Ruby trunk - Bug #5047

Segfault (most likely involving require)

07/19/2011 03:47 AM - jackc (Jack Christensen)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	1.9.3		
ruby -v:	-	**Backport:**	

Description

Rails application randomly segfaults when running rspec. It will segfault about 80% of the time before any specs run. It seems to be related to requiring files. If it succeeds in getting to any specs it will run all of them without segfaulting.

The catalyst for the segfault seems to be the prawn gem. I have tested several rails applications where adding prawn to the Gemfile crashes them. However, there need to be certain other gems in place as well.

source 'http://rubygems.org'

gem 'rails', '3.1.0.rc4'

gem 'tiny_tds'
gem 'activerecord-sqlserver-adapter', :git => 'git://github.com/rails-sqlserver/activerecord-sqlserver-adapter.git'

group :test do
gem 'rspec-rails', '>= 2.6.0'
end

gem "prawn"

This Gemfile will crash most of the time when running bundle exec rspec spec. If I replace the tiny_tds and activerecord-sql-server-adapter with sqlite3 it does not crash. But I do not believe the fault is with the SQL server gems as in my other rails projects I am using the pg gem and they also crash.

Another interesting point is the order of the Gemfile matters. If I move the prawn gem to the top of the Gemfile the segfaults do not occur.

The crashes occur at multiple locations, but the most most common is:
activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237.

I've put a whole stack dump below.

/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237: [BUG] Segmentation fault
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-linux]

-- control frame ----------
c:0042 p:---- s:0143 b:0143 l:000142 d:000142 CFUNC :require
c:0041 p:0012 s:0139 b:0139 l:000131 d:000138 BLOCK
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0040 p:0054 s:0137 b:0137 l:000136 d:000136 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225
c:0039 p:0013 s:0132 b:0132 l:000131 d:000131 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0038 p:0119 s:0127 b:0127 l:000126 d:000126 TOP
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-mocks-2.6.0/lib/rspec/mocks/framework.rb:14
c:0037 p:---- s:0125 b:0125 l:000124 d:000124 FINISH
c:0036 p:---- s:0123 b:0123 l:000122 d:000122 CFUNC :require
c:0035 p:0012 s:0119 b:0119 l:000111 d:000118 BLOCK
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0034 p:0054 s:0117 b:0117 l:000116 d:000116 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225
c:0033 p:0013 s:0112 b:0112 l:000111 d:000111 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237

c:0032 p:0011 s:0107 b:0107 l:000106 d:000106 TOP /usr/local/lib/ruby/gems/1.9.1/gems/rspec-mocks-2.6.0/lib/rspec/mocks.rb:1
c:0031 p:---- s:0105 b:0105 l:000104 d:000104 FINISH
c:0030 p:---- s:0103 b:0103 l:000102 d:000102 CFUNC :require
c:0029 p:0012 s:0099 b:0099 l:000091 d:000098 BLOCK
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0028 p:0054 s:0097 b:0097 l:000096 d:000096 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225
c:0027 p:0013 s:0092 b:0092 l:000091 d:000091 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0026 p:0011 s:0087 b:0087 l:000086 d:000086 TOP
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/mocking/with_rspec.rb:1
c:0025 p:---- s:0085 b:0085 l:000084 d:000084 FINISH
c:0024 p:---- s:0083 b:0083 l:000082 d:000082 CFUNC :require
c:0023 p:0012 s:0079 b:0079 l:000071 d:000078 BLOCK
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0022 p:0054 s:0077 b:0077 l:000076 d:000076 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225
c:0021 p:0013 s:0072 b:0072 l:000071 d:000071 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237
c:0020 p:0186 s:0067 b:0067 l:000066 d:000066 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:176
c:0019 p:0016 s:0063 b:0062 l:000061 d:000061 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:135
c:0018 p:0013 s:0058 b:0058 l:000051 d:000057 BLOCK /home/jackc/work/segfault/spec/spec_helper.rb:18
c:0017 p:0062 s:0055 b:0055 l:000054 d:000054 METHOD /usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core.rb:79
c:0016 p:0125 s:0052 b:0052 l:000051 d:000051 TOP /home/jackc/work/segfault/spec/spec_helper.rb:10
c:0015 p:---- s:0050 b:0050 l:000049 d:000049 FINISH
c:0014 p:---- s:0048 b:0048 l:000047 d:000047 CFUNC :require
c:0013 p:0011 s:0044 b:0044 l:000043 d:000043 TOP /home/jackc/work/segfault/spec/controllers/application_controller_spec.rb:1
c:0012 p:---- s:0042 b:0042 l:000041 d:000041 FINISH
c:0011 p:---- s:0040 b:0040 l:000039 d:000039 CFUNC :load
c:0010 p:0025 s:0036 b:0036 l:000027 d:000035 BLOCK
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:419
c:0009 p:---- s:0033 b:0033 l:000032 d:000032 FINISH
c:0008 p:---- s:0031 b:0031 l:000030 d:000030 CFUNC :map
c:0007 p:0017 s:0028 b:0028 l:000027 d:000027 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:419
c:0006 p:0074 s:0025 b:0025 l:000024 d:000024 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/command_line.rb:18
c:0005 p:0055 s:0020 b:0020 l:000019 d:000019 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/runner.rb:80
c:0004 p:0101 s:0014 b:0014 l:000013 d:000013 METHOD
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/runner.rb:69
c:0003 p:0021 s:0007 b:0006 l:002108 d:000005 BLOCK
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/runner.rb:11
c:0002 p:---- s:0004 b:0004 l:000003 d:000003 FINISH

