Ruby master - Bug #7951

Cannot rescue SignalException in 1.9.3 or 2.0.0

02/25/2013 10:55 AM - jamesotron (James Harton)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>2.0.0</td>
</tr>
<tr>
<td>ruby -v:</td>
<td>ruby 2.0.0p0 (2013-02-24 revision 39474) [x86_64-darwin12.2.1]</td>
</tr>
<tr>
<td>Backport:</td>
<td></td>
</tr>
</tbody>
</table>

Description
When running the example provided on [http://www.ruby-doc.org/core-1.9.3/SignalException.html](http://www.ruby-doc.org/core-1.9.3/SignalException.html) we find that neither Ruby 1.9.3p392 nor Ruby 2.0.0p0 are able to rescue SignalException:

```
$ cat <<EOF >signal_exception_test.rb
puts "#{RUBY_VERSION}p#{RUBY_PATCHLEVEL}"
begin
  Process.kill('HUP',Process.pid)
  rescue SignalException => e
    puts "received Exception #{e}"  # (e)
end
EOF

$ rvm 1.8.7,1.9.3,2.0.0 do ruby signal_exception_test.rb
1.8.7p371
received Exception SIGHUP
1.9.3p392
/Users/jnh/.rvm/scripts/set: line 20: 99759 Hangup: 1
  "${args[@]}"
2.0.0p0
/Users/jnh/.rvm/scripts/set: line 20: 99816 Hangup: 1
  "${args[@]}"
```

Associated revisions
Revision ffbd120e - 02/25/2013 06:43 PM - kosaki (Motohiro KOSAKI)
- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]
git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@39501 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 39501 - 02/25/2013 06:43 PM - kosaki (Motohiro KOSAKI)
- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]

Revision 39501 - 02/25/2013 06:43 PM - kosaki (Motohiro KOSAKI)
- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]

Revision 39501 - 02/25/2013 06:43 PM - kosaki (Motohiro KOSAKI)
- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]

Revision 39501 - 02/25/2013 06:43 PM - kosaki (Motohiro KOSAKI)
- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]
error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]

Revision 39501 - 02/25/2013 06:43 PM - kosaki (Motohiro KOSAKI)

- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]

Revision 2994c621 - 03/10/2013 04:11 PM - nagachika (Tomoyuki Chikanaga)
merge revision(s) 39501,39502:

* error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951] [ruby-core:52864]
* error.c: clarify reason for sleep in SignalException example

Revision 4ea96ece - 03/19/2013 04:40 AM - kosaki (Motohiro KOSAKI)

- internal.h: added a declaration of ruby_kill().
- thread.c (ruby_kill): helper function of kill().
- signal.c (rb_f_kill): use ruby_kill() instead of kill().
- signal.c (rb_f_kill): call rb_thread_execute_interrupts() to ensure that make SignalException if sent a signal to myself. [Bug #7951] [ruby-core:52864]
- vm_core.h (typedef struct rb_thread_struct): added th->interrupt_cond.
- thread.c (rb_threadptr_interrupt_common): added to initialization of th->interrupt_cond.
- thread.c (thread_create_core): ditto.
- test/ruby/test_signal.rb (TestSignal#test_hup_me): test for the above.

Revision 39819 - 03/19/2013 04:40 AM - kosaki (Motohiro KOSAKI)

- internal.h: added a declaration of ruby_kill().
- thread.c (ruby_kill): helper function of kill().
- signal.c (rb_f_kill): use ruby_kill() instead of kill().
- signal.c (rb_f_kill): call rb_thread_execute_interrupts() to ensure that make SignalException if sent a signal to myself. [Bug #7951] [ruby-core:52864]
- vm_core.h (typedef struct rb_thread_struct): added th->interrupt_cond.
- thread.c (rb_threadptr_interrupt_common): added to initialization of th->interrupt_cond.
- thread.c (thread_create_core): ditto.
- test/ruby/test_signal.rb (TestSignal#test_hup_me): test for the above.
internal.h: added a declaration of ruby_kill().

thread.c (ruby_kill): helper function of kill().

signal.c (rb_f_kill): use ruby_kill() instead of kill().

signal.c (rb_f_kill): call rb_thread_execute_interrups() to ensure that make SignalException if sent a signal to myself. [Bug #7951] [ruby-core:52864]

vm_core.h (typedef struct rb_thread_struct): added th->interrupt_cond.

thread.c (rb_threadptr_interrupt_common): added to initialization of th->interrupt_cond.

thread.c (thread_create_core): ditto.

test/ruby/test_signal.rb (TestSignal#test_hup_me): test for the above.

---

internal.h: added a declaration of ruby_kill().

thread.c (ruby_kill): helper function of kill().

signal.c (rb_f_kill): use ruby_kill() instead of kill().

signal.c (rb_f_kill): call rb_thread_execute_interrups() to ensure that make SignalException if sent a signal to myself. [Bug #7951] [ruby-core:52864]

vm_core.h (typedef struct rb_thread_struct): added th->interrupt_cond.

thread.c (rb_threadptr_interrupt_common): added to initialization of th->interrupt_cond.

thread.c (thread_create_core): ditto.

test/ruby/test_signal.rb (TestSignal#test_hup_me): test for the above.

