# Ruby - Bug #5811

## Ruby Process Deadlocks With Fork on Mac OS X Lion

12/27/2011 03:42 AM - netshade (Chris Zelenak)

| | | | |
|---|---|---|---|
| **Status:** | Closed | | |
| **Priority:** | Normal | | |
| **Assignee:** | akr (Akira Tanaka) | | |
| **Target version:** | | | |
| **ruby -v:** | ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-darwin11.2.0] | **Backport:** | |

**Description**

=begin
Given a Ruby process that acts like the following:

- Spawn new thread that initializes a TCPSocket
- Execute script using backticks in main thread

there is a chance that it will deadlock on Lion. The GDB traces for the threads show:

- The TCP connecting thread stuck on native_cond_wait/thread_pthread.c:321 by way of rsock_getaddrinfo/raddrinfo.c:359
- The main thread stuck on read() by way of rb_f_backquote/io.c:7266

Meanwhile, in the forked process from rb_f_backquote:

- The main thread is stuck at (longer trace):

```
#0  0x00007fff9160c6b6 in semaphore_wait_trap ()
#1  0x00007fff8fc03bc2 in _dispatch_thread_semaphore_wait ()
#2  0x00007fff8fc04286 in dispatch_once_f ()
#3  0x00007fff95e12f20 in si_module_static_search ()
#4  0x00007fff95e16a3d in si_module_with_name ()
#5  0x00007fff95e0eac8 in getpwuid ()
#6  0x00007fff90daa842 in getgroups$DARWIN_EXTSN ()
#7  0x000000010b82b020 in rb_group_member (gid=0) at file.c:1002
#8  0x000000010b82b10f in eaccess (path=0x7fff6b3d3570 "/bin/hostname", mode=1) at file.c:1052
...
```

The documentation for getpwuid in Mac OS X Lion states that getpwuid now is threadsafe, much like getpwuid_r - however, the values returned by getpwuid are thread local and disposed automatically, as opposed to getpwuid_r's allocation of results. The disassembly of semaphore_wait_trap and __psynch_cvwait both show syscalls being made (I don't know how to go much further here), but the arguments are all void to these functions too when snooping in GDB. I believe that the posix wait and semaphore_wait taking place are in fact making syscalls to wait on a condition variable of the same value - this value is the same due to the shared memory state of the fork.

When an artificial delay ("sleep 1") is introduced after the creation of the TCP connect thread, this deadlock no longer occurs.

Attached is a test script that uses the Instrumental Agent gem for the TCP connect and can reliably cause the deadlock under 1.9.3.
=end

### History

**#1 - 03/11/2012 05:09 PM - ko1 (Koichi Sasada)**

*- Assignee set to mrkn (Kenta Murata)*

**#2 - 03/18/2012 06:46 PM - shyouhei (Shyouhei Urabe)**

*- Status changed from Open to Assigned*

**#3 - 07/14/2012 04:41 PM - mrkn (Kenta Murata)**

*- Status changed from Assigned to Feedback*

I need Gemfile to run your test.rb.
Please attach it.

**#4 - 01/26/2013 06:47 AM - drbrain (Eric Hodel)**

*- Status changed from Feedback to Rejected*

The requested Gemfile was not attached by the submitter so I am rejecting this.

**#5 - 02/08/2013 05:02 AM - samg (Sam Goldstein)**

*- File socket_backtick_test.rb added*

We've encountered the same issue on Mac OS X (Lion and Mountain Lion).  It's been causing one of our test suites to hang regularly and the process can only be stopped with a kill -9.

We've created a reproduction case which reproduces the issue every time on my machine, running OS 10.7.3, with ruby 1.9.3p286 (2012-10-12 revision 37165) [x86_64-darwin11.3.0]

The bug seems to be triggered by the timing of calls to eaccess so it may be necessary to mess with the repro case's sleep duration or to sleep(rand) on other machines.

