Ruby - Bug #6009

Rapid signal delivery via kill(2) causes SystemStackError

02/13/2012 04:51 PM - drbrain (Eric Hodel)

Status:	Closed		
Priority:	Normal		
Assignee:	kosaki (Motohiro KOSAKI)		
Target version:			
ruby -v:	ruby 2.0.0dev (2012-02-11 trunk 34547) [x86_64-darwin11.2.0]	**Backport:**	

Description

=begin
Running the following program with a trivial signal handler can crash with a SystemStackError if signals are delivered rapidly:

$ ruby -ve 'p Process.pid; trap "USR1" do 10 ** 100; end; sleep'
ruby 2.0.0dev (2012-02-11 trunk 34547) [x86_64-darwin11.2.0]
4504
-e: SystemStackError

In a separate terminal:

ruby -e 'loop do Process.kill "USR1", 4504 end'

As each signal is delivered, ruby interrupts the current signal handler to perform the newly arrived one and quickly runs out of stack.

With an empty system handler the SystemStackError takes slightly longer to occur.

This also occurs with 1.9.3-p0:

$ ~/.multiruby/install/1.9.3-p0/bin/ruby -ve 'p Process.pid; trap "USR1" do 10 ** 100; end; sleep'
ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-darwin11.2.0]
4529
-e: SystemStackError

and with ruby-1.9.2-p290:

$ ~/.multiruby/install/1.9.2-p290/bin/ruby -ve 'p Process.pid; trap "USR1" do 10 ** 100; end; sleep'
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin11.2.0]
4534

ruby-1.8.7-p330 exits with a zero exit code:

$ ~/.multiruby/install/1.8.7-p330/bin/ruby -ve 'p Process.pid; trap "USR1" do 10 ** 100; end; sleep'
ruby 1.8.7 (2010-12-23 patchlevel 330) [i686-darwin11.2.0]
4564
$ echo $?
0

=end

Related issues:

Has duplicate Ruby - Bug #4909: trap□□□□□□□□□□□□□□□□□□□□□□□□	**Closed**	**06/20/2011**

Associated revisions

Revision 6190bb4d8ad7a07ddb1da8fc687b20612743a34a - 11/26/2012 10:57 AM - kosaki (Motohiro KOSAKI)

- ruby_atomic.h (ATOMIC_CAS): new macro for compare-and-exchange.
- vm_core.h (struct rb_thread_struct): add interrupt_mask member.

- thread.c (thread_create_core, Init_Thread): initialize
  th->thread_mask.
- vm_core.h (RUBY_VM_INTERRUPTED_ANY): new macro for avoiding
  bare th->interrupt_flag.
- vm_core.h (RUBY_VM_INTERRUPTED, RUBY_VM_INTERRUPTED): check
  th->interrupt_mask.
- thread.c (set_unblock_function, rb_thread_schedule): replace
  th->interrupt_flag with RUBY_VM_INTERRUPTED_ANY()
- signal.c (signal_exec): set up thread->interrupt_mask for
  preventing recursive trap handler.
- vm_core.h (RUBY_VM_CHECK_INTS, RUBY_VM_CHECK_INTS_BLOCKING): ditto.
- thread.c (rb_threadptr_execute_interrupts):
  don't process interrupt if it is masked.
  [Bug #6009] [ruby-core:42524]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37861 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 6190bb4d - 11/26/2012 10:57 AM - kosaki (Motohiro KOSAKI)

- ruby_atomic.h (ATOMIC_CAS): new macro for compare-and-exchange.
- vm_core.h (struct rb_thread_struct): add interrupt_mask member.
- thread.c (thread_create_core, Init_Thread): initialize
  th->thread_mask.
- vm_core.h (RUBY_VM_INTERRUPTED_ANY): new macro for avoiding
  bare th->interrupt_flag.
- vm_core.h (RUBY_VM_INTERRUPTED, RUBY_VM_INTERRUPTED): check
  th->interrupt_mask.
- thread.c (set_unblock_function, rb_thread_schedule): replace
  th->interrupt_flag with RUBY_VM_INTERRUPTED_ANY()
- signal.c (signal_exec): set up thread->interrupt_mask for
  preventing recursive trap handler.
- vm_core.h (RUBY_VM_CHECK_INTS, RUBY_VM_CHECK_INTS_BLOCKING): ditto.
- thread.c (rb_threadptr_execute_interrupts):
  don't process interrupt if it is masked.
  [Bug #6009] [ruby-core:42524]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@37861 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 02/13/2012 09:54 PM - kosaki (Motohiro KOSAKI)

- Status changed from Open to Assigned

- Assignee set to kosaki (Motohiro KOSAKI)

#2 - 11/26/2012 07:57 PM - kosaki (Motohiro KOSAKI)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r37861.
Eric, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- ruby_atomic.h (ATOMIC_CAS): new macro for compare-and-exchange.
- vm_core.h (struct rb_thread_struct): add interrupt_mask member.
- thread.c (thread_create_core, Init_Thread): initialize th->thread_mask.
- vm_core.h (RUBY_VM_INTERRUPTED_ANY): new macro for avoiding bare th->interrupt_flag.
- vm_core.h (RUBY_VM_INTERRUPTED, RUBY_VM_INTERRUPTED): check th->interrupt_mask.
- thread.c (set_unblock_function, rb_thread_schedule): replace th->interrupt_flag with RUBY_VM_INTERRUPTED_ANY()
- signal.c (signal_exec): set up thread->interrupt_mask for preventing recursive trap handler.
- vm_core.h (RUBY_VM_CHECK_INTS, RUBY_VM_CHECK_INTS_BLOCKING): ditto.
- thread.c (rb_threadptr_execute_interrupts): don't process interrupt if it is masked. [Bug #6009] [ruby-core:42524]