Ruby master - Bug #16814

Segmentation fault in GC while running test/ruby/test_fiber.rb on s390x

04/24/2020 05:31 AM - ReiOdaira (Rei Odaira)

Status: Closed
Priority: Normal
Assignee: ioquatix (Samuel Williams)
Target version:
ruby -v: ruby 2.8.0dev (2020-04-12T03:45:22Z master 5c27681813) [s390x-linux]

Backport:
2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONE

Description
A segmentation fault almost always happens in test/ruby/test_fiber.rb with certain commits of latest Ruby on s390x.

$ make test-all TESTS=test/ruby/test_fiber.rb
Run options:
--seed=90044
"--ruby=./miniruby -I./lib -I. -l.ext/common ./tool/runruby.rb --extout=.ext -- --disable-gems"
--excludes-dir=./test/excludes
--name=!/memory_leak/

# Running tests:
[24/29] TestFiber#test_stack_size = 0.89 s
1) Failure:
TestFiber#test_stack_size [/home/chkbuild/my-tmp/build/20200412T043305Z/ruby/test/ruby/test_fiber.rb:356]:
pid 5713 killed by SIGABRT (signal 6) (core dumped)
| -e:1:in `print': stack level too deep (SystemStackError)
| from -e:1:in `rec'
| from -e:1:in `block (3 levels) in rec'
| from -e:1:in `times'
| from -e:1:in `block (2 levels) in rec'
| from -e:1:in `times'
| from -e:1:in `block in rec'
| from -e:1:in `times'
| from -e:1:in `rec'
| ... 172 levels...
| from -e:1:in `block in rec'

&: [BUG] Segmentation fault at 0x0000000000000000
ruby 2.8.0dev (2020-04-12T03:45:22Z master 5c27681813) [s390x-linux]

-- Control frame information -----------------------------------------------
c:0001 p:0000 s:0003 E:001e20 (none) [FINISH]

-- Other runtime information -----------------------------------------------

* Loaded script: -e

* Loaded features:
  0 enumerator.so
  1 thread.rb
  2 rational.so
  3 complex.so
  4 ruby2_keywords.rb
  5 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/enc/encdb.so
  6 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/enc/trans/transdb.so
  7 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/lib/rbconfig.rb
  8 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/lib/rubygems/compatibility.rb
  9 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/lib/rubygems/defaults.rb

09/02/2022 1/7
Process memory map:

```
2aa15380000-2aa15772000 r-xp 00000000 5e:01 1198050 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/ruby
2aa15772000-2aa15777000 r--p 003f1000 5e:01 1198050 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/ruby
2aa15777000-2aa15779000 rw-p 00000000 00:00 0
```

```
3ffaa180000-3ffaa182000 r-xp 00000000 5e:01 1197862 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/monitor.so
3ffaa182000-3ffaa183000 r--p 00001000 5e:01 1197862 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/monitor.so
3ffaa183000-3ffaa184000 rw-p 00002000 5e:01 1197862 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/monitor.so
3ffaa200000-3ffaa202000 r-xp 00000000 5e:01 1198135 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/enc/trans/transdb.so
3ffaa202000-3ffaa203000 r--p 00002000 5e:01 1198135 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/enc/trans/transdb.so
3ffaa203000-3ffaa204000 rw-p 00000000 5e:01 1198135 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/enc/trans/transdb.so
3ffaa204000-3ffaa205000 rw-p 00000000 5e:01 1198135 /home/chkbuild/my-tmp/build/20200412T043305Z/ruby/.ext/s390x-linux/enc/trans/transdb.so
```
<table>
<thead>
<tr>
<th>Process</th>
<th>File Path</th>
<th>Offset</th>
<th>Size</th>
<th>Permissions</th>
<th>Address</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3ffaab76000-3ffacc80000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffacc80000-3ffacc82000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffacc82000-3ffacc83000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffacc83000-3ffacc84000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffaccf0000-3ffaceb3000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>001ae000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3fffaceb3000-3fffaceb5000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>001b2000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3fffaceb5000-3fffaceb9000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffacfa0000-3ffacfa8000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00001000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffacfa8000-3ffacfa9000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00a70000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffacfa9000-3ffacfaa000</td>
<td>/usr/lib64/libfreebl3.so</td>
<td>00a80000</td>
<td>5e:01</td>
<td>133388</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad000000-3ffad00d000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad00d000-3ffad00e000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad00e000-3ffad00f000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad080000-3ffad083000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad083000-3ffad084000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad084000-3ffad085000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad100000-3ffad187000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad187000-3ffad189000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad189000-3ffad18a000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad200000-3ffad208000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad208000-3ffad209000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad209000-3ffad20a000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad280000-3ffad298000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad298000-3ffad299000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad299000-3ffad29a000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad29a000-3ffad29e000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad300000-3ffad317000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad317000-3ffad318000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad318000-3ffad319000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad380000-3ffad382000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
<tr>
<td>3ffad382000-3ffad383000</td>
<td>/usr/lib64/libcrypt-2.1.so</td>
<td>00000000</td>
<td>5e:01</td>
<td>135760</td>
<td>00000000</td>
<td>00:00</td>
</tr>
</tbody>
</table>
The segmentation fault happens in a Ruby script invoked from test_fiber.rb by EnvUtil.invoke_ruby(). The Ruby script is to deliberately cause a stack overflow as follows.

```ruby
$stdout.sync=true; def rec; print "."; 1.times{1.times{1.times{rec}}}; end; Fiber.new(rec).resume
```

