Chris Roos)

I'm not sure whether this is useful information but calling #instance_method on Bar's metaclass also causes Ruby to hang. Using James's most recent code snippet, the following code seems to cause the same problem as Bar.method(:foo).

```
metaclass = (class << Bar; self; end)
p metaclass.instance_method(:foo) # => hangs in Ruby v2.3.1, but not v2.2.5
```

#5 - 10/25/2016 03:54 AM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Closed

Applied in changeset r56489.

---

proc.c: follow the original class

- proc.c (mnew_internal): follow the original class, not to loop the prepended module. [ruby-core:77591] [Bug #12832]

#6 - 11/01/2016 09:34 AM - floehopper (James Mead)

nobu (Nobuyoshi Nakada): Thank you for fixing this. Is it expected that a bug like this would cause the interpreter not to accept interrupt signals?

#7 - 11/01/2016 11:45 AM - nobu (Nobuyoshi Nakada)

It's not expected but a bug is always what causes unexpected behavior.

#8 - 11/01/2016 12:33 PM - floehopper (James Mead)

nobu (Nobuyoshi Nakada): Is it possible there is a separate problem with the code such that interrupt signals are incorrectly being ignored...?

#9 - 11/01/2016 08:10 PM - kernigh (George Koehler)

Ruby defers signals. At safe moments, Ruby checks if a signal arrived, then handles it. If Ruby is stuck in an infinite loop (because of a bug like this one), then Ruby would never check if a signal arrived, so the signal would be ignored. After fixing the bug, signals should work again. Ruby's deferred signals seem like Perl's, see http://perldoc.perl.org/perlipc.html#Deferred-Signals-(Safe-Signals)

#10 - 11/01/2016 08:29 PM - floehopper (James Mead)

@George: Thanks for explaining. Makes sense.

#11 - 11/11/2016 04:02 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.1: REQUIRED, 2.2: REQUIRED, 2.3: REQUIRED to 2.1: REQUIRED, 2.2: REQUIRED, 2.3: DONE

ruby_2_3 r56720 merged revision(s) 56489.

#12 - 11/11/2016 10:52 PM - usa (Usaku NAKAMURA)

- Backport changed from 2.1: REQUIRED, 2.2: REQUIRED, 2.3: DONE to 2.1: REQUIRED, 2.2: DONE, 2.3: DONE

ruby_2_2 r56731 merged revision(s) 56489.