# Ruby - Bug #6614

## GC doesn't collect objects bound to (collectable) proc

06/21/2012 08:24 AM - rogerdpack (Roger Pack)

| | | | |
|---|---|---|---|
| **Status:** | Rejected | | |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Target version:** | | | |
| **ruby -v:** | tcs-ruby 1.9.3p28 (2012-01-28, TCS patched 2012-01-30) [i386-mingw32] | **Backport:** | |

**Description**

Hello all.
Previously discussed here: http://www.ruby-forum.com/topic/4402823

I would expect the following code to eventually run the finalizers mentioned:

```
def finalized_puts bad
ObjectSpace.define_finalizer(bad) { print 'collector run', bad }
end

loop { finalized_puts [] }
```

This works in jruby as expected, but not in MRI. This has the unfortunate side effect of making define_finalizer *very* difficult to actually use right http://www.mikeperham.com/2010/02/24/the-trouble-with-ruby-finalizers/

Cheers!

### History

#### #1 - 06/21/2012 09:33 AM - nobu (Nobuyoshi Nakada)

*- Status changed from Open to Rejected*

=begin
Finalizers will run ((*AFTER*)) the target object has been destroyed and collected, so finalizers ((*MUST NOT*)) refer the object.
I'm not certain why it is called in JRuby, but suspect it might be optimized out?
=end

#### #2 - 06/21/2012 10:03 AM - nobu (Nobuyoshi Nakada)

=begin
A correction.

Your code uses (({print})) but not (({printf})), so the arguments cannot be optimized out, even if necessary assumptions are all met.
A possibility would be JRuby calls finalizers ((*BEFORE*)) the destruction.
What's printed in JRuby?
=end

#### #3 - 06/27/2012 02:23 AM - Anonymous

> Your code uses print' but not printf', so the arguments cannot be optimized out, even if necessary assumptions are all met.
> A possibility would be JRuby calls finalizers *BEFORE* the destruction.
> What's printed in JRuby?

runcollector runcollector

so it still has a reference to the old object. Which way seems more
sane somehow, but is probably wrong.

I suppose my gripe is more that define_finalizer too many times
creates proc's that are essentially "orphaned" but cannot be
collected, which is basically a memory leak waiting to happen.
Cheers!

-r

#### #4 - 06/27/2012 02:03 PM - headius (Charles Nutter)

Finalizers in JRuby are stored on a reference from the object itself, so they don't actually prevent the object from collecting.

I'd vote for MRI to do the same, but I don't think I'll win that fight :)

#### #5 - 06/27/2012 03:35 PM - nobu (Nobuyoshi Nakada)

headius (Charles Nutter) wrote:

> Finalizers in JRuby are stored on a reference from the object itself, so they don't actually prevent the object from collecting.

You're missing the point.

#### #6 - 06/27/2012 05:23 PM - cjheath (Clifford Heath)

On 27/06/2012, at 4:35 PM, nobu (Nobuyoshi Nakada) wrote:

> headius (Charles Nutter) wrote:
>
> > Finalizers in JRuby are stored on a reference from the object itself, so they don't actually prevent the object from collecting.
>
> You're missing the point.

Is he? Surely somewhere in that loop{finalized_puts []} the GC will activate, and that will find collectable objects (with finalisers attached in JRuby). In MRI, they might be found but they won't be finalised, if I understand the behaviour right.

Clifford Heath.

---

> Bug #6614: GC doesn't collect objects bound to (collectable) proc
> https://bugs.ruby-lang.org/issues/6614#change-27513
>
> Author: rogerdpack (Roger Pack)
> Status: Rejected
> Priority: Normal
> Assignee:
> Category:
> Target version:
> ruby -v: tcs-ruby 1.9.3p28 (2012-01-28, TCS patched 2012-01-30) [i386-mingw32]
>
> Hello all.
> Previously discussed here: http://www.ruby-forum.com/topic/4402823
>
> I would expect the following code to eventually run the finalizers mentioned:
>
> def finalized_puts bad
> ObjectSpace.define_finalizer(bad) { print 'collector run', bad }
> end
>
> loop { finalized_puts [] }
>
> This works in jruby as expected, but not in MRI. This has the unfortunate side effect of making define_finalizer *very* difficult to actually use right
> http://www.mikeperham.com/2010/02/24/the-trouble-with-ruby-finalizers/
>
> Cheers!
>
> --
> http://bugs.ruby-lang.org/

#### #7 - 06/27/2012 06:32 PM - shyouhei (Shyouhei Urabe)

cjheath (Clifford Heath) wrote:

> On 27/06/2012, at 4:35 PM, nobu (Nobuyoshi Nakada) wrote:

> headius (Charles Nutter) wrote:
>
> > Finalizers in JRuby are stored on a reference from the object itself, so they don't actually prevent the object from collecting.
>
> You're missing the point.

Is he? Surely somewhere in that loop{finalized_puts []} the GC will activate, and that will find collectable objects (with finalisers attached in JRuby). In MRI, they might be found but they won't be finalised, if I understand the behaviour right.

When a proc that touches an object is attatched to that object's finalizer, that finalizer can "escape" that object.

ObjectSpace.define_finalizer(obj) { $escaped = obj } # Woo!

This would break the whole concept of finalizing.  Nobu's point is that how JRuby prevents this situation.

#### #8 - 06/28/2012 05:29 PM - funny_falcon (Yura Sokolov)

It seems that ObjectSpace.define_finalizer is complitely broken with Ruby 1.9.3
And even in Ruby 1.8.7 finalizer proc ought to be created in other scope to not hold reference to object in its binding.
See #6664 and https://gist.github.com/3009867 for more