Ruby - Bug #19363

Fix rb_transient_heap_mark: wrong header (T_STRUCT) segfault

01/21/2023 06:53 PM - bkuhlmann (Brooke Kuhlmann)

Status:	Assigned		
Priority:	Normal		
Assignee:	mame (Yusuke Endoh)		
Target version:			
ruby -v:	ruby 3.2.0 (2022-12-25 revision a528908271) +YJIT [arm64-darwin22.2.0]	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN

Description

Overview

Hello. I'm hitting an issue where my build is constantly failing with a segfault. The following is a snippet taken from my local machine with YJIT enabled (see attachments for details):

```
/Users/bkuhlmann/.cache/frum/versions/3.2.0/lib/ruby/gems/3.2.0/gems/puma-6.0.2/lib/puma/runner.rb
: [BUG] rb_transient_heap_mark: wrong header, T_STRUCT (0x0000000109ea98a0)
ruby 3.2.0 (2022-12-25 revision a528908271) +YJIT [arm64-darwin22.2.0]
```

The closest issue I could find that might be related to this issue (but not sure) is this issue: #15358.

Steps to Recreate

You should be able to quickly recreate this issue via these steps:

- Download/clone my Hemo project.
- Run the setup steps.
- Run the test suite by running bin/rspec.

If you need an example of the same segfault (but not on my macOS machine), you can see the same segfault via my Circle CI Build. My Circle CI build is using my Docker Alpine Linux Ruby image which might be of interest as well. This Docker image is also built with YJIT enabled.

Interestingly, is if you were to run the test suite with bin/guard instead of bin/rspec then the segfault doesn't occur.

Environment

```
ruby 3.2.0 (2022-12-25 revision a528908271) +YJIT [arm64-darwin22.2.0]

1.43.0 (using Parser 3.2.0.0, rubocop-ast 1.24.1, running on ruby 3.2.0) [arm64-darwin22.2.0]
  - rubocop-performance 1.15.2
  - rubocop-rake 0.6.0
  - rubocop-rspec 2.18.1
  - rubocop-sequel 0.3.4
  - rubocop-thread_safety 0.4.4
```

History

#1 - 01/21/2023 09:02 PM - alanwu (Alan Wu)

Thanks for the report, and for the comprehensive reproduction steps.
Triage note, it seems this issue can happen on x86_64-linux-musl without YJIT at runtime.
The crash logs there in the Circle CI build does not say +YJIT.

#2 - 01/29/2023 02:45 PM - bkuhlmann (Brooke Kuhlmann)

I was able to narrow down where this bug is occuring. Turns out that when enabling eval in SimpleCov, the segfault consistently happens. Here's the code in question as found in the spec_helper.rb of the above application:

```
unless ENV["NO_COVERAGE"]
  SimpleCov.start do
```

```
    add_filter %r(^/spec/)
    enable_coverage :branch
    enable_coverage_for_eval  # <-- When this is enabled, the segmentation fault consistently occurs.
    minimum_coverage_by_file line: 95, branch: 95
  end
end
```

If the SimpleCov enable_coverage_for_eval statement is removed entirely, then there is no segmentation fault.

#3 - 03/26/2023 10:37 AM - wanabe (_ wanabe)

- File segv.log added

I made a short reproduction code.
There are three points:

- unexpected negative lineno for eval (or for class_eval)
- Coverage.start with lines: true, eval: true
- GC verification

```
require "coverage"

Coverage.start(lines: true, eval: true)
eval(<<~EOS, binding, "", -1)
  Kernel.module_eval do
    def bar(locals)
      bar = locals[:bar]
    end
  end
EOS
bar({})
GC.verify_compaction_references
```

And I attached SEGV log on ruby 3.3.0dev (2023-03-26T06:23:11Z master 2f916812a9) [x86_64-linux] + WSL2.

#4 - 03/27/2023 09:45 AM - wanabe (_ wanabe)

- Assignee set to mame (Yusuke Endoh)

I guess that update_line_coverage() does not assume negative line numbers.
https://bugs.ruby-lang.org/projects/ruby-master/repository/git/revisions/v3_2_1/entry/thread.c#L5512

(I think ideally eval should reject negative line numbers.
However, it is not a bug but feature request, and it may be compatibility issues.)

mame-san, can you have a look at the issue?
Please allow me to assign it to you.

#5 - 03/27/2023 02:23 PM - jeremyevans0 (Jeremy Evans)

wanabe (_ wanabe) wrote in #note-4:

> (I think ideally eval should reject negative line numbers.
> However, it is not a bug but feature request, and it may be compatibility issues.)

tilt (~445 million downloads, dependency of Sinatra) uses eval with negative line numbers, so that the line numbers reported in back traces match the lines in the user-provided template file, even though tilt adds lines before those lines. There would be significant backwards compatibility issues if negative line numbers were removed.

#6 - 04/10/2023 07:30 AM - Eregon (Benoit Daloze)

IMO it would be good to deprecate negative line numbers, they are a hack that's pretty ugly to replicate in Ruby implementations.
Tilt could just use prelude_code; original_first_line, no?

#7 - 04/10/2023 07:36 AM - byroot (Jean Boussier)

@Eregon (Benoit Daloze) that doesn't work if your prelude contains magic comments, typically # frozen_string_literal: true

#8 - 04/10/2023 08:21 AM - Eregon (Benoit Daloze)

Ah, yes. Maybe eval could accept such magic comments as keyword arguments or so to do it more cleanly.
eval already does a bit of that by using the String's encoding as source encoding (if no encoding magic comment used).

#9 - 04/03/2024 03:50 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Assigned

Files

segfault.txt	237 KB	01/21/2023	bkuhlmann (Brooke Kuhlmann)
ruby-2023-01-21-113841.ips	19.6 KB	01/21/2023	bkuhlmann (Brooke Kuhlmann)
segv.log	14.7 KB	03/26/2023	wanabe (_ wanabe)