Ruby master - Bug #16776

Regression in coverage library

04/10/2020 04:39 PM - deivid (David Rodríguez)

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

Description

Hi!

I noticed a regression in the coverage library. I tried to write a minimal program to show it, hopefully it gives some clues or where the issue might lie.

In ruby 2.5.8 and earlier, the following program would print {:lines=>[1, 1, nil]}, showing that the body of the "foo" method was run once. However, on newer rubies, it prints {:lines=>[1, 0, nil]}, which is incorrect because the "foo" method body has actually been run once.

This is the repro script:

```
# frozen_string_literal: true

require "coverage"
Coverage.start(lines: true)

code = <<~RUBY
  def foo
    "LOL"
  end
RUBY

File.open("foo.rb", "w") { |f| f.write(code) }

require_relative "foo"

TracePoint.new(:line) do |_tp|
  foo
end.enable do
  sleep 0
end

res = Coverage.result
puts res[File.expand_path("foo.rb")]
```

History

#1 - 04/10/2020 05:20 PM - mame (Yusuke Endoh)

Thank you for your report. The reason why the method execution is missed is because:

- coverage measurement is based on TracePoint mechanism (since 2.6), and
- no TracePoint event is fired during the execution of TracePoint hooks (to avoid infinite chain of hooks).

So, this is a limitation of TracePoint. Currently, the combination of multiple TracePoint hooks works not so great; they are completely exclusive.

This issue may be solved by improving TracePoint mechanism, but it is expensive to implement and ko1 (Koichi Sasada) is negative. How severe is this issue for you?

#2 - 04/10/2020 05:25 PM - deivid (David Rodríguez)

I see.

It's a severe as "I can no longer track code coverage on my library that makes heavy use of TracePoint" is considered.

(Edited the message to add thank you for your response and your work in general :))

#3 - 04/12/2020 05:32 AM - marcandre (Marc-Andre Lafortune)

I never realized this limitation. I imagine that for the vast majority of users it's a non issue, but for byebug it is quite serious. Would you consider using deep-cover instead?

#4 - 04/12/2020 06:27 AM - marcandre (Marc-Andre Lafortune)

I'm not super familiar with TracePoint, but at first sight DeepCover shouldn't generate any extra callbacks as the instrumenting is minimal, typically adding $tracker[42] += 1 and taking care to never change the line numbers.

So it should work even with TracePoint enabled and shouldn't impact your tests. I tested quickly with DeepCover's clone mode and confirm that your test suite appear to pass :-)

#5 - 04/12/2020 01:14 PM - Eregon (Benoit Daloze)

I'm surprised too the coverage stdlib gets hit by this issue.

Maybe coverage could use a second type of internal tracepoint to fix this issue (internal coverage handlers are likely pure C and so don't emit any extra event)?

#6 - 04/12/2020 02:01 PM - mame (Yusuke Endoh)

IMO, TracePoint hooks should have a priority. A TracePoint hook that is first enabled should monitor a hook code that is second enabled:

```
1: TracePoint.new(:line) do |tp|
2:   p [:parent, tp]
3: end.enable do
4:   TracePoint.new(:line) do |tp|
5:     p [:child, tp]
6:   end.enable do
7:     p :main
8:   end
9: end
```

```
$ ruby t.rb
[:parent, #<TracePoint:line@t.rb:4>]
[:parent, #<TracePoint:line@<internal:trace_point>:96 in `new'>]
[:parent, #<TracePoint:line@<internal:trace_point>:196 in `enable'>]
[:child, #<TracePoint:line@t.rb:7>]
[:parent, #<TracePoint:line@t.rb:7>]
:main
```

I expect [:parent, #<TracePoint:line@t.rb:5>] before [:child, #<TracePoint:line@t.rb:7>]. ko1 (Koichi Sasada), what do you think?

#7 - 04/12/2020 02:04 PM - mame (Yusuke Endoh)

Eregon (Benoit Daloze) wrote in #note-5:

> Maybe coverage could use a second type of internal tracepoint to fix this issue (internal coverage handlers are likely pure C and so don't emit any extra event)?

Yes, it is the plan B that I have. I don't like it as it is very ad-hoc, though. If ko1 (Koichi Sasada) rejects the priority, I'll give it a try to implement.

#8 - 04/18/2020 02:31 PM - deivid (David Rodríguez)

marcandre (Marc-Andre Lafortune) wrote in #note-3:

> I never realized this limitation. I imagine that for the vast majority of users it's a non issue, but for byebug it is quite serious. Would you consider using deep-cover instead?

It sounds like the plan is to fix this one way or another, so I'll wait for now. But thanks for pointing me to deep-cover, it sounds interesting!