Ruby master - Bug #10684

Block arity changes through Enumerable methods

12/31/2014 08:39 PM - jakesower (Jake Sower)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.2.0	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN

Description

Blocks traveling through methods in Enumerable have their arity changed before reaching #each. Example:

```
class MyEnumerator
include Enumerable

def initialize(ary)
@ary = ary
end

def each(&block)
puts block.arity
@ary.each(&block)
end
end

my_enum = MyEnumerator.new([1,2,3])
my_enum.each{|x| x}    # outputs 1
my_enum.detect{|x| x}  # outputs -1
```

This is surprising behavior. I would expect the output to be 1 in both cases since the blocks appear identical to me as a programmer.

History

#1 - 12/31/2014 08:43 PM - jakesower (Jake Sower)

Blocks traveling through methods in Enumerable have their arity changed before reaching #each. Example:

```
class MyEnumerable
  include Enumerable

  def initialize(ary)
    @ary = ary
  end
  def each(&block)
    puts block.arity
    @ary.each(&block)
  end
end

my_enum = MyEnumerable.new([1,2,3])
my_enum.each{|x| x}    # outputs 1
my_enum.detect{|x| x}  # outputs -1
```

This is surprising behavior. I would expect the output to be 1 in both cases since the blocks appear identical to me as a programmer.

#2 - 01/01/2015 12:03 AM - austin (Austin Ziegler)

It's not that surprising to me.

While Enumerable#detect is written in C, in Ruby I might implement it as:

```
module Enumerable
  def detect2
    return enum_for(:detect2) unless block_given?
    v = each { |x| break x if yield x }
```

```
    v == self ? nil : v
  end
end
```

In this case, the C code is instructive:

```
static VALUE
enum_find(int argc, VALUE *argv, VALUE obj)
{
    NODE *memo;
    VALUE if_none;

    rb_scan_args(argc, argv, "01", &if_none);
    RETURN_ENUMERATOR(obj, argc, argv);
    memo = NEW_MEMO(Qundef, 0, 0);
    rb_block_call(obj, id_each, 0, 0, find_i, (VALUE)memo);
    if (memo->u3.cnt) {
    return memo->u1.value;
    }
    if (!NIL_P(if_none)) {
    return rb_funcall(if_none, id_call, 0, 0);
    }
    return Qnil;
}
```

In practice, it's doing it more like:

```
module Enumerable
  def detect3
    return enum_for(:detect3) unless block_given?
    inject(nil) { |m, x|
      v = yield x
      m ||= v if v
      m
    }
  end
end
```

#3 - 09/23/2020 10:01 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Rejected