c:0001 p:0000 s:0002 b:0002 l:000688 d:000688 TOP

-- Ruby level backtrace information --
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/runner.rb:11:in block in autorun'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/runner.rb:69:inrun'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/runner.rb:80:in run_in_process'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/command_line.rb:18:inrun'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:419:in load_spec_files'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:419:inmap'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:419:in block in load_spec_files'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:419:inload'
/home/jackc/work/segfault/spec/controllers/application_controller_spec.rb:1:in <top (required)>'
/home/jackc/work/segfault/spec/controllers/application_controller_spec.rb:1:inrequire'
/home/jackc/work/segfault/spec/spec_helper.rb:10:in <top (required)>'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core.rb:79:inconfigure'
/home/jackc/work/segfault/spec/spec_helper.rb:18:in block in <top (required)>'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:135:inmock_with'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/configuration.rb:176:in mock_framework='
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:inrequire'

```
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225:in load_dependency'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:inblock in require'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:in require'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-core-2.6.4/lib/rspec/core/mocking/with_rspec.rb:1:in'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:in require'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225:inload_dependency'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:in block in require'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:inrequire'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-mocks-2.6.0/lib/rspec/mocks.rb:1:in <top (required)>'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:inrequire'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225:in load_dependency'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:inblock in require'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:in require'
/usr/local/lib/ruby/gems/1.9.1/gems/rspec-mocks-2.6.0/lib/rspec/mocks/framework.rb:14:in'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:in require'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:225:inload_dependency'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:in block in require'
/usr/local/lib/ruby/gems/1.9.1/gems/activesupport-3.1.0.rc4/lib/active_support/dependencies.rb:237:inrequire'

-- C level backtrace information -------------------------------------------
/usr/local/bin/ruby(rb_vm_bugreport+0x9e) [0x523b6e]
/usr/local/bin/ruby() [0x565508]
/usr/local/bin/ruby(rb_bug+0xb1) [0x5656a1]
/usr/local/bin/ruby() [0x4b2ca8]
/lib/libpthread.so.0(+0xf8f0) [0x7fa523e988f0]
/usr/local/bin/ruby(st_free_table+0x43) [0x4bbb33]
/usr/local/bin/ruby() [0x426cf5]
/usr/local/bin/ruby() [0x4277ed]
/usr/local/bin/ruby() [0x4c0562]
/usr/local/bin/ruby(rb_usascii_str_new+0x13) [0x4c0683]
/usr/local/bin/ruby(rb_file_expand_path+0x42) [0x5706c2]
/usr/local/bin/ruby() [0x56a63a]
/usr/local/bin/ruby(rb_require_safe+0x25d) [0x56aa8d]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x5168c1]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_iseq_eval+0x1ee) [0x519aee]
/usr/local/bin/ruby() [0x569a08]
/usr/local/bin/ruby(rb_require_safe+0x647) [0x56ae77]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x5168c1]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_iseq_eval+0x1ee) [0x519aee]
/usr/local/bin/ruby() [0x569a08]
/usr/local/bin/ruby(rb_require_safe+0x647) [0x56ae77]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x5168c1]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_iseq_eval+0x1ee) [0x519aee]
/usr/local/bin/ruby() [0x569a08]
/usr/local/bin/ruby(rb_require_safe+0x647) [0x56ae77]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x5168c1]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_iseq_eval+0x1ee) [0x519aee]
/usr/local/bin/ruby() [0x569a08]
/usr/local/bin/ruby(rb_require_safe+0x647) [0x56ae77]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x515546]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_iseq_eval+0x1ee) [0x519aee]
/usr/local/bin/ruby() [0x569a08]
/usr/local/bin/ruby() [0x569b6b]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x515546]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_yield+0x66) [0x521d56]
```

```
/usr/local/bin/ruby() [0x53ac15]
/usr/local/bin/ruby() [0x51307f]
/usr/local/bin/ruby() [0x515546]
/usr/local/bin/ruby() [0x519499]
/usr/local/bin/ruby(rb_vm_invoke_proc+0x9f) [0x51c38f]
/usr/local/bin/ruby(rb_exec_end_proc+0x238) [0x41c5b8]
/usr/local/bin/ruby() [0x41c684]
/usr/local/bin/ruby(ruby_cleanup+0x12d) [0x41c7fd]
/usr/local/bin/ruby(ruby_run_node+0x3d) [0x41cb0d]
/usr/local/bin/ruby(main+0x49) [0x419c49]
/lib/libc.so.6(__libc_start_main+0xfd) [0x7fa52325cc4d]
/usr/local/bin/ruby() [0x419b39]
```

