Ruby master - Bug #10290

segfault when calling a lambda recursively after rescuing SystemStackError

09/24/2014 06:20 AM - jacknagel (Jack Nagel)

Status:	Closed		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:			
ruby -v:	ruby 2.1.3p242 (2014-09-19 revision 47629) [x86_64-darwin13.0]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

The following code segfaults on Ruby 2.1.3:

```
l = -> { l.() }

begin
  l.()
rescue SystemStackError
  l.() # segfault
end
```

the issue does not occur on trunk.

History

#1 - 09/24/2014 06:28 AM - jacknagel (Jack Nagel)

The segfault also occurs on the OS X 10.9 system ruby, 2.0.0-p481; not sure about the most recent patch release.

#2 - 09/25/2014 05:11 AM - jacknagel (Jack Nagel)

- File ruby_2014-09-25-000925_haswell.log added

I've attached the OS X crash report.

#3 - 09/25/2014 05:18 AM - jacknagel (Jack Nagel)

- File ruby_2014-09-25-001644_haswell.log added

As I noted, running the example code on trunk does not segfault (it raises SystemStackError a second time, as I would expect):

```
$ ruby -ve "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
ruby 2.2.0dev (2014-09-25 trunk 47651) [x86_64-darwin13]
-e:1:in `block in <main>': stack level too deep (SystemStackError)
    from -e:1:in `call'
    from -e:1:in `block in <main>'
    from -e:1:in `call'
    from -e:1:in `block in <main>'
    from -e:1:in `call'
    from -e:1:in `block in <main>'
    from -e:1:in `call'
    from -e:1:in `block in <main>'
     ... 9604 levels...
    from -e:1:in `call'
    from -e:1:in `block in <main>'
    from -e:1:in `call'
    from -e:1:in `<main>'
```

However, if I run the same code in IRB, it *does* segfault:

```
$ irb
irb(main):001:0> RUBY_DESCRIPTION
=> "ruby 2.2.0dev (2014-09-25 trunk 47651) [x86_64-darwin13]"
irb(main):002:0> l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end
Segmentation fault: 11
```

I've attached the crash report for this one as well.

#4 - 09/26/2014 05:45 AM - nobu (Nobuyoshi Nakada)

I can't reproduce it with irb, but it might access over the top of stack.
Possibly, we need more margin for the guard page.

#5 - 10/13/2014 07:37 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Open to Feedback

I can't reproduce following versions:

```
ruby 1.9.3p548 (2014-09-06) [x86_64-darwin13.4.0]
ruby 2.0.0p576 (2014-09-19 revision 47628) [x86_64-darwin13.4.0]
ruby 2.1.3p242 (2014-09-19 revision 47630) [x86_64-darwin13.0]
ruby 2.2.0preview1 (2014-09-17 trunk 47616) [x86_64-darwin13]
ruby 2.2.0dev (2014-10-13 trunk 47898) [x86_64-darwin13]
```

#6 - 10/13/2014 05:16 PM - jacknagel (Jack Nagel)

I can reproduce it on 2.0.0-p576 when compiled with -Os, but not -O2:

```
$ make clean && ./configure --disable-install-doc CC=clang CFLAGS=-O2 && make -j12
$ ./miniruby -v
ruby 2.0.0p576 (2014-09-19 revision 47627) [x86_64-darwin13.4.0]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
-e:1: stack level too deep (SystemStackError)

$ make clean && ./configure --disable-install-doc CC=clang CFLAGS=-Os && make -j12
$ ./miniruby -v
ruby 2.0.0p576 (2014-09-19 revision 47627) [x86_64-darwin13.4.0]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
Segmentation fault: 11
```

On 2.1.3, it does not happen when compiled without optimizations, but even using -O1 is enough to trigger it:

```
$ make clean && ./configure --disable-install-doc CC=clang && make -j12
$ ./miniruby -v
ruby 2.1.3p242 (2014-09-19 revision 47629) [x86_64-darwin13.0]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
-e:1: stack level too deep (SystemStackError)

$ make clean && ./configure --disable-install-doc CC=clang CFLAGS=-O1 && make -j12
$ ./miniruby -v
ruby 2.1.3p242 (2014-09-19 revision 47629) [x86_64-darwin13.0]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
Segmentation fault: 11
```

And similarly on trunk, it is triggered with -O1 or higher:

```
$ make clean && ./configure --disable-install-doc CC=clang && make -j12
$ ./miniruby -v
ruby 2.2.0dev (2014-10-14 trunk 47906) [x86_64-darwin13]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
-e:1:in `call': stack level too deep (SystemStackError)

make clean && ./configure --disable-install-doc CC=clang CFLAGS=-O1 && make -j12
$ ./miniruby -v
ruby 2.2.0dev (2014-10-14 trunk 47906) [x86_64-darwin13]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
Segmentation fault: 11
```

I'm using the latest Apple clang:

```
$ clang --version
Apple LLVM version 6.0 (clang-600.0.54) (based on LLVM 3.5svn)
Target: x86_64-apple-darwin13.4.0
Thread model: posix
```

#7 - 10/14/2014 01:49 AM - hsbt (Hiroshi SHIBATA)

I can't reproduce with gcc-4.9(not clang).

