Ruby master - Bug #13164

A second `SystemStackError` exception results in `Segmentation fault (core dumped)`

01/27/2017 01:41 PM - myst (Boaz Segev)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.0p0 (2016-12-24 revision 57164) [x86_64-darwin16]	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

This issue is was exposed by leveraging the fact that Object#hash is implemented recursively for core Ruby datatypes (i.e., Hash and Array). See the discussion here: https://github.com/boazsegev/combine_pdf/pull/91#issuecomment-275552131.

TO reproduce the issue, explode the stack **twice**.

Expected results:

SystemStackError will be raised both times.

Actual results:

SystemStackError is raised once. The second time will cause a core dump.

Code to cause core dump:

```
def compute_nest_depth
  h = {nest: {}}
  nest = h[:nest]
  i = 0

  while true
    i += 1
    puts "nested #{i}" if ((i & 511) == 0)
    next_nest = { nest: {} }
    nest[:nest] = next_nest
    nest = next_nest[:nest]
    h.hash
  end

rescue SystemStackError
  puts "Stack exploded at nesting #{i}"
end

counter = 0;
while(true)
  begin
    counter +=1
    puts "starting test #{counter}"
    compute_nest_depth
  rescue SystemStackError => e
    nil
  ensure
    puts "test #{counter} complete"
  end
end
```

results:

```
starting test 1
nested 512
nested 1024
```

```
nested 1536
nested 2048
nested 2560
Stack exploded at nesting 2783
test 1 complete
starting test 2
nested 512
nested 1024
nested 1536
nested 2048
nested 2560
Segmentation fault (core dumped)
```

Related issues:

Related to Ruby master - Bug #13412: Infinite recursion with define_method ma...	**Closed**
Related to Ruby master - Bug #13948: Segfault instead of recursion depth error	**Closed**
Has duplicate Ruby master - Bug #13596: Segfault when catching SystemStackErr...	**Closed**

Associated revisions

Revision 58352 - 04/14/2017 12:51 PM - nobu (Nobuyoshi Nakada)

configure.in: sigsetjmp sivesigs flag

- configure.in (RUBY_SETJMP_TYPE): optional flag to save signal mask.

Revision 1e1a5853 - 04/14/2017 12:59 PM - nobu (Nobuyoshi Nakada)

signal.c: unblock signal

- signal.c (raise_stack_overflow): unblock the received signal, to receive the same signal again. [ruby-core:79285] [Bug #13164]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58353 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 58353 - 04/14/2017 12:59 PM - nobu (Nobuyoshi Nakada)

signal.c: unblock signal

- signal.c (raise_stack_overflow): unblock the received signal, to receive the same signal again. [ruby-core:79285] [Bug #13164]

Revision 58353 - 04/14/2017 12:59 PM - nobu (Nobuyoshi Nakada)

signal.c: unblock signal

- signal.c (raise_stack_overflow): unblock the received signal, to receive the same signal again. [ruby-core:79285] [Bug #13164]

Revision 58353 - 04/14/2017 12:59 PM - nobu (Nobuyoshi Nakada)

signal.c: unblock signal

- signal.c (raise_stack_overflow): unblock the received signal, to receive the same signal again. [ruby-core:79285] [Bug #13164]

Revision 58361 - 04/15/2017 02:07 AM - nobu (Nobuyoshi Nakada)

signal.c: prefer pthread_sigmask

- signal.c (raise_stack_overflow): prefer pthread_sigmask to sigprocmask, for multithreading.

History

#1 - 01/28/2017 01:10 PM - nobu (Nobuyoshi Nakada)

By doubling rb_sigaltstack_size(), it doesn't segfault and the second or more stack overflows never happen now.
I suspect that the stack guard page may need to be reset, but not sure.

```
diff --git a/signal.c b/signal.c
index 888c8eaa72..8947b0ea95 100644
--- a/signal.c
+++ b/signal.c
@@ -563,7 +563,7 @@ rb_sigaltstack_size(void)
     }
 #endif

-    return size;
```

```
+     return size * 2;
 }

 /* alternate stack for SIGSEGV */
```

#2 - 01/28/2017 08:46 PM - myst (Boaz Segev)

This is a good observation and I'm happy you found this...

However, I'm not sure that using return size * 2 as a patch will solve the issue. It might end up masking the real issue, making it harder to find (although I might be wrong).

At the moment, there is a segmentation fault. Is it possible that the size returned is somehow effecting a memory address / pointer in a way that it shouldn't...?

#3 - 01/29/2017 01:07 PM - nobu (Nobuyoshi Nakada)

When configured with --with-setjmp-type=sigsetjmp, it seemed working.
But segfaulted at the fourth system stack overflow.

#4 - 02/01/2017 08:32 PM - myst (Boaz Segev)

What about flattening recursion in core types (Hash, Array and Set)?

I know this won't resolve the issue, but it will prevent eql? and hash from exploding the stack, so the issue is less likely to occur when there isn't an error in the code being executed.

#5 - 04/09/2017 03:54 PM - nobu (Nobuyoshi Nakada)

- Has duplicate Bug #13412: Infinite recursion with define_method may cause silent SEGV or cfp consistency error added

#6 - 04/11/2017 05:25 AM - nobu (Nobuyoshi Nakada)

- Has duplicate deleted (Bug #13412: Infinite recursion with define_method may cause silent SEGV or cfp consistency error)

#7 - 04/11/2017 05:25 AM - nobu (Nobuyoshi Nakada)

- Related to Bug #13412: Infinite recursion with define_method may cause silent SEGV or cfp consistency error added

#8 - 04/14/2017 01:00 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset trunk|r58353.

---

signal.c: unblock signal

- signal.c (raise_stack_overflow): unblock the received signal, to receive the same signal again. [ruby-core:79285] [Bug #13164]

#9 - 04/14/2017 01:05 PM - nobu (Nobuyoshi Nakada)

- Status changed from Closed to Open

On Linux, fixed by unblocking the received signal.
But it has no effect on mac OS and seems to need --with-setjmp-type=setjmp.

#10 - 05/24/2017 08:04 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Bug #13596: Segfault when catching SystemStackError in eval added

#11 - 09/29/2017 12:54 AM - shyouhei (Shyouhei Urabe)

- Related to Bug #13948: Segfault instead of recursion depth error added