[NOTE]
You may have encountered a bug in the Ruby interpreter or extension libraries.
Bug reports are welcome.
For details: http://www.ruby-lang.org/bugreport.html

Aborted

I've attached the minimal rails test app I could get to fail below.

Associated revisions

Revision 9a272395 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@32669 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 32669 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

Revision 32669 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

Revision 32669 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

Revision 32669 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

Revision 32669 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

Revision 32669 - 07/25/2011 02:29 PM - mame (Yusuke Endoh)

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct

METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

Revision 43e8a7a3 - 07/28/2011 03:50 PM - mame (Yusuke Endoh)

- backport r32669 from trunk.
- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_1_9_3@32728 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 07/21/2011 10:47 AM - nahi (Hiroshi Nakamura)

- Category set to core

- Status changed from Open to Assigned

- Assignee set to nahi (Hiroshi Nakamura)

- Target version set to 1.9.2

Confirmed the SEGV on my machine. I'll investigate it.

#2 - 07/21/2011 10:55 AM - nahi (Hiroshi Nakamura)

- Target version changed from 1.9.2 to 1.9.3

I get SEGV with ruby 1.9.3dev (2011-07-21 revision 32592) [x86_64-linux] as well. Still investigating.

Note: Don't forget to backport when we fixes it.

#3 - 07/21/2011 01:57 PM - kosaki (Motohiro KOSAKI)

- Priority changed from Normal to 5

#4 - 07/22/2011 11:47 AM - nahi (Hiroshi Nakamura)

Can someone enlighten me to find the bug?

Here's a way to replicate the bug. 4 steps. I'm on Ubuntu 11.04 64bit.

1. extract the attached app

% unzip segfault.zip
% cd segfault

2. install bundler via rubygems

% gem install bundler --user-install
% export PATH=$PATH:/home/nahi/.gem/ruby/1.9.1/bin

3. run bundle to install gems to ./ruby

% bundle install --path=.

4. execute it

% ruby -rbundler -e 'Bundler.setup; require "rspec/core"; RSpec::Core::Runner.run(["spec"], $stderr, $stdout)'