On s390x, this script caused SystemStackError, which I think is expected. However, it was during the handling of the stack overflow when the segmentation fault happened.

The core dump shows the following stack trace.

```plaintext
(gdb) bt
#0 0x0000003ff862c1350 in raise () from /lib64/libc.so.6
#1 0x0000003ff862c2bd8 in abort () from /lib64/libc.so.6
#2 0x000002a0c84bf9a in die () at error.c:646
#3 0x000002a0c84bf9a in rb_bug_for_fatal_signal (default_sighandler=0x0, sig=sig@entry=11,
    cxt=ctx@entry=0x2a2a2d18fc50,
    fmt=fmt@entry=0x2a2a0c89352c "Segmentation fault at %p") at error.c:678
#4 0x000002a0c86bf6d0 in sigsegv (sig=<optimized out>, info=0x2a2a2d18fc50, cxt=0x2a2a2d18fc50) at signal.c:955
#5 <signal handler called>
#6 0x000002a0c85df36 in gc_mark_children (objspace=objspace@entry=0x2a2a2d09c6b0, obj=obj@entry=2929923415200)
at gc.c:5478
#7 0x000002a0c84d3ac8 in rgengc_rememberset_mark (heap=0x2a2a2d09c6d8, objspace=0x2a2a2d09c6b0) at gc.c:6747
#8 gc_marks_start (full_mark=<optimized out>, objspace=0x2a2a2d09c6b0) at gc.c:6314
#9 gc_marks (full_mark=<optimized out>, objspace=0x2a2a2d09c6b0) at gc.c:6583
#10 gc_start (objspace=objspace@entry=0x2a2a2d09c6b0, reason=<optimized out>,
    reason@entry=256) at gc.c:7370
#11 0x000002a0c859370 in heap_prepare (heap=0x2a2a2d09c6d8, 
    objspace=<optimized out>) at gc.c:1977
#12 heap_get_freeobj_from_next_freepage (objspace=objspace@entry=0x2a2a2d09c6b0, heap=<optimized out>) at gc.c:1989
---Type <return> to continue, or q <return> to quit---
#13 0x000002a0c854d7b0 in heap_get_freeobj (heap=0x2a2a2d09c6d8, 
    objspace=0x2a2a2d09c6b0) at gc.c:2028
#14 newobj_slowpath (wb_protected=1, objspace=0x2a2a2d09c6b0, v3=v3@entry=0, 
    v2=0, v1=0, flags=5, klass=2929923683840) at gc.c:2170
#15 newobj_slowpath_wb_protected (klass=2929923683840, flags=5, v1=v1@entry=0, 
    v2=v2@entry=0, v3=v3@entry=0, objspace=0x2a2a2d09c6b0) at gc.c:2182
#16 0x000002a0c854d817c in newobj_of (wb_protected=1, v3=0, v2=0, v1=0, 
    flags=5, klass=<optimized out>) at gc.c:2218
#17 rb_wb_protected_newobj_of (klass=<optimized out>, flags=flags@entry=5) at gc.c:2234
#18 0x0000002a0c7102b8 in str_alloc (klass=<optimized out>) at string.c:745
#19 str_new0 (klass=<optimized out>, ptr=0x2a2a0c85ad0e "$t", len=1, 
    termlen=<optimized out>) at string.c:767
#20 0x0000002a0c7b1f28 in ruby3_str_new_cstr (str=0x2a2a0c85ad0e "$t") at /include/ruby/3/intern/string.h:159
#21 print_backtrace (eclass=2929923530680, errat=errat@entry=2929923354920,
    str=str@entry=8, reverse=reverse@entry=0) at eval_error.c:250
#22 0x0000002a0c75b3f68 in print_backtrace (reverse=0, str=8, 
    errat=errat@entry=2929923530680, eclass=<optimized out>) at eval_error.c:233
#23 rb_error_write (reverse=0, highlight=0, str=8, errat=<optimized out>,
    emesg=2929923530600, errinfo=<optimized out>) at eval_error.c:340
```
At gc.c:5478, the segmentation fault happened because any->as.typeddata.type was 0. as.typeddata.type should not be 0 for RTypedData.

```c
    case T_DATA:
        {
            void *const ptr = DATA_PTR(obj);
            if (ptr) {
                RUBY_DATA_FUNC mark_func = RTYPEDDATA_P(obj) ?
                    any->as.typeddata.type->function.dmark :
                    any->as.data.dmark;
                if (mark_func) (*mark_func)(ptr);
            }
            break;
        }
```

