Ruby master - Bug #16414

Incompatible behavior of Proc/lambda with single argument when using `Enumerator::Lazy#with_index`

12/10/2019 01:09 PM - tomog105 (Tomohiro Ogoke)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.7.0dev (2019-12-10T10:12:21Z master af11efd377) [x86_64-darwin18]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

The following code raised an error wrong number of arguments (given 1, expected 2) (ArgumentError) in master.

```
$ ruby -e 'lambda = -> (s, i) { "#{i}:#{s}" }; p %w(a b c).each.lazy.with_index.map(&lambda).first(2)'
# expected result => ["0:a", "1:b"]
```

This code is valid up till Ruby 2.6.5 and 2.7.0-preview1, but it raised the error in Ruby 2.7.0-preview2 or later. Maybe, this behavior has been there since the implementation of Enumerator::Lazy#with_index.

Is this behaviour intended?

Associated revisions

Revision 85e43e1d - 12/11/2019 02:59 AM - jeremyevans (Jeremy Evans)

Fix Enumerator::Lazy#with_index

- Make it correctly handle lambdas
- Make it iterate over the block if block is given

The original implementation was flawed, based on lazy_set_method instead of lazy_add_method.

Note that there is no implicit map when passing a block, the return value of the block passed to with_index is ignored, just as it is for Enumerator#with_index. Also like Enumerator#with_index, when called with a block, the return value is an enumerator without the index.

Fixes [Bug #16414]

History

#1 - 12/10/2019 03:31 PM - sawa (Tsuyoshi Sawada)

- Description updated

#2 - 12/10/2019 08:27 PM - jeremyevans0 (Jeremy Evans)

I agree this is a bug. The with_index block should be called with two arguments, not an array with one argument.

The reason this happens is that Enumerator::Lazy#with_index ends up calling Enumerator::Yielder#<< (yielder_yield_push C function) on the yielder with an array with two entries. That in turn calls rb_proc_call_with_block(ptr->proc, 1, &arg, Qnil);. This is why the block gets called with a single array argument.

After experimenting more, Enumerator::Lazy#with_index has other issues. The most critical is it ignores the block passed to it. I'm fairly sure it needs to be rewritten. I'll see if I can fix the problems with it. If the problems can't be fixed, we should probably back out the changes, so that Enumerator::Lazy#with_index is no longer lazy.

#3 - 12/10/2019 11:23 PM - jeremyevans0 (Jeremy Evans)

I have a possible fix for this: https://github.com/ruby/ruby/pull/2742

After CI completes, I'll try to merge it so it makes 2.7.0-rc1.

#4 - 12/11/2019 03:00 AM - jeremyevans (Jeremy Evans)

- Status changed from Open to Closed

Applied in changeset git|85e43e1dfecef69b935c48c235cc20f21bd4f0d4.

Fix Enumerator::Lazy#with_index

- Make it correctly handle lambdas
- Make it iterate over the block if block is given

The original implementation was flawed, based on lazy_set_method
instead of lazy_add_method.

Note that there is no implicit map when passing a block, the return
value of the block passed to with_index is ignored, just as it
is for Enumerator#with_index. Also like Enumerator#with_index,
when called with a block, the return value is an enumerator without
the index.

Fixes [Bug #16414]