And here's my gdb session.

```
% gdb /usr/local/bin/ruby
GNU gdb (Ubuntu/Linaro 7.2-1ubuntu11) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later http://gnu.org/licenses/gpl.html
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
http://www.gnu.org/software/gdb/bugs/...
Reading symbols from /usr/local/bin/ruby...done.
(gdb) run -rbundler -e 'Bundler.setup; require "rspec/core"; RSpec::Core::Runner.run(["spec"], $stderr, $stdout)'
Starting program: /usr/local/bin/ruby -rbundler -e 'Bundler.setup; require "rspec/core"; RSpec::Core::Runner.run(["spec"], $stderr, $stdout)'
[Thread debugging using libthread_db enabled]
[New Thread 0x7ffff6682700 (LWP 13255)]
[Thread 0x7ffff6682700 (LWP 13255) exited]
[New Thread 0x7ffff6682700 (LWP 13257)]

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7ab5c99 in st_foreach (table=0x28b5020, func=0x7ffff7a0c230 , arg=140737488315296) at ../st.c:745
745             key = (st_data_t)table->bins[i*2];
(gdb) where
#0  0x00007ffff7ab5c99 in st_foreach (table=0x28b5020, func=0x7ffff7a0c230 , arg=140737488315296) at ../st.c:745
#1  0x00007ffff7a0be88 in mark_tbl (objspace=0x6038a0, ptr=19563040, lev=5) at ../gc.c:1445
#2  gc_mark_children (objspace=0x6038a0, ptr=19563040, lev=5) at ../gc.c:1801
#3  0x00007ffff7a0b8d8 in gc_mark_children (objspace=0x6038a0, ptr=35333200, lev=4) at ../gc.c:1795
#4  0x00007ffff7a0b8d8 in gc_mark_children (objspace=0x6038a0, ptr=35333280, lev=3) at ../gc.c:1795
#5  0x00007ffff7a0c1d6 in mark_const_entry_i (key=, ce=, data=)
at ../gc.c:1562
#6  0x00007ffff7ab5d1c in st_foreach (table=0x19131a0, func=0x7ffff7a0c1c0 , arg=140737488315616)
at ../st.c:787
#7  0x00007ffff7a0bead in mark_const_tbl (objspace=0x6038a0, ptr=10364280, lev=2) at ../gc.c:1573
#8  gc_mark_children (objspace=0x6038a0, ptr=10364280, lev=2) at ../gc.c:1802
#9  0x00007ffff7a0bb7e in gc_mark_children (objspace=0x6038a0, ptr=27521040, lev=1) at ../gc.c:1688
#10 0x00007ffff7b02bb5 in iseq_mark (ptr=0x2398250) at ../iseq.c:106
#11 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=30252840, lev=3) at ../gc.c:1839
#12 0x00007ffff7a0c2c8 in mark_method_entry (key=, me=, data=)
at ../gc.c:1506
#13 mark_method_entry_i (key=, me=, data=) at ../gc.c:1530
#14 0x00007ffff7ab5d1c in st_foreach (table=0x23d4f70, func=0x7ffff7a0c270 , arg=140737488315968)
at ../st.c:787
#15 0x00007ffff7a0be59 in mark_m_tbl (objspace=0x6038a0, ptr=21024760, lev=2) at ../gc.c:1541
#16 gc_mark_children (objspace=0x6038a0, ptr=21024760, lev=2) at ../gc.c:1800
#17 0x00007ffff7a0b8d8 in gc_mark_children (objspace=0x6038a0, ptr=41607400, lev=1) at ../gc.c:1795
#18 0x00007ffff7a0df82 in mark_locations_array (start=, end=) at ../gc.c:1401
#19 gc_mark_locations (start=, end=) at ../gc.c:1414
#20 rb_gc_mark_locations (start=, end=) at ../gc.c:1420
#21 0x00007ffff7b0768e in env_mark (ptr=0x2bfda70) at ../vm.c:238
#22 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=41607040, lev=1) at ../gc.c:1839
#23 0x00007ffff7a0df82 in mark_locations_array (start=, end=) at ../gc.c:1401
#24 gc_mark_locations (start=, end=) at ../gc.c:1414
#25 rb_gc_mark_locations (start=, end=) at ../gc.c:1420
#26 0x00007ffff7b0768e in env_mark (ptr=0x2562950) at ../vm.c:238
#27 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=41607000, lev=1) at ../gc.c:1839
#28 0x00007ffff79fc9d5 in proc_mark (ptr=0x277bf80) at ../proc.c:53
#29 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=41606920, lev=2) at ../gc.c:1839
#30 0x00007ffff7a0c2fc in mark_method_entry (key=, me=, data=)
at ../gc.c:1509
#31 mark_method_entry_i (key=, me=, data=) at ../gc.c:1530
#32 0x00007ffff7ab5d1c in st_foreach (table=0x25b2070, func=0x7ffff7a0c270 , arg=140737488316608)
at ../st.c:787
#33 0x00007ffff7a0be59 in mark_m_tbl (objspace=0x6038a0, ptr=33366880, lev=1) at ../gc.c:1541
#34 gc_mark_children (objspace=0x6038a0, ptr=33366880, lev=1) at ../gc.c:1800
#35 0x00007ffff7b02ba5 in iseq_mark (ptr=0x27528f0) at ../iseq.c:107
#36 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=33147080, lev=2) at ../gc.c:1839
#37 0x00007ffff7a0be2b in gc_mark_children (objspace=0x6038a0, ptr=, lev=1) at ../gc.c:1815
#38 0x00007ffff7b02bf5 in iseq_mark (ptr=0x27520f0) at ../iseq.c:102
#39 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=33147560, lev=2) at ../gc.c:1839
#40 0x00007ffff7a0be2b in gc_mark_children (objspace=0x6038a0, ptr=, lev=1) at ../gc.c:1815
#41 0x00007ffff7b02bf5 in iseq_mark (ptr=0x2751db0) at ../iseq.c:102
#42 0x00007ffff7a0baba in gc_mark_children (objspace=0x6038a0, ptr=33148280, lev=5) at ../gc.c:1839
#43 0x00007ffff7a0c2c8 in mark_method_entry (key=, me=, data=)
at ../gc.c:1506
```

```
#44 mark_method_entry_i (key=, me=, data=) at ../gc.c:1530
#45 0x00007ffff7ab5d1c in st_foreach (table=0x27e52d0, func=0x7ffff7a0c270 , arg=140737488317232)
---Type  to continue, or q  to quit---quit
at ../st.Quit
(gdb) up 2
#2  gc_mark_children (objspace=0x6038a0, ptr=19563040, lev=5) at ../gc.c:1801
1801        mark_tbl(objspace, RCLASS_IV_TBL(obj), lev);
(gdb) p *obj
$1 = {as = {free = {flags = 4130, next = 0x665b20}, basic = {flags = 4130, klass = 6708000}, object = {basic = {flags = 4130,
klass = 6708000}, as = {heap = {numiv = 42751520, ivptr = 0x2af3720, iv_index_tbl = 0x0}, ary = {42751520, 45037344,
0}}}, klass = {basic = {flags = 4130, klass = 6708000}, ptr = 0x28c5620, m_tbl = 0x2af3720, iv_index_tbl = 0x0},
flonum = {basic = {flags = 4130, klass = 6708000}, float_value = 2.1122057339494968e-316}, string = {basic = {
flags = 4130, klass = 6708000}, as = {heap = {len = 42751520, ptr = 0x2af3720 "\340\263\333\367\377\177", aux = {
capa = 0, shared = 0}}, ary = " V\214\002\000\000\000\000 7\257\002", '\000' }}, array = {
basic = {flags = 4130, klass = 6708000}, as = {heap = {len = 42751520, aux = {capa = 45037344, shared = 45037344},
ptr = 0x0}, ary = {42751520, 45037344, 0}}}, regexp = {basic = {flags = 4130, klass = 6708000}, ptr = 0x28c5620,
src = 45037344, usecnt = 0}, hash = {basic = {flags = 4130, klass = 6708000}, ntbl = 0x28c5620, iter_lev = 45037344,
ifnone = 0}, data = {basic = {flags = 4130, klass = 6708000}, dmark = 0x28c5620, dfree = 0x2af3720, data = 0x0},
typeddata = {basic = {flags = 4130, klass = 6708000}, type = 0x28c5620, typed_flag = 45037344, data = 0x0}, rstruct = {
basic = {flags = 4130, klass = 6708000}, as = {heap = {len = 42751520, ptr = 0x2af3720}, ary = {42751520, 45037344,
0}}}, bignum = {basic = {flags = 4130, klass = 6708000}, as = {heap = {len = 42751520, digits = 0x2af3720}, ary = {
42751520, 0, 45037344, 0, 0, 0}}}, file = {basic = {flags = 4130, klass = 6708000}, fptr = 0x28c5620}, node = {
flags = 4130, nd_reserved = 6708000, u1 = {node = 0x28c5620, id = 42751520, value = 42751520, cfunc = 0x28c5620,
tbl = 0x28c5620}, u2 = {node = 0x2af3720, id = 45037344, argc = 45037344, value = 45037344}, u3 = {node = 0x0, id = 0,
state = 0, entry = 0x0, cnt = 0, value = 0}}, match = {basic = {flags = 4130, klass = 6708000}, str = 42751520,
rmatch = 0x2af3720, regexp = 0}, rational = {basic = {flags = 4130, klass = 6708000}, num = 42751520, den = 45037344},
complex = {basic = {flags = 4130, klass = 6708000}, real = 42751520, imag = 45037344}}}
(gdb) p 4130 & 0x1f
$2 = 2
(gdb) p obj.as.klass
$3 = {basic = {flags = 4130, klass = 6708000}, ptr = 0x28c5620, m_tbl = 0x2af3720, iv_index_tbl = 0x0}
(gdb) p obj.as.klass.m_tbl
$4 = (struct st_table *) 0x2af3720
(gdb) p *obj.as.klass.m_tbl
$5 = {type = 0x7ffff7dbb3e0, num_bins = 11, entries_packed = 1, num_entries = 0, bins = 0x28c1730, head = 0x0, tail = 0x0}
(gdb) p obj.as.klass.m_tbl.bins[0]
$6 = (struct st_table_entry *) 0x0
```