This is a timing bug, but it almost always happens with 5c27681813. It is not clear to which commit this issue is related. In Ruby CI, it started happening in early February 2020 and stopped showing up after increasing the stack size by ulimit -s. It started happening again in early April 2020 and disappeared on April 15.

Anybody has any ideas how I should debug this?

Associated revisions
Revision 755a349a - 03/20/2021 06:54 AM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 4bff8e84232594ecb9914e2a8437b7c40a63b799: [Backport #16814]

Ensure that the head of the vacancy list is correctly inserted into the linked list.

See <https://bugs.ruby-lang.org/issues/16814> for more details.

History
#1 - 04/24/2020 06:07 AM - nobu (Nobuyoshi Nakada)
Where does any->as.data.free point?
Is any->as.basic.klass a valid class object?
If you compile gc.c as make DEFS=-DGC_DEBUG gc.o, any->file and any->line have the location in ruby level, and could help you.

#2 - 04/24/2020 06:38 AM - mame (Yusuke Endoh)

disappeared on April 15.

You may know, but the test has been skipped on s390x since 9948addda674b7a6e3575f1eba9025f998811d2.

#3 - 04/24/2020 11:05 PM - ReiOdaira (Rei Odaira)
Did you mean any->as.data.dfree? It points to no valid location.

(gdb) print any->as.data
$4 = {flags = 12, klass = 2930849422520, dmark = 0x0, dfree = 0x1,
     data = 0x2aa6449f9e0}
(gdb) print any->as.typeddata
$5 = {basic = {flags = 12, klass = 2930849422520}, type = 0x0, typed_flag = 1,
     data = 0x2aa6449f9e0)
any->as.basic.klass seems to be a valid class. Is there any way to figure out what class it is using the core dump file?

```plaintext
(gdb) print *(struct RBasic *)any->as.basic.klass
$7 = {flags = 98, klass = 2930849422480}
(gdb) print ((struct RBasic *)any->as.basic.klass)->flags & 0x1f
$9 = 2
```

I've tried make DEFS=-DGC_DEBUG gc.o. It made the test fail quite less often than before, and when it failed, it did at a different location in GC (gc.c:5240), but it will help a lot. Thanks.

