Freeze all Range object

01/04/2019 03:12 AM - ko1 (Koichi Sasada)

Status: Open
Priority: Normal
Assignee: matz (Yukihiro Matsumoto)
Target version:

Description

Abstract
Range is now non-frozen. How about to freeze all of Range objects?

Background
We freeze some type of objects, Numerics (r47523) and Symbols [Feature #8906].
I believe making objects immutable solves some kind of programming difficulties.

Range is mutable, at least it is written in Range literal. So we can write the following weird program.

```ruby
2.times{
  r = (1..3)
  p r.instance_variable_get(:@foo)
  #=> 1st time: nil
  #=> 2nd time: :bar
  r.instance_variable_set(:@foo, :bar)
}
```

in range.c, there is a comment (thanks znz-san):

```c
static void
range_modify(VALUE range)
{
  rb_check_frozen(range);
  /* Ranges are immutable, so that they should be initialized only once. */
  if (RANGE_EXCL(range) != Qnil) {
    rb_name_err_raise("'initialize' called twice", range, ID2SYM(idInitialize));
  }
}
```

Patch

Index: range.c
===================================================================
--- range.c (リビジョン 66699)
+++ range.c (作業コピー)
@@ -45,6 +45,8 @@
   RANGE_SET_EXCL(range, exclude_end);
   RANGE_SET_BEG(range, beg);
   RANGE_SET_END(range, end);
+   rb_obj_freeze(range);
}

Discussion
There are several usage of mutable Range in tests.
• (1) taint-flag
• (2) add singleton methods.
• (3) subclass with mutable states

Maybe (2) and (3) are points.

Thanks,
Koichi

History

#1 - 01/04/2019 07:29 AM - marcandre (Marc-Andre Lafortune)
I think that (2) and (3) are indeed capital points. Freezing range literals (only) might be a better idea? with an approach like frozen string literals?

Not that even frozen ranges aren't completely immutable:

```ruby
r = ('a'..'z').freeze
r.end.upcase!
r # => "a"..'Z'
```

#2 - 06/22/2019 02:38 PM - Eregon (Benoit Daloze)
I think it would make sense to freeze Range literals.

Adding methods to Range might be reasonable, but singleton methods, I would think much less so.

#3 - 06/23/2019 04:47 AM - mame (Yusuke Endoh)
I guess Eregon (Benoit Daloze) came from #15950. Will ary[-3..] be as efficient as ary[-3, 3] by freezing and deduping a literal (-3..)? Looks good if we can confirm it by an experiment.

Some thoughts:

• Even if a range is frozen, ("a"..'z") should not be deduped because of the reason marcandre (Marc-Andre Lafortune) said.
• I'm for freezing all Ranges, not only Range literals. I hate the idea of freezing only literals because casually mixing frozen and unfrozen objects leads to hard-to-debug bugs that depend upon data flow.
• It would be the most elegant if the combination of MJIT and escape analysis solves this kind of performance problems.