(gdb)

#5 - 07/23/2011 06:53 AM - normalperson (Eric Wong)

Hiroshi Nakamura nakahiro@gmail.com wrote:

> Can someone enlighten me to find the bug?

I started looking into this, too, but haven't found the exact
cause/solution, either.

It does not seem to be in the tinytds nor bcrypt C extensions,
I put in early returns for each extensions' Init_* functions
to avoid hitting C code outside of Ruby trunk.

It seems to be related to how the parser/loader sets up (or fails to set
up) objects for GC. Some backtraces I get are very deep inside gc_mark*
functions (500-700+ C stack frames).

I cannot reproduce this with GC.disable or a huge malloc limit
(RUBY_GC_MALLOC_LIMIT=800000000).

#6 - 07/23/2011 07:53 AM - drbrain (Eric Hodel)

- ruby -v changed from ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-linux] to -

On Jul 22, 2011, at 2:41 PM, Eric Wong wrote:

> Hiroshi Nakamura nakahiro@gmail.com wrote:
>
> > Can someone enlighten me to find the bug?
>
> I started looking into this, too, but haven't found the exact
> cause/solution, either.

> It does not seem to be in the tinytds nor bcrypt C extensions,
> I put in early returns for each extensions' Init_* functions
> to avoid hitting C code outside of Ruby trunk.
>
> It seems to be related to how the parser/loader sets up (or fails to set
> up) objects for GC. Some backtraces I get are very deep inside gc_mark*
> functions (500-700+ C stack frames).
>
> I cannot reproduce this with GC.disable or a huge malloc limit
> (RUBY_GC_MALLOC_LIMIT=800000000).

Does GC.stress help?

#7 - 07/23/2011 07:59 AM - normalperson (Eric Wong)

Eric Hodel drbrain@segment7.net wrote:

> Does GC.stress help?

I tried it for a while but lost patience while waiting. I suppose
I'll try it again while I poke at other things...

--
Eric Wong

#8 - 07/23/2011 08:53 AM - nahi (Hiroshi Nakamura)

On Sat, Jul 23, 2011 at 07:56, Eric Wong normalperson@yhbt.net wrote:

> > Does GC.stress help?
>
> I tried it for a while but lost patience while waiting. Â I suppose
> I'll try it again while I poke at other things...

Yeah, I waited 2 hours before I quit the way. :-(

Adding rb_gc() to several part at load.c but I still cannot narrow
down where the broken object created...

Regards,
// NaHi

#9 - 07/24/2011 09:30 AM - nahi (Hiroshi Nakamura)

Random note;

- SEGV on trunk, ruby_1_9_3 and 1.9.2-p280
- SEGV on x86_64-linux but NOT on x86_64-darwin10.8.0 according to nagachika (Tomoyuki Chikanaga)
- SEGV at GC mark or sweep, and the broken object looks like a singleton class object
- SEGV caused at autoload from autoload from autoload... or at finalizing VM at exit.
- slight change of code, different location of SEGV.
- lack of RB_GC_GUARD for a Node which is used in particular code?

#10 - 07/24/2011 10:53 AM - mame (Yusuke Endoh)

Hello,

Thank you for the reproducing process. I could also comfirmed it.
This is one of the most difficult bug that I have seen...

In short, I've not solved this issue, but I found valgrind with -O0
enlightens us the (maybe) important point:

#11 - 07/24/2011 12:23 PM - mame (Yusuke Endoh)

Follow-up.

I found activesupport uses UnboundMethod and #bind to check whether
a class instance is singleton or not.

```
102   def singleton_class?
```

```
103     # in case somebody is crazy enough to overwrite allocate
104     allocate = Class.instance_method(:allocate)
105     # object.class always points to a real (non-singleton) class
106     allocate.bind(self).call.class != self
107   rescue TypeError
108     # MRI/YARV/JRuby all disallow creating new instances of a
singleton class
109     true
110   end
```

Of course, this is not a fault of activesupport, but this seems
one factor to this SEGV.
If you need to work around this issue right now, the following patch
for activesupport might work.

```
$ diff -u active_support/core_ext/class/attribute.rb.orig
active_support/core_ext/class/attribute.rb
--- active_support/core_ext/class/attribute.rb.orig     2011-07-24
11:51:32.690364370 +0900
+++ active_support/core_ext/class/attribute.rb  2011-07-24
11:51:35.990380754 +0900
@@ -100,6 +100,7 @@

 private
 def singleton_class?
```

- return !ancestors.include?(self)   # in case somebody is crazy enough to overwrite allocate   allocate = Class.instance_method(:allocate)   # object.class always points to a real (non-singleton) class

At least, SEGV does not occur with this patch on my machine.
Could anyone please try it?

