Ruby - Bug #19003

TracePoint behavior inconsistency in 3.2.0-preview2

09/09/2022 09:17 AM - hurricup (Alexandr Evstigneev)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.2.0preview2 (2022-09-09 master 35cfc9a3bb) [x86_64-linux]	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN

Description

This is kind of continuation of my previous report about global/local TP processing (#18730).

Sample script:

```
def foo
  return 1
end

puts RubyVM::InstructionSequence.of(method :foo).disasm

def step_over
  TracePoint.new(:line, :return, :b_return) do |tp|
    puts "Step over hits by #{tp.event} at #{tp.lineno}"
    step_over
    tp.disable
  end.enable(target: RubyVM::InstructionSequence.of(method :foo), target_thread: Thread.current)
end

TracePoint.new(:line, :return, :b_return) do |tp|
  if tp.lineno == 2
    puts "Step into hits by #{tp.event} at #{tp.lineno}"
    step_over
    tp.disable
  end
end.enable(target_thread: Thread.current)

a = foo
```

In ruby 3.1.2 we have expected behavior. Output:

```
== disasm: #<ISeq:foo@/home/hurricup/Projects/ruby-debugger/jb-debase-30/test_sample.rb:1 (1,0)-(3
,3)> (catch: FALSE)
0000 putobject_INT2FIX_1_                                             (   2)[LiCa]
0001 leave                                                            (   3)[Re]
Step into hits by line at 2
Step over hits by return at 3
```

In ruby 3.2.0-preview2 - not so much. Output:

```
== disasm: #<ISeq:foo@/home/hurricup/Projects/ruby-debugger/jb-debase-30/test_sample.rb:1 (1,0)-(3
,3)> (catch: false)
0000 putobject_INT2FIX_1_                                             (   2)[LiCa]
0001 leave                                                            (   3)[Re]
Step into hits by line at 2
Step over hits by line at 2
Step over hits by return at 3
```

History

#1 - 09/09/2022 09:22 AM - hurricup (Alexandr Evstigneev)

- Description updated

#2 - 09/16/2022 05:58 PM - jeremyevans0 (Jeremy Evans)

This issue shows a case where you are adding a local tracepoint during global tracepoint processing. There are a couple approaches Ruby could take here:

1. For a local tracepoint added during global tracepoint processing, only call it for future events and not the current event. This appears to be the Ruby 3.1 behavior.

2. For a local tracepoint added during global tracepoint processing, call it for future events and the current event. This appears to be the Ruby 3.2 behavior.

In both cases, a local tracepoint added during local tracepoint processing (the recursive call inside step_over) is not called for the current event. If it was, the code example would result in an infinite loop.

Can you explain why you think this behavior change is a bug? Does Ruby specify that local tracepoints added during global tracepoint processing for an event should not apply to the current event, and only future events?

#3 - 09/17/2022 05:06 AM - hurricup (Alexandr Evstigneev)

jeremyevans0 (Jeremy Evans) wrote in #note-2:

I just saying this behavior is inconsistent between versions.

Current behavior looks like a bug to me, because it feels like only hooks already set at the moment of the event arrival should handle it, not ones set while processing the event by another hook.

If API user need some additional actions to be performed on the current event, he can achieve that by additional logic in the hook. I can't imagine any case where current ruby behavior may be beneficial in any way. And from other side it's unintuitive and confusing.

And scope of the hook (local/global) should not affect behavior in any way. This should be implementation details without any guaranteed order or behavior specifics.

For API user it's just a TracePoint, with or without constructor arguments.

#4 - 11/29/2022 06:11 AM - hurricup (Alexandr Evstigneev)

This stays the same in the preview3. Are there any chances this will change until release?

Current behavior reduces usefulness and complicates usage of local tracepoints, because I can't just set smarter TP, but also I need to think about what fires when and in which cases we have chained events handling. This is really frustrating.

#5 - 12/01/2022 08:08 AM - ko1 (Koichi Sasada)

Sorry for late.

We left this issue as an implementation details and the current behavior will be shipped with Ruby 3.2. Sorry for inconvenient for your purpose.

The reason is to implement the proposed behavior strictly needs a much work and we can't do that now. In my opinion the proposal behavior is preferable, so it can be fixed in future.

BTW on the debug.gem we avoid duplicated TracePoint manually.

#6 - 12/01/2022 08:08 AM - ko1 (Koichi Sasada)

- Status changed from Open to Rejected