Ruby - Bug #19394

cvars in instance of cloned class point to source class's cvars even after class_variable_set on clone

01/31/2023 07:12 PM - jamescdavis (James Davis)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.1.3p185 (2022-11-24 revision 1a6b16756e) [x86_64-darwin21]	**Backport:**	3.0: UNKNOWN, 3.1: DONE, 3.2: DONE

Description

This unexpected change in behavior happens between Ruby 3.0.x and 3.1.x. In Ruby >= 3.1, when a class with a cvar is cloned (or duped), the cvar in instances of the cloned class continues to point to the source class's cvar after the clone has its cvar updated with class_variable_set. In Ruby < 3.1, the cloned class instance points to the updated cvar, as expected.

It seems likely that this is a bug in the cvar cache introduced in Ruby 3.1.

Repro:

```
class Foo
  @@bar = 'bar'

  def print_bar
    puts "#{self.class.name} (from instance): #{@@bar} #{@@bar.object_id}"
  end
end

foo_bar = Foo.class_variable_get(:@@bar)
puts "Foo (class_variable_get): #{foo_bar} #{foo_bar.object_id}"

Foo.new.print_bar

FooClone = Foo.clone

FooClone.class_variable_set(:@@bar, 'bar_clone')

foo_clone_bar = FooClone.class_variable_get(:@@bar)
puts "FooClone (class_variable_get): #{foo_clone_bar} #{foo_clone_bar.object_id}"

FooClone.new.print_bar
```

Ruby 3.0.5:

```
Foo (class_variable_get): bar 60
Foo (from instance): bar 60
FooClone (class_variable_get): bar_clone 80
FooClone (from instance): bar_clone 80
```

Ruby 3.1.3, 3.2.0:

```
Foo (class_variable_get): bar 60
Foo (from instance): bar 60
FooClone (class_variable_get): bar_clone 80
FooClone (from instance): bar 60
```

Something similar happens when there are multiple clones and a cvar that the source class does not have defined is set on the clones. In this case, the cvars in instances of the clones all point to the first clone's cvar.

Repro:

```
class Foo
  def print_bar
```

```
    puts "#{self.class.name} (from instance): #{@@bar} #{@@bar.object_id}"
  end
end

Foo1 = Foo.clone
Foo2 = Foo.clone
Foo3 = Foo.clone

Foo1.class_variable_set(:@@bar, 'bar1')
Foo2.class_variable_set(:@@bar, 'bar2')
Foo3.class_variable_set(:@@bar, 'bar3')

foo1_bar = Foo1.class_variable_get(:@@bar)
foo2_bar = Foo2.class_variable_get(:@@bar)
foo3_bar = Foo3.class_variable_get(:@@bar)

puts "Foo1 (class_variable_get): #{foo1_bar} #{foo1_bar.object_id}"
puts "Foo2 (class_variable_get): #{foo2_bar} #{foo2_bar.object_id}"
puts "Foo3 (class_variable_get): #{foo3_bar} #{foo3_bar.object_id}"

Foo1.new.print_bar
Foo2.new.print_bar
Foo3.new.print_bar
```

Ruby 3.0.5:

```
Foo1 (class_variable_get): bar1 60
Foo2 (class_variable_get): bar2 80
Foo3 (class_variable_get): bar3 100
Foo1 (from instance): bar1 60
Foo2 (from instance): bar2 80
Foo3 (from instance): bar3 100
```

Ruby 3.1.3, 3.2.0:

```
Foo1 (class_variable_get): bar1 60
Foo2 (class_variable_get): bar2 80
Foo3 (class_variable_get): bar3 100
Foo1 (from instance): bar1 60
Foo2 (from instance): bar1 60
Foo3 (from instance): bar1 60
```

Related issues:

Related to Ruby - Feature #17763: Implement cache for cvars	**Closed**

History

#1 - 01/31/2023 07:44 PM - byroot (Jean Boussier)

- Related to Feature #17763: Implement cache for cvars added

#2 - 01/31/2023 07:45 PM - byroot (Jean Boussier)

Based on the timeline and description, this is likely a result of [Feature #17763]. cc @eileencodes (Eileen Uchitelle) & @tenderlovemaking (Aaron Patterson)

#3 - 01/31/2023 07:52 PM - eileencodes (Eileen Uchitelle)

I will take a look, thanks for the ping Jean.

#4 - 02/07/2023 08:53 PM - eileencodes (Eileen Uchitelle)

I've pushed a fix up to github https://github.com/ruby/ruby/pull/7265. We added a cref to the cvar cache that we can check if it's the same as the cref we have. If they are different we shouldn't read from the cache.

#5 - 02/08/2023 08:57 PM - eileencodes (Eileen Uchitelle)

The fix ended up being a bit more involved. We needed two PRs.

One to copy the CVAR table on clone: https://github.com/ruby/ruby/pull/7275

And one to check the cref and not read from the cache if they differ (ie is the cloned class): https://github.com/ruby/ruby/pull/7265

Both will need to be backported to 3.2.x and 3.1.x when merged.

#6 - 07/01/2023 04:49 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN to 3.0: UNKNOWN, 3.1: REQUIRED, 3.2: REQUIRED

#7 - 07/01/2023 04:55 AM - nagachika (Tomoyuki Chikanaga)

- Status changed from Open to Closed

This issue should be fixed at 40f090f4339820d19da8ecdf81a981489c22eb57 and 135a5eb716399443da58db342de6093c91b5ad62 in master branch.

Thank you for creating the PRs to backport them into stable branches.
I will handle it (the one for ruby_3_2 branch) soon.

3.1: https://github.com/ruby/ruby/pull/7889
3.2: https://github.com/ruby/ruby/pull/7888

#8 - 07/01/2023 05:19 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 3.0: UNKNOWN, 3.1: REQUIRED, 3.2: REQUIRED to 3.0: UNKNOWN, 3.1: REQUIRED, 3.2: DONE

Merged https://github.com/ruby/ruby/pull/7888

#9 - 07/25/2023 11:06 AM - usa (Usaku NAKAMURA)

- Backport changed from 3.0: UNKNOWN, 3.1: REQUIRED, 3.2: DONE to 3.0: UNKNOWN, 3.1: DONE, 3.2: DONE

merged https://github.com/ruby/ruby/pull/7889
thx!