Ruby master - Bug #17379

Refinement with modules redefinition bug

12/09/2020 04:21 AM - marcandre (Marc-Andre Lafortune)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>ko1 (Koichi Sasada)</td>
</tr>
<tr>
<td>Target version:</td>
<td>ruby 3.0.0dev (2020-12-05T10:40:00Z master 9dbb2bfd73) [x86_64-darwin18]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN</td>
</tr>
</tbody>
</table>

Description

Depending on the circumstance, a refinement can be modified even after being used:

```ruby
def foo
  [:base]
end

module M
  def foo
    super << :M
  end
end

module Ext
  refine Object do
    include M
  end
end

using Ext

p 'asd'.foo unless ENV['SKIP'] # => [:base, :M] (ok)

module M
  def foo
    super << :new_ref
  end
end

p 'asd'.foo # => depends (not ok)
```

Running this gives:

```
$ ruby refinement.rb
[:base, :M]
[:base, :M] # => ok
$ SKIP=t ruby refinement.rb
[:base, :new_ref] # => should be [:base, :M]
```

Related issues:

Related to Ruby master - Bug #17429: Prohibit include/prepend in refinement m...

03/15/2021

History

#1 - 12/09/2020 04:21 AM - marcandre (Marc-Andre Lafortune)

- Subject changed from Refinement with modules redefinition issues to Refinement with modules redefinition bug

#2 - 12/09/2020 08:54 AM - shugo (Shugo Maeda)

- Assignee changed from shugo (Shugo Maeda) to matz (Yukihiro Matsumoto)

It's an inline method cache issue and hard to solve without performance regression.
If the behavior is not acceptable as a limitation of Refinements, it may be better to prohibit module inclusion in Refinements.

#3 - 12/09/2020 09:02 AM - shugo (Shugo Maeda)
shugo (Shugo Maeda) wrote in #note-2:

If the behavior is not acceptable as a limitation of Refinements, it may be better to prohibit module inclusion in Refinements.

I think there is no need to prohibit inclusion of frozen modules, at least for this issue.

#4 - 12/09/2020 10:14 AM - shugo (Shugo Maeda)
- Assignee changed from matz (Yukihiro Matsumoto) to shugo (Shugo Maeda)

shugo (Shugo Maeda) wrote in #note-2:

It's an inline method cache issue and hard to solve without performance regression.

I might be wrong, and will investigate it further.

#5 - 12/10/2020 05:35 PM - marcandre (Marc-Andre Lafortune)
Probable same bug, without using, found by Daniel DeLorme:

```ruby
class Foo
  def foo
    p :hello
  end
end

module Code
  def foo
    p :A
  end
end

module Extension
  refine Foo do
    prepend Code
  end
end

Foo.new.foo unless ENV['SKIP'] # => :hello (ok)
Foo.prepend Code
Foo.new.foo # => depends (not ok)
```
gives:

$ ruby refinement.rb
:hello
:hello
$ SKIP=t ruby refinement.rb
:A

#6 - 12/11/2020 12:55 AM - shugo (Shugo Maeda)
marcandre (Marc-Andre Lafortune) wrote in #note-5:

Probably same bug, without using, found by Daniel DeLorme:

It seems a different issue, so I've filed #17386.

#7 - 12/23/2020 02:05 AM - shugo (Shugo Maeda)
- Assignee changed from shugo (Shugo Maeda) to ko1 (Koichi Sasada)

It seems that callable method entry cache caused the problem.
The problem doesn't occur with the following patch:

```diff
index a0ccdb8a51..d3a3926780 100644
--- a/vm_method.c
+++ b/vm_method.c
diff --git a/vm_method.c b/vm_method.c
index a0ccdb8a51..d3a3926780 100644
```
a/vm_method.c
+++ b/vm_method.c
@@ -1023,7 +1023,7 @@ prepare_callable_method_entry(VALUE defined_class, ID id, const rb_method_entry_
 mtbl = RCLASS_EXT(defined_class)->callable_m_tbl = rb_id_table_create(0);
 }
 cme = rb_method_entry_complement_defined_class(me, me->called_id, defined_class);
- rb_id_table_insert(mtbl, id, (VALUE)cme);
+ // rb_id_table_insert(mtbl, id, (VALUE)cme);
 RB_OBJ_WRITTEN(defined_class, Qundef, (VALUE)cme);
 VM_ASSERT(callable_method_entry_p(cme));
 }
@@ -1122,7 +1122,8 @@ callable_method_entry(VALUE klass, ID mid, VALUE *defined_class_ptr)
 VM_ASSERT(RB_TYPE_P(klass, T_CLASS) || RB_TYPE_P(klass, T_ICLASS));
 RB_VM_LOCK_ENTER();
 {
- cme = cached_callable_method_entry(klass, mid);
+ // cme = cached_callable_method_entry(klass, mid);
+ cme = NULL;
 if (cme) {
- if (defined_class_ptr != NULL) *defined_class_ptr = cme->defined_class;
- cache_callable_method_entry(klass, mid, cme);
@@ -1139,7 +1140,7 @@ callable_method_entry(VALUE klass, ID mid, VALUE *defined_class_ptr)
 cme = negative_cme(mid);
 }
- cache_callable_method_entry(klass, mid, cme);
+ // cache_callable_method_entry(klass, mid, cme);
 }
 RB_VM_LOCK_LEAVE();

(The fix of prepare_callable_method_entry() is for Kernel#method)

ko1 [Koichi Sasada], is there any way to fix the problem without performance regression?

#8 - 01/06/2021 02:02 AM - jeremyevans0 (Jeremy Evans)

- Related to Bug #17429: Prohibit include/prepend in refinement modules added