When attaching to the hung process in GDB we get the following backtrace:

```
0x00007fff96c6c6b6 in semaphore_wait_trap ()
(gdb) bt
#0  0x00007fff96c6c6b6 in semaphore_wait_trap ()
#1  0x00007fff9246fbc2 in _dispatch_thread_semaphore_wait ()
#2  0x00007fff92470286 in dispatch_once_f ()
#3  0x00007fff99048cb0 in si_module_static_search ()
#4  0x00007fff9904ca05 in si_module_with_name ()
#5  0x00007fff99044858 in getpwuid ()
#6  0x00007fff95889842 in getgroups$DARWIN_EXTSN ()
#7  0x000000010d7d6895 in rb_group_member ()
#8  0x000000010d7d6c67 in eaccess ()
#9  0x000000010d7bffc1 in dln_find_1 ()
#10 0x000000010d7c01bd in dln_find_exe_r ()
#11 0x000000010d85073d in proc_exec_v ()
#12 0x000000010d8539e2 in rb_proc_exec ()
#13 0x000000010d853a9b in rb_exec_err ()
#14 0x000000010d84cdb5 in chfunc_protect ()
#15 0x000000010d7cd858 in rb_protect ()
#16 0x000000010d852d2b in rb_fork_err ()
#17 0x000000010d7f3c0e in pipe_open ()
#18 0x000000010d7f8e53 in rb_f_backquote ()
#19 0x000000010d90e967 in vm_call_method ()
#20 0x000000010d8fceb8 in vm_exec_core ()
#21 0x000000010d901453 in vm_exec ()
#22 0x000000010d911201 in loop_i ()
#23 0x000000010d7cdaf7 in rb_rescue2 ()
#24 0x000000010d8f4cd6 in rb_f_loop ()
#25 0x000000010d90e967 in vm_call_method ()
#26 0x000000010d8fceb8 in vm_exec_core ()
#27 0x000000010d901453 in vm_exec ()
#28 0x000000010d901828 in rb_iseq_eval_main ()
#29 0x000000010d7cdd62 in ruby_exec_internal ()
#30 0x000000010d7d085c in ruby_run_node ()
#31 0x000000010d78e73f in main ()
```

I've attached the reproduction case to this comment. Let us know if there's anything we can do to help reproduce/resolve this.

**#6 - 02/08/2013 01:44 PM - kosaki (Motohiro KOSAKI)**

*- Status changed from Rejected to Assigned*

*- Assignee changed from mrkn (Kenta Murata) to akr (Akira Tanaka)*

AFAIK, pipe and command are unsafe if multi thread is used. this issues was fixed at trunk (aka 2.0).

akr-san, do you have any commnet?

**#7 - 02/08/2013 02:41 PM - akr (Akira Tanaka)**

I guess this problem is caused by Ruby 1.9.3 invokes non-async-signal-safe functions, such as getpwuid(), in a forked child process before exec.

Ruby trunk fixed the cause: dln_find_exe_r() is called from a parent process, for example.

So, I'd like to know Ruby trunk actually fix this problem or not.
(I can't test because I don't have Mac.)

Anyway, I feel the change in Ruby trunk is too big to backport to Ruby 1.9.3, though.

**#8 - 02/12/2013 11:19 AM - kosaki (Motohiro KOSAKI)**

*- Status changed from Assigned to Closed*

OK. then I will close this ticket.

Please reopen this if anyone hit the same issue on 2.0 or trunk.

**#9 - 03/30/2013 05:01 AM - samg (Sam Goldstein)**

This does appear to be fixed in Ruby 2.0.0-p0.  I'm no longer able to get my reproduction case to hang (It doesn't hang every time in 1.9.3, but it only takes a try or two to trigger the deadlock).

I'm curious if there's any suggested workaround for this in ruby 1.9.3.  I develop a library (newrelic) so I can't control the ruby version which it is run under.  Generally it's okay if the backticked command fails (since we can recover from that) but deadlocking the whole process is obviously problematic.

Thanks!

## Files

| | | | |
|---|---|---|---|
| test.rb | 331 Bytes | 12/27/2011 | netshade (Chris Zelenak) |
| socket_backtick_test.rb | 270 Bytes | 02/08/2013 | samg (Sam Goldstein) |