---

internal.h: added a declaration of ruby_kill().

thread.c (ruby_kill): helper function of kill().

signal.c (rb_f_kill): use ruby_kill() instead of kill().

signal.c (rb_f_kill): call rb_thread_execute_interrups() to ensure that make SignalException if sent a signal to myself. [Bug #7951] [ruby-core:52864]

vm_core.h (typedef struct rb_thread_struct): added th->interrupt_cond.

thread.c (rb_threadptr_interrupt_common): added to initialization of th->interrupt_cond.
Revision 39819 - 03/19/2013 04:40 AM - kosaki (Motohiro KOSAKI)

- internal.h: added a declaration of ruby_kill().
- thread.c (ruby_kill): helper function of kill().
- signal.c (rb_f_kill): use ruby_kill() instead of kill().
- signal.c (rb_f_kill): call rb_thread_execute_interrupts() to ensure that make SignalException if sent a signal to myself. [Bug #7951] [ruby-core:52864]
- vm_core.h (typedef struct rb_thread_struct): added th->interrupt_cond.
- thread.c (rb_threadptr_interrupt_common): added to initialization of th->interrupt_cond.
- thread.c (thread_create_core): ditto.
- test/ruby/test_signal.rb (TestSignal#test_hup_me): test for the above.

Revision 6c3f1461 - 06/12/2017 04:52 AM - ko1 (Koichi Sasada)

- thread.c (ruby_kill): removed. This function is used with SIGSEGV, SIGBUS, SIGKILL, SIGILL, SIGFPE and SIGSTOP and these signals are affect immediately. So that `kill(2)` is enough for them.
- signal.c (rb_f_kill): ditto.
- vm_core.h (rb_thread_t::interrupt_cond): removed because only `ruby_kill()` uses this field.
- test/ruby/test_signal.rb: Without this patch sending SIGSTOP to own process wait another interrupt even if another process sends SIGCONT.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59066 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 59066 - 06/12/2017 04:52 AM - ko1 (Koichi Sasada)
remove ruby_kill() introduced for [Bug #7951].

- thread.c (rbuy_kill): removed. This function is used with SIGSEGV, SIGBUS, SIGKILL, SIGILL, SIGFPE and SIGSTOP and these signals are affect immediately. So that `kill(2)` is enough for them.
- signal.c (rb_f_kill): ditto.
- vm_core.h (rb_thread_t::interrupt_cond): removed because only `ruby_kill()` uses this field.
- test/ruby/test_signal.rb: Without this patch sending SIGSTOP to own process wait another interrupt even if another process sends SIGCONT.

Revision 59066 - 06/12/2017 04:52 AM - ko1 (Koichi Sasada)
remove ruby_kill() introduced for [Bug #7951].

- thread.c (rbuy_kill): removed. This function is used with SIGSEGV, SIGBUS, SIGKILL, SIGILL, SIGFPE and SIGSTOP and these signals are affect immediately. So that `kill(2)` is enough for them.
- signal.c (rb_f_kill): ditto.
- vm_core.h (rb_thread_t::interrupt_cond): removed because only `ruby_kill()` uses this field.
- test/ruby/test_signal.rb: Without this patch sending SIGSTOP to own process wait another interrupt even if another process sends SIGCONT.

Revision 59066 - 06/12/2017 04:52 AM - ko1 (Koichi Sasada)
remove ruby_kill() introduced for [Bug #7951].

- thread.c (rbuy_kill): removed. This function is used with SIGSEGV, SIGBUS, SIGKILL, SIGILL, SIGFPE and SIGSTOP and these signals are affect immediately. So that `kill(2)` is enough for them.
- signal.c (rb_f_kill): ditto.
- vm_core.h (rb_thread_t::interrupt_cond): removed because only `ruby_kill()` uses this field.
- test/ruby/test_signal.rb: Without this patch sending SIGSTOP to own process wait another interrupt even if another process sends SIGCONT.

History

#1 - 02/26/2013 03:38 AM - kosaki (Motohiro KOSAKI)
Following modified script works fine. That is a issue of YARV signal handling.

puts "#(RUBY_VERSION)p#(RUBY_PATCHLEVEL)"
begin
  Process.kill('HUP',Process.pid)
sleep 0.1
rescue SignalException => e
  puts "received Exception #{e}" 
end

08/07/2021
This issue was solved with changeset r39501.
James, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- error.c: clarify a document of SignalException. Process.kill() doesn't have any guarantee when signal will be delivered. [Bug #7951]
[ruby-core:52864]

#3 - 02/26/2013 03:45 AM - kosaki (Motohiro KOSAKI)
I changed a doc because the example has a mistake of usage of Process.kill(). Process.kill() doesn't have any guarantee when signal will be delivered.

#4 - 06/08/2017 07:41 AM - ko1 (Koichi Sasada)
- Description updated

#5 - 06/29/2017 05:07 PM - usa (Usaku NAKAMURA)
- Backport deleted (2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN)