Ruby - Bug #20253

`Proc.dup` and `Proc#clone` don't preserve finalizers

02/09/2024 04:47 PM - byroot (Jean Boussier)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	3.0: WONTFIX, 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

While reviewing the fix for [Bug #20250] @peterzhu2118 (Peter Zhu) pointed that FL_FINALIZE should probably also be cleared.

However after some extra testing, it appears Object#dup and Object#clone do copy over the finalizer (which makes sense).

But for some reason Proc has its own dup/clone implementation, which does copy the FL_FINALIZE flag, but doesn't copy the finalizer:

Test script:

```
def fin(sym)
  ->(_) { p sym }
end

obj = Object.new
ObjectSpace.define_finalizer(obj, fin(:obj))
obj.dup
obj.clone

proc = Proc.new { }
ObjectSpace.define_finalizer(proc, fin(:proc))
proc.dup
proc.clone
```

Expected output:

```
:proc
:proc
:proc
:obj
:obj
:obj
```

Actual output:

```
:proc
:obj
:obj
:obj
```

This discrepancy is present all the way back to Ruby 1.9.

It's so niche I'm not sure it's worth a backport though...

Related issues:

Related to Ruby - Bug #20250: Crash with "Object ID seen, but not in mapping ...	**Closed**

Associated revisions

Revision de1a586ecc2ee7f465f0c0a69291054136a3a819 - 02/12/2024 05:31 PM - byroot (Jean Boussier)

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com

Revision de1a586ecc2ee7f465f0c0a69291054136a3a819 - 02/12/2024 05:31 PM - byroot (Jean Boussier)

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com

Revision de1a586e - 02/12/2024 05:31 PM - byroot (Jean Boussier)

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com

Revision a63e979853783601a60228b45741f8b3776e5507 - 03/21/2024 01:45 AM - NARUSE, Yui

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819: [Backport #20250] (#10308)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

```
    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.

    proc.c: get rid of `CLONESETUP`
    MIME-Version: 1.0
    Content-Type: text/plain; charset=UTF-8
    Content-Transfer-Encoding: 8bit

    [Bug #20253]

    All the way down to Ruby 1.9, `Proc`, `Method`, `UnboundMethod`
    and `Binding` always had their own specific clone and dup routine.

    This caused various discrepancies with how other objects behave
    on `dup` and `clone. [Bug #20250], [Bug #20253].
```

```
    This commit get rid of `CLONESETUP` and use the the same codepath
    as all other types, so ensure consistency.

    NB: It's still not accepting the `freeze` keyword argument on `clone`.

    Co-Authored-By: Étienne Barrié <etienne.barrie@gmail.com>
```

Revision a63e979853783601a60228b45741f8b3776e5507 - 03/21/2024 01:45 AM - NARUSE, Yui

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819: [Backport #20250] (#10308)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

```
    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.

    proc.c: get rid of `CLONESETUP`
    MIME-Version: 1.0
    Content-Type: text/plain; charset=UTF-8
    Content-Transfer-Encoding: 8bit

    [Bug #20253]

    All the way down to Ruby 1.9, `Proc`, `Method`, `UnboundMethod`
    and `Binding` always had their own specific clone and dup routine.

    This caused various discrepancies with how other objects behave
    on `dup` and `clone. [Bug #20250], [Bug #20253].

    This commit get rid of `CLONESETUP` and use the the same codepath
    as all other types, so ensure consistency.

    NB: It's still not accepting the `freeze` keyword argument on `clone`.

    Co-Authored-By: Étienne Barrié <etienne.barrie@gmail.com>
```

Revision a63e9798 - 03/21/2024 01:45 AM - NARUSE, Yui

merge revision(s) d19d683a354530a27b4cbb049223f8dc70c75849,de1a586ecc2ee7f465f0c0a69291054136a3a819: [Backport #20250] (#10308)

rb_obj_setup: do not copy RUBY_FL_SEEN_OBJ_ID

```
    [Bug #20250]

    We're seting up a new instance, so it never had an associated
    object_id.

    proc.c: get rid of `CLONESETUP`
    MIME-Version: 1.0
    Content-Type: text/plain; charset=UTF-8
    Content-Transfer-Encoding: 8bit

    [Bug #20253]

    All the way down to Ruby 1.9, `Proc`, `Method`, `UnboundMethod`
    and `Binding` always had their own specific clone and dup routine.

    This caused various discrepancies with how other objects behave
    on `dup` and `clone. [Bug #20250], [Bug #20253].

    This commit get rid of `CLONESETUP` and use the the same codepath
    as all other types, so ensure consistency.

    NB: It's still not accepting the `freeze` keyword argument on `clone`.

    Co-Authored-By: Étienne Barrié <etienne.barrie@gmail.com>
```

History

#1 - 02/09/2024 04:48 PM - byroot (Jean Boussier)

- Related to Bug #20250: Crash with "Object ID seen, but not in mapping table: proc" error added

#2 - 02/09/2024 04:59 PM - byroot (Jean Boussier)

While looking at this, it appears that instance variables are also not copied over in Proc#dup, but somehow are in Proc#clone:

```
def ivar_dup(obj)
  obj.instance_variable_set(:@foo, 1)
  p [:dup, obj.dup.instance_variable_get(:@foo)]
  p [:clone, obj.clone.instance_variable_get(:@foo)]
end

ivar_dup(Object.new)
ivar_dup(Proc.new { })
```

```
[:dup, 1]
[:clone, 1]
[:dup, nil]
[:clone, 1]
```

Which really raise the question of why Proc has a completely different codepath for duping/cloning, AFAIK no other core type does this.

#3 - 02/12/2024 11:28 AM - byroot (Jean Boussier)

We just found yet another issue with @etienne:

```
>> Proc.new { }.freeze.clone
(irb):1:in `initialize_copy': can't modify frozen Proc: #<Proc:0x000000011e3a96b0 (irb):1> (FrozenError)
 from (irb):1:in `initialize_clone'
 from (irb):1:in `clone'
 from (irb):1:in `<main>'
```

This one was introduced in Ruby 3.3.0.

#4 - 02/12/2024 11:30 AM - byroot (Jean Boussier)

https://github.com/ruby/ruby/pull/9926 fixes all the issues above and a few others.

#5 - 02/12/2024 05:31 PM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset git|de1a586ecc2ee7f465f0c0a69291054136a3a819.

---

proc.c: get rid of CLONESETUP

[Bug #20253]

All the way down to Ruby 1.9, Proc, Method, UnboundMethod
and Binding always had their own specific clone and dup routine.

This caused various discrepancies with how other objects behave
on dup and `clone. [Bug #20250], [Bug #20253].

This commit get rid of CLONESETUP and use the the same codepath
as all other types, so ensure consistency.

NB: It's still not accepting the freeze keyword argument on clone.

Co-Authored-By: Étienne Barrié etienne.barrie@gmail.com