Ruby - Bug #20807

String#gsub fails when called from string subclass with a block passed

10/21/2024 11:07 AM - koilanetroc (Oleg Tolmashov)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.3.4 (2024-07-09 revision be1089c8ec) [arm64-darwin23]	**Backport:**	3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

When String#gsub is called from a string subclass with a block, Regexp.last_match is nil, but passed block is executed. Here is example code:

```
def call_gsub(str)
  str.gsub(/%/) do
    puts "checking #{str.class}"
    puts "Special variable value: #{$&}"
    puts "Regexp.last_match = #{Regexp.last_match.inspect}\n\n"

    raise "Special variable $& is not assigned, but block is called" if $&.nil?
  end
end

class MyString < String
  def gsub(*args, &block)
    super(*args, &block) # just forward everything
  end
end

text = 'test%text_with_special_character'

call_gsub(String.new(text)) # original string
call_gsub(MyString.new(text)) # string subclass
```

Result:

```
checking String
Special variable value: %
Regexp.last_match = #<MatchData "%">

checking MyString
Special variable value:
Regexp.last_match = nil

gsub_bug.rb:7:in `block in call_gsub': Special variable $& is not assigned, but block is called (R
untimeError)
 from gsub_bug.rb:13:in `gsub'
 from gsub_bug.rb:13:in `gsub'
 from gsub_bug.rb:2:in `call_gsub'
 from gsub_bug.rb:20:in `<main>'
```

I expect result to be the same for both classes since MyString just wraps the same method:

```
checking String
Special variable value: %
Regexp.last_match = #<MatchData "%">

checking MyString
Special variable value: %
Regexp.last_match = #<MatchData "%">
```

Maybe there is something off with with control frame when params are forwarded?

Thanks in advance!

Related issues:

Related to Ruby - Bug #8444: Regexp vars $~ and friends are not thread local	**Assigned**
Related to Ruby - Bug #12689: Thread isolation of $~ and $_	**Open**
Related to Ruby - Bug #14364: Regexp last match variable in procs	**Open**
Related to Ruby - Bug #11808: Different behavior between Enumerable#grep and ...	**Assigned**

History

#1 - 10/23/2024 03:03 AM - Dan0042 (Daniel DeLorme)

Regexp.last_match and other regexp-related pseudo globals do not work across more than one stack frame. Since you override #gsub, they are set only inside MyString#gsub
You can confirm with this:

```
def test(klass)
  p klass
  klass.new("test").gsub(/s/,'x')
  p result: $~
end

class MyString1 < String
end
test(MyString1)
#prints:
#{:result=>#<MatchData "s">}

class MyString2 < String
  def gsub(...)
    super
  ensure
    p ensure: $~
  end
end
test(MyString2)
#prints:
#{:ensure=>#<MatchData "s">}
#{:result=>nil}
```

It would be possible to fix this by propagating Regexp.last_match up every "super" stack frame until we reach the originating non-super frame. It would allow some interesting use cases (like logging the time spent in every Regexp#match). But it's a lot of work for a very niche use.

#2 - 11/18/2024 04:37 AM - jeremyevans0 (Jeremy Evans)

- Related to Bug #8444: Regexp vars $~ and friends are not thread local added

- Related to Bug #12689: Thread isolation of $~ and $_ added

- Related to Bug #14364: Regexp last match variable in procs added

#3 - 11/18/2024 04:38 AM - jeremyevans0 (Jeremy Evans)

- Related to Bug #11808: Different behavior between Enumerable#grep and Array#grep added