Ruby master - Bug #11189
alias prepended module

05/27/2015 07:58 PM - ko1 (Koichi Sasada)

<table>
<thead>
<tr>
<th>Status</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
<tr>
<td>ruby -v:</td>
<td>Backport: 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN</td>
</tr>
</tbody>
</table>

Description
module P
  def m1
    p :P_m1
    super
  end

  def m2
    p :P_m2
    super
  end
end

class C0
  def m1
    p :C0_m1
  end
end

class C1 < C0
  def m1
    p :C1_m1
    super
  end
  prepend P
  alias m2 m1
end
C1.new.m2

このプログラムは、

:C0_m1
:C1_m1
:P_m2
:P_m1

という結果になります。super で辿っているはずなのに、同じモジュールのメソッドが 2 回呼ばれます。super で辿っていったら、必ず継承元へ行く、2度同じクラスは来ない、という常識があると思っていたので、Ruby のにはかなり驚きました。

この挙動は、設定時に:

- alias で C1 のメソッドテーブルに、C1#m2 を P#m1 へ飛ばすように設定する

次に、呼び出された時に:

- m2 が呼ばれる
- P#m2 を実行する
- P#m2 の super を呼ぶ
- C1#m2 を見ると、P#m1 へ飛ぶように alias が設定されている
- P#m1 を実行する、super する
- (なんでか C1#m1 をスキップする なんで?)
- C0#m1 が呼ばれる

2022/03/15 1/5
C1#foo を呼ぶのは気持ち悪いですが。

ちなみに、2.1 では C1#m1 が呼ばれていました。

たしか、alias + prepend の議論が以前あったと思うのですが。

あ、[Bug #7842] だ。

Related issues:
Related to Ruby master - Bug #7842: An alias of a "prepend"ed method skips th...

Closed 02/13/2013

Associated revisions
Revision c60aaed1 - 08/27/2020 03:37 PM - jeremyevans (Jeremy Evans)
Fix Method#super_method for aliased methods

Previously, Method#super_method looked at the called_id to determine the method id to use, but that isn't correct for aliased methods, because the super target depends on the original method id, not the called_id.

Additionally, aliases can reference methods defined in other classes and modules, and super lookup needs to start in the super of the defined class in such cases.

This adds tests for Method#super_method for both types of aliases, one that uses VM_METHOD_TYPE_ALIAS and another that does not. Both check that the results for calling super methods return the expected values.

To find the defined class for alias methods, add an rb_prefix to find_defined_class_by_owner in vm_insnhelper.c and make it non-static, so that it can be called from method_super_method in proc.c.

This bug was original discovered while researching [Bug #11189].

Fixes [Bug #17130]

Revision c98aa2db - 03/20/2021 04:29 AM - nagachika (Tomoyuki Chikanaga)
merge revision(s) c60aaed1856b2b6f90de0992c34771830019e021: [Backport #17130]
methods return the expected values.

To find the defined class for alias methods, add an rb_ prefix to find_defined_class_by_owner in vm_insnhelper.c and make it non-static, so that it can be called from method_super_method in proc.c.

This bug was originally discovered while researching [Bug #11189].

Fixes [Bug #17130]

Revision 04601ea6 - 04/05/2021 12:12 AM - usa (Usaku NAKAMURA)

merge revision(s) c60aaed1856b2b6f90d0e992c347718300190e021: [Backport #17130]

Fix Method#super_method for aliased methods

Previously, Method#super_method looked at the called_id to determine the method id to use, but that isn't correct for aliased methods, because the super target depends on the original method id, not the called_id.

Additionally, aliases can reference methods defined in other classes and modules, and super lookup needs to start in the super of the defined class in such cases.

This adds tests for Method#super_method for both types of aliases, one that uses VM_METHOD_TYPE_ALIAS and another that does not. Both check that the results for calling super methods return the expected values.

To find the defined class for alias methods, add an rb_ prefix to find_defined_class_by_owner in vm_insnhelper.c and make it non-static, so that it can be called from method_super_method in proc.c.

This bug was originally discovered while researching [Bug #11189].

Fixes [Bug #17130]

Revision 67933 - 04/05/2021 12:12 AM - usa (Usaku NAKAMURA)

merge revision(s) c60aaed1856b2b6f90d0e992c347718300190e021: [Backport #17130]

Fix Method#super_method for aliased methods

Previously, Method#super_method looked at the called_id to determine the method id to use, but that isn't correct for aliased methods, because the super target depends on the original method id, not the called_id.

Additionally, aliases can reference methods defined in other classes and modules, and super lookup needs to start in the super of the defined class in such cases.

This adds tests for Method#super_method for both types of aliases, one that uses VM_METHOD_TYPE_ALIAS and another that does not. Both check that the results for calling super methods return the expected values.

To find the defined class for alias methods, add an rb_ prefix to find_defined_class_by_owner in vm_insnhelper.c and make it non-static, so that it can be called from method_super_method in proc.c.

This bug was originally discovered while researching [Bug #11189].

Fixes [Bug #17130]
今、こんなコードだと、
C1(m_tbl 空)
T_ICLASS(m_tbl 空) -> P
T_ICLASS(C1 m_tbl) -> C1
C0
Object...

という継承関係(下が親)が作られるけど、alias のために C1 に m_tbl 作らないといけよね。

メソッド定義については、
alias の時は、C1::m_tbl へ
メソッド定義の時は T_ICLASS(C1 の m_tbl)へ
と分けなきゃいけないのも、なんかかっこ悪いですね。

- Related to Bug #7842: An alias of a "prepend"ed method skips the original method when calling super added

This output would be against my expection.  If C1 prepends P, then methods in P must be considered before methods in C1.  Consider C1.ancestors:

[P, C1, C0, Object, Kernel, BasicObject]

P#m2 must be called before C1#m2, and alias m2 m1 in C1 only modifies the method table in C1, not in P.

All Ruby versions I tested have the same behavior as Ruby 2.1:

There does appear to be a related bug in super_method, though:

Note that because of the way alias works with prepend, you can get an infinite loop in method lookup with aliases and only calling super:
Fix Method#super_method for aliased methods

Previously, Method#super_method looked at the called_id to determine the method id to use, but that isn't correct for aliased methods, because the super target depends on the original method id, not the called_id.

Additionally, aliases can reference methods defined in other classes and modules, and super lookup needs to start in the super of the defined class in such cases.

This adds tests for Method#super_method for both types of aliases, one that uses VM_METHOD_TYPE_ALIAS and another that does not. Both check that the results for calling super methods return the expected values.

To find the defined class for alias methods, add an rb_prefix to find_defined_class_by_owner in vm_insnhelper.c and make it non-static, so that it can be called from method_super_method in proc.c.

This bug was original discovered while researching [Bug #11189].

Fixes [Bug #17130]