Again, this is not a fault of activesupport. This is just workaround
patch.

--
Yusuke Endoh mame@tsg.ne.jp

#12 - 07/24/2011 12:23 PM - mame (Yusuke Endoh)

2011/7/24 Yusuke ENDOH mame@tsg.ne.jp:

> If you need to work around this issue right now, the following patch
> for activesupport might work.

FYI, Aaron fixed activesupport in trunk so quickly :-)

https://github.com/rails/rails/commit/cdc4274931c2d6bafdf2b97f7e4ecedf89a8202e

--
Yusuke Endoh mame@tsg.ne.jp

#13 - 07/24/2011 09:09 PM - mame (Yusuke Endoh)

- File 5047.patch added

- Assignee changed from nahi (Hiroshi Nakamura) to ko1 (Koichi Sasada)

I probably managed to fix this issue.
Could anyone try the attached patch?
ko1: May I commit it?

This problem is caused by the combination of the wrong design
of Method class and lack of YARV method entry marking, with
obfuscated by conservative GC and Fiber's machine stack...

Each method has struct rb_method_entry_t. The struct is free'd
when its method is removed, except the method is "live" (i.e.,
method being now executed).
To prevent live rb_method_entry_t from being free'd, YARV
traverses the YARV stack and marks each live rb_method_entry_t
at GC marking phase.
However, when the method is BMETHOD (i.e., Method#call), the
system does not work correctly because rb_method_entry_t is

embedded in the internal structure of Method class (i.e.,
struct METHOD). The rb_method_entry_t is free'd when the Method
object is GC'd, even if the YARV attempts to prevent it.

My patch separates rb_method_entry_t from struct METHOD,

struct METHOD {
VALUE recv;
VALUE rclass;
ID id;

- rb_method_entry_t me;
- rb_method_entry_t *me; };

and marks them explicitly at each marking phase.

I tried to create a small test for this issue, but didn't
succeeded because of conservative GC.

The following is Japanese translation.

拙い英語で書いただけだと後でわからなくなる自信があるので日本語でも
簡単に。

要は、struct METHOD 中の rb_method_entry_t をポインタにするパッチ
です。
rb_method_entry_t は YARV スタックに載っている限りは free しちゃ
ダメなんですが、struct METHOD に埋めこまれた状態だと、Method オブ
ジェクトが解放された時にあわせて free されてしまって、でもスタック
には乗ってるので解放領域を mark とかしてメモリが破壊されているので
した。
rb_method_entry_t をポインタに切り分けて別途 malloc するようにし
たら直った気がします。

小さい例も作りたかったんですが、conservative GC がなかなか Method
を解放してくれず、作れませんでした。

rb_method_entry_t を作ったのはささださんなので、一応ささださんに
確認して欲しいです。

--
Yusuke Endoh mame@tsg.ne.jp

#14 - 07/25/2011 03:23 AM - normalperson (Eric Wong)

Yusuke Endoh mame@tsg.ne.jp wrote:

> I probably managed to fix this issue.
> Could anyone try the attached patch?

Seems to work for me.

Explanation is good and makes sense to me. Thanks!

sidenote: I didn't realize xmalloc() (via rb_unlink_method_entry) is
safe/allowed inside bm_free(); but apparently it's only rb_new_obj()
that's prevented inside GC.

--
Eric Wong

#15 - 07/25/2011 06:34 AM - nahi (Hiroshi Nakamura)

Yusuke Endoh wrote:

> I probably managed to fix this issue.
> Could anyone try the attached patch?
> ko1: May I commit it?

With the patch, both trunk and ruby_1_9_3 run without SEGV. My hat's off to you for your exhausting job.

#16 - 07/25/2011 07:29 AM - ko1 (Koichi Sasada)

(2011/07/25 6:35), Hiroshi Nakamura wrote:

> With the patch, both trunk and ruby_1_9_3 run without SEGV. My hat's off to you for your exhausting job.

Me too. Thanks a lot.

--
// SASADA Koichi at atdot dot net

#17 - 07/25/2011 08:38 AM - kosaki (Motohiro KOSAKI)

> > With the patch, both trunk and ruby_1_9_3 run without SEGV. My hat's off to you for your exhausting job.
>
> Me too. Thanks a lot.

Me too. You are great.

Offtopic: now 1.9.3 have two high priority (i.e. crash related) bug, #5039 and this. And this was
most mysterious and difficult one. Now I don't doubt 1.9.3 is going to be released on a planned schedule.
Of course, there still are several normal priority bugs. we'll do our best.

