Ruby master - Bug #16889

TracePoint.enable { ... } also activates the TracePoint for other threads, even outside the block

05/14/2020 01:37 PM - Eregon (Benoit Daloze)

Status:	Open		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	ruby 2.6.6p146 (2020-03-31 revision 67876) [x86_64-linux]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

```
threads = []
inspects = []
trace = TracePoint.new(:line) do |tp|
  threads << Thread.current
  inspects << tp.inspect
end

done = false
thread = Thread.new do
  Thread.pass until done
end

trace.enable do
  line_event = true
  done = true
  sleep 1
end
thread.join

# Expected only within enable block (lines 14-16)
puts inspects

# Expected just 1
p threads.uniq
```

Results in:

```
$ ruby tpbug.rb
ruby tpbug.rb
#<TracePoint:line@tpbug.rb:14>
#<TracePoint:line@tpbug.rb:15>
#<TracePoint:line@tpbug.rb:16>
#<TracePoint:line@tpbug.rb:10>
[#<Thread:0x00005571134e3340 run>, #<Thread:0x00005571138ac828@tpbug.rb:9 dead>]
```

But I expected:

```
#<TracePoint:line@tpbug.rb:14>
#<TracePoint:line@tpbug.rb:15>
#<TracePoint:line@tpbug.rb:16>
[#<Thread:0x00005571134e3340 run>]
```

Because the RDoc says:

```
If a block is given, the trace will only be enabled within the scope of
the block.
```

For background I'm trying to improve the TracePoint specs in ruby/spec, but they are proving quite unreliable due to this.

ko1 (Koichi Sasada) Thoughts?

History

#1 - 05/14/2020 01:40 PM - Eregon (Benoit Daloze)

- Description updated

#2 - 05/14/2020 01:44 PM - Eregon (Benoit Daloze)

Maybe enable(&block) should behave like enable(target: block); disable?