Thanks, Endoh-san, I didn't know the test was skipped.

#4 - 04/26/2020 05:13 PM - mame (Yusuke Endoh)
FYI: I re-enabled the test in question with 93ed465dcd0866013d93c3662937497900c8086

#5 - 05/14/2020 10:50 AM - ioquatix (Samuel Williams)
@mame (Yusuke Endoh) I have merged the light weight concurrency patch, and it included some changes to these tests to make them less flaky, by putting it in separate test file. In my experience it seems much more reliable now. Just FYI.

#6 - 05/25/2020 02:08 AM - ioquatix (Samuel Williams)
Can you check if this is still a problem, I merged my changes which should make this test more reliable. But I did not fix any underlying problems.

#7 - 05/31/2020 07:42 AM - ReiOdaira (Rei Odaira)
On s390x, FIBER_POOL_ALLOCATION_FREE is enabled. The doubly linked list of fiber_pool->vacancies assumes that the head fiber_pool_vacancy has NULL in its previous field. However, when a fiber is released, fiber_pool_vacancy_push() called from fiber_pool_stack_release() does not store NULL to vacancy->previous.

Why this caused the observed symptom:
As test_stack_size uses up the VM stack of the fiber, it writes something into the memory location where struct fiber_pool_vacancy would reside if the stack were free. When the fiber is released, the stack's fiber_pool_vacancy is returned to the head of the vacancies doubly linked list, and then fiber_pool_allocation_free() is triggered. fiber_pool_vacancy_remove() manipulates the doubly linked list, and the vacancy->previous of the released fiber should have been NULL because it is at the head of the list.

```plaintext
if (vacancy->previous) {
    vacancy->previous->next = vacancy->next;
}
```

However, since vacancy->previous contains arbitrary data, the code snippet above destroys the memory location that happens to be pointed to by vacancy->previous. In test_stack_size, vacancy->previous happens to point to an encoding object that is live, and vacancy->next happens to be 0. This means vacancy->previous->next = vacancy->next; writes 0 into the as.typeddata.type field of the live object. This finally leads to the segmentation fault during GC.

#8 - 05/31/2020 08:21 AM - ioquatix (Samuel Williams)
@ReiOdaira (Rei Odaira) thanks for your careful analysis. It is very useful! I will review the code and get back to you.

#9 - 06/04/2020 10:05 AM - ioquatix (Samuel Williams)
inline static struct fiber_pool_vacancy *
fiber_pool_vacancy_push(struct fiber_pool_vacancy * vacancy, struct fiber_pool_vacancy * head)
{
    vacancy->next = head;

    #ifdef FIBER_POOL_ALLOCATION_FREE
    if (head) {
        head->previous = vacancy;
        vacancy->previous = NULL; // added
    }
    #endif

    return vacancy;
}

@ReiOdaira (Rei Odaira) do you think that's sufficient?

#10 - 06/04/2020 10:38 AM - ioquatix (Samuel Williams)
https://github.com/ruby/ruby/pull/3182

#11 - 06/04/2020 11:11 AM - ioquatix (Samuel Williams)
By the way, I've also removed all skips when I rewrote tests into test/ruby/test_stack.rb.

#12 - 06/04/2020 11:42 PM - ioquatix (Samuel Williams)
- Status changed from Open to Assigned
- Assignee set to ioquatix (Samuel Williams)

I have merged this.

@ReiOdaira (Rei Odaira) thanks for your effort, you deserve all the credit for tracking down this issue.

Can you please confirm whether the original issue is fixed? If so, we can close this issue.

Thanks!

#13 - 06/06/2020 02:23 AM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED

#14 - 02/26/2021 08:20 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Assigned to Closed

#15 - 03/20/2021 06:59 AM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.5: DONTNEED, 2.6: DONTNEED, 2.7: REQUIRED to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONE

ruby_2_7 755a349a3a66f5731995296fe3bb7d2b1712167f merged revision(s) 4bff8e84232594ecb9914e2a8437b7c40a63b799.