#18 - 07/25/2011 10:29 PM - mame (Yusuke Endoh)

Thank you for the trying the patch.
Nobu pointed out that the patch may cause memory leak.
I'll commit a revised version soon.

2011/7/25 Eric Wong normalperson@yhbt.net:

> sidenote: I didn't realize xmalloc() (via rb_unlink_method_entry) is
> safe/allowed inside bm_free(); but apparently it's only rb_new_obj()
> that's prevented inside GC.

Good point. I was not aware of this.
Indeed, GC does not occur recursively even if vm_xmalloc is called
during GC. But NoMemoryError may be raised from bm_free()... It
will probably make the interpreter state inconsistent, so I guess
it is not possible to sanely continue the execution by rescue'ing
the exception.
But I have no solution about the case. Frankly speaking, I think
we can do nothing if just few dozens byte allocation is failed.

2011/7/25 Motohiro KOSAKI kosaki.motohiro@gmail.com:

> > > With the patch, both trunk and ruby_1_9_3 run without SEGV. My hat's off to you for your exhausting job.
> >
> > Me too. Thanks a lot.
>
> Me too. You are great.

Thanks for the compliment!

--
Yusuke Endoh mame@tsg.ne.jp

#19 - 07/25/2011 11:23 PM - normalperson (Eric Wong)

Yusuke ENDOH mame@tsg.ne.jp wrote:

> 2011/7/25 Eric Wong normalperson@yhbt.net:
>
> > sidenote: I didn't realize xmalloc() (via rb_unlink_method_entry) is
> > safe/allowed inside bm_free(); but apparently it's only rb_new_obj()
> > that's prevented inside GC.

Good point. I was not aware of this.
Indeed, GC does not occur recursively even if vm_xmalloc is called during GC. But NoMemoryError may be raised from bm_free()... It will probably make the interpreter state inconsistent, so I guess it is not possible to sanely continue the execution by rescue'ing the exception.
But I have no solution about the case. Frankly speaking, I think we can do nothing if just few dozens byte allocation is failed.

We can pre-allocate the unlinked_method_entry_list_entry struct
in the METHOD struct and then push it into unlinked_method_entry_list
in bm_free (untested patch below):

--- a/proc.c
+++ b/proc.c
@@ -19,6 +19,7 @@ struct METHOD {
VALUE rclass;
ID id;
rb_method_entry_t *me;

- struct unlinked_method_entry_list_entry *ume; };

VALUE rb_cUnboundMethod;
@@ -868,7 +869,11 @@ static void
bm_free(void *ptr)
{
struct METHOD *data = ptr;

- rb_unlink_method_entry(data->me);
- struct unlinked_method_entry_list_entry *ume = data->ume; +
- ume->me = data->me;
- ume->next = GET_VM()->unlinked_method_entry_list;
- GET_VM()->unlinked_method_entry_list = ume; xfree(ptr); }

@@ -978,6 +983,7 @@ mnew(VALUE klass, VALUE obj, ID id, VALUE mclass, int scope)
data->me = me2;
*data->me = *me;
data->me->def->alias_count++;

- data->ume = ALLOC(struct unlinked_method_entry_list_entry);

OBJ_INFECT(method, klass);

@@ -1088,6 +1094,7 @@ method_unbind(VALUE obj)
*data->me = *orig->me;
if (orig->me->def) orig->me->def->alias_count++;
data->rclass = orig->rclass;

- data->ume = ALLOC(struct unlinked_method_entry_list_entry);
OBJ_INFECT(method, obj);

return method;

Eric Wong

#20 - 07/25/2011 11:29 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r32669.
Jack, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- proc.c (struct METHOD), gc.c (gc_marks), vm_method.c (rb_gc_mark_unlinked_live_method_entries): fix SEGV bug. rb_method_entry_t was free'd even when the method is still on the stack if it is BMETHOD (i.e., Method#call). This is because rb_method_entry_t is embedded in struct METHOD. This commit separates them and marks the live method entries. See [ruby-core:38449] in detail. fix [Bug #5047] [ruby-core:38171]

#21 - 07/25/2011 11:53 PM - mame (Yusuke Endoh)

2011/7/25 Eric Wong normalperson@yhbt.net:

> We can pre-allocate the unlinked_method_entry_list_entry struct
> in the METHOD struct and then push it into unlinked_method_entry_list
> in bm_free (untested patch below):

I see! I'll import it if make check passes. Thanks.

--
Yusuke Endoh mame@tsg.ne.jp

Files

segfault.zip	33.6 KB	07/19/2011	jackc (Jack Christensen)
5047.patch	8.27 KB	07/24/2011	mame (Yusuke Endoh)