Ruby master - Bug #16829

Exceptions raised from within an enumerated method lose part of their stacktrace

05/04/2020 09:15 PM - doliveirakn (Kyle d'Oliveira)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.7.1	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

Consider the following code:

```
class Test
  include Enumerable

  def each(&block)
    raise "Boom"
  end
end

def execution_method_a
  Test.new.to_enum(:each).next
end

def execution_method_b
  Test.new.each do
    # Never gets run
  end
end

begin
  execution_method_a
rescue RuntimeError => e
  puts "Using to_enum and next"
  puts e.message
  puts e.backtrace
end


begin
  execution_method_b
rescue RuntimeError => e
  puts "Calling a block directly"
  puts e.message
  puts e.backtrace
end
```

When this file (located at lib/script.rb) is run the result is:

```
Using to_enum and next
Boom
lib/script.rb:5:in `each'
lib/script.rb:1:in `each'
Calling a block directly
Boom
lib/script.rb:5:in `each'
lib/script.rb:14:in `execution_method_b'
lib/script.rb:29:in `<main>'
```

This is a little unusual. Effectively, if we create an enumerator and use next to iterate through the results, the backtrace is modified to the point where the calling method(s) are entirely lose. Notice when the each method is used directly and an exception is thrown, we

see execution_method_b present in the stacktrace, but if we use next we do not see execution_method_a present at all.

This means that if there is some code that uses the enumerator/next approach deep within a callstack, the exception that comes out does not have any crucial information of where the call originated from.

History

#1 - 05/05/2020 08:36 PM - marcandre (Marc-Andre Lafortune)

I believe this is due to the fact that next's implementation uses a Fiber.

If you use to_a instead of next, you will get the stacktrace you were hoping for.

If you replace your definition of execution_method_a with:

```
$fiber = Fiber.new do
  raise 'Boom'
end

def execution_method_a
  $fiber.resume
end
```

You'll also get a shorter stack trace then you'd like; the exception is raised within a fiber and doesn't know where it was resumed from.

It would be a good idea to specify this in the doc of next, peek, etc.

#2 - 05/05/2020 09:06 PM - marcandre (Marc-Andre Lafortune)

I've made by best to improve the documentation in 7bde981.

nobu (Nobuyoshi Nakada): is it feasible to improve stacktraces raised within fibers?