#8 - 10/14/2014 12:02 PM - hsbt (Hiroshi SHIBATA)

I can reproduce clang on linux.

```
[hsbt@chkbuild001 ~]$ clang -v
clang version 3.5.0 (tags/RELEASE_350/final)
Target: x86_64-amazon-linux-gnu
Thread model: posix
Found candidate GCC installation: /usr/bin/../lib/gcc/x86_64-amazon-linux/4.8.2
Found candidate GCC installation: /usr/lib/gcc/x86_64-amazon-linux/4.8.2
Selected GCC installation: /usr/bin/../lib/gcc/x86_64-amazon-linux/4.8.2
Candidate multilib: .;@m64
Candidate multilib: 32;@m32
Selected multilib: .;@m64
```

#9 - 10/16/2014 01:41 AM - jacknagel (Jack Nagel)

I can reproduce it with gcc 4.9.1 at -Os:

```
 $ gcc-4.9 --version
gcc-4.9 (Homebrew gcc 4.9.1) 4.9.1
Copyright (C) 2014 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ make clean && ./configure --disable-install-doc CC=gcc-4.9 CFLAGS=-Os && make -j12
$ ./miniruby -v
ruby 2.2.0dev (2014-10-16 trunk 47971) [x86_64-darwin13]
$ ./miniruby -e "l = -> { l.() }; begin; l.(); rescue SystemStackError; l.(); end"
Segmentation fault: 11
```

#10 - 10/16/2014 05:22 AM - nobu (Nobuyoshi Nakada)

- Description updated

I could reproduce it with gcc-4.9 on either Linux and OS X.
On Linux, SIGSEGV seems masked after the first stack overflow occurred, and it seems working by enabling interrupts.

```
diff --git c/eval.c i/eval.c
index 3e4ea16..6bd6ac9 100644
--- c/eval.c
+++ i/eval.c
@@ -500,7 +500,6 @@ setup_exception(rb_thread_t *th, int tag, volatile VALUE mesg, VALUE cause)
     if (cause == Qundef) {
     cause = nocause ? Qnil : get_thread_errinfo(th);
     }
-    exc_setup_cause(mesg, cause);

     file = rb_sourcefile();
     if (file) line = rb_sourceline();
@@ -526,6 +525,7 @@ setup_exception(rb_thread_t *th, int tag, volatile VALUE mesg, VALUE cause)
         set_backtrace(mesg, at);
     }
     }
+    exc_setup_cause(mesg, cause);

     if (!NIL_P(mesg)) {
     th->errinfo = mesg;
diff --git c/signal.c i/signal.c
index d3c7cb8..b939930 100644
--- c/signal.c
+++ i/signal.c
@@ -767,6 +767,7 @@ check_stack_overflow(const uintptr_t addr, const ucontext_t *ctx)
          * place. */
         th->tag = th->tag->prev;
     }
+    rb_enable_interrupt();
     ruby_thread_stack_overflow(th);
     }
 }
```

However, on OS X it makes the second stack overflow SIGILL, with no outputs.
I have no idea what happens there.

#11 - 09/19/2016 07:48 AM - backus (John Backus)

Any update on this issue? I ran into this bug yesterday and spent hours investigating the source of the segfault. Here is the code that caused the issue for me:

```
def foo
  define_singleton_method(:method_missing)           { |*| invalid_method }
  define_singleton_method(:define_singleton_method) { |*| invalid_method }
end

foo

begin
  do_not_segfault
rescue SystemStackError
end

do_not_segfault
```

#12 - 09/20/2016 12:48 AM - nobu (Nobuyoshi Nakada)

It causes SystemStackError as expected, with ruby 2.3.

#13 - 09/22/2016 07:24 AM - backus (John Backus)

Nobuyoshi Nakada wrote:

> It causes SystemStackError as expected, with ruby 2.3.

No it definitely segfaults for me:

```
$ cat ex.rb
l = -> { l.() }

begin
  l.()
rescue SystemStackError
  l.() # segfault
end
$ which ruby
/Users/johnbackus/.rubies/ruby-2.3.1/bin/ruby
$ ruby -v
ruby 2.3.1p112 (2016-04-26 revision 54768) [x86_64-darwin15]
$ ruby ex.rb
[1]    82276 segmentation fault  ruby ex.rb
```

anything I can run to help you reproduce for 2.3 on OS X?

#14 - 11/25/2016 01:28 PM - shyouhei (Shyouhei Urabe)

- Assignee set to nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Assigned

ping nobu.

#15 - 05/11/2017 05:47 AM - backus (John Backus)

ping nobu

#16 - 05/12/2017 06:27 AM - nobu (Nobuyoshi Nakada)

On macOS, --with-setjmp-type=setjmp configuration option may fix it.

#17 - 05/12/2017 07:54 AM - backus (John Backus)

nobu (Nobuyoshi Nakada) wrote:

> On macOS, --with-setjmp-type=setjmp configuration option may fix it.

This does fix the issue on macOS for me, thank you. Maybe this should be the macOS default?

#18 - 06/16/2017 09:47 AM - nobu (Nobuyoshi Nakada)

SystemStackError seems uncatchable since r58492.

#19 - 08/20/2019 08:20 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Assigned to Closed

I was able to reproduce the segfault in ruby 2.3 and 2.4, but not in 2.5, 2.6, 2.7.0-preview1, or the master branch, so I think this problem is fixed. If this problem still occurs for you, please reply with your environment details.

Files

ruby_2014-09-25-000925_haswell.log	46.4 KB	09/25/2014	jacknagel (Jack Nagel)
ruby_2014-09-25-001644_haswell.log	46.8 KB	09/25/2014	jacknagel (Jack Nagel)