## Ruby master - Bug #11336

**TestProcess#test_exec_fd_3_redirect failed on Solaris 10**

07/07/2015 09:36 AM - ngoto (Naohisa Goto)

| Status:  | Closed                  |
| Priority: | Normal                  |
| Assignee: |                        |
| Target version: |                |
| ruby -v: | ruby 2.3.0dev (2015-07-06) [sparc64-solaris2.10] |
| Backport: | 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN |

### Description

On Solaris 10, TestProcess#test_exec_fd_3_redirect failed since r51146 (where the test was added).

1) Failure:

TestProcess#test_exec_fd_3_redirect [/XXXXX-51146/test/ruby/test_process.rb:2049]:

<"."> expected but was <nil>.

### Related issues:

- Related to Ruby master - Bug #11350: When Process.exec failed, redirections w... Closed
- Related to Ruby master - Bug #11353: ASYNC BUG after failure of Process.exec ... Closed

### Associated revisions

**Revision 4d23a784 - 07/10/2015 11:27 AM - ngoto (Naohisa Goto)**

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]

  git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51209 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 51209 - 07/10/2015 11:27 AM - ngoto (Naohisa Goto)**

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]

**Revision 51209 - 07/10/2015 11:27 AM - ngoto (Naohisa Goto)**

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]

**Revision 51209 - 07/10/2015 11:27 AM - ngoto (Naohisa Goto)**

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]

**Revision 51209 - 07/10/2015 11:27 AM - ngoto (Naohisa Goto)**

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]

**Revision 51209 - 07/10/2015 11:27 AM - ngoto (Naohisa Goto)**

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]

**Revision f242e17f - 07/16/2020 01:06 PM - ngoto (Naohisa Goto)**

- process.c (redirect_dup2): when the new FD of dup2() conflicts
with one of the timer thread FDs, the internal FD is diverted.
[Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]

- process.c (dup2_with_divert): new function for the above purpose.
- thread_pthread.c (rb_divert_reserved_fd): new function for
diverting reserved FD. If the given FD is the same as one of the
reserved FDs, the reserved FD number is internally changed.
It returns -1 when error. Otherwise, returns 0. It also returns
0 if there is no need to change reserved FD number.
- thread_win32.c (rb_divert_reserved_fd): always returns 0 because
of no reserved FDs.
- internal.h (rb_divert_reserved_fd): prototype declaration.
  It is Ruby internal use only.

Revision 51268 - 07/16/2015 01:06 PM - ngoto (Naohisa Goto)

- process.c (redirect_dup2): when the new FD of dup2() conflits
  with one of the timer thread FDs, the internal FD is diverted.
  [Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]
- process.c (dup2_with_divert): new function for the above purpose.
- thread_pthread.c (rb_divert_reserved_fd): new function for
diverting reserved FD. If the given FD is the same as one of the
reserved FDs, the reserved FD number is internally changed.
It returns -1 when error. Otherwise, returns 0. It also returns
0 if there is no need to change reserved FD number.
- thread_win32.c (rb_divert_reserved_fd): always returns 0 because
  of no reserved FDs.
- internal.h (rb_divert_reserved_fd): prototype declaration.
  It is Ruby internal use only.

Revision 51268 - 07/16/2015 01:06 PM - ngoto (Naohisa Goto)

- process.c (redirect_dup2): when the new FD of dup2() conflits
  with one of the timer thread FDs, the internal FD is diverted.
  [Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]
- process.c (dup2_with_divert): new function for the above purpose.
- thread_pthread.c (rb_divert_reserved_fd): new function for
diverting reserved FD. If the given FD is the same as one of the
reserved FDs, the reserved FD number is internally changed.
It returns -1 when error. Otherwise, returns 0. It also returns
0 if there is no need to change reserved FD number.
- thread_win32.c (rb_divert_reserved_fd): always returns 0 because
  of no reserved FDs.
- internal.h (rb_divert_reserved_fd): prototype declaration.
  It is Ruby internal use only.
• process.c (redirect_dup2): when the new FD of dup2() conflicts with one of the timer thread FDs, the internal FD is diverted. [Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]

• process.c (dup2_with_divert): new function for the above purpose.

• thread_pthread.c (rb_divert_reserved_fd): new function for diverting reserved FD. If the given FD is the same as one of the reserved FDs, the reserved FD number is internally changed. It returns -1 when error. Otherwise, returns 0. It also returns 0 if there is no need to change reserved FD number.

• thread_win32.c (rb_divert_reserved_fd): always returns 0 because of no reserved FDs.

• internal.h (rb_divert_reserved_fd): prototype declaration. It is Ruby internal use only.

Revision 51268 - 07/16/2015 01:06 PM - ngoto (Naohisa Goto)

• process.c (redirect_dup2): when the new FD of dup2() conflicts with one of the timer thread FDs, the internal FD is diverted. [Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]

• process.c (dup2_with_divert): new function for the above purpose.

• thread_pthread.c (rb_divert_reserved_fd): new function for diverting reserved FD. If the given FD is the same as one of the reserved FDs, the reserved FD number is internally changed. It returns -1 when error. Otherwise, returns 0. It also returns 0 if there is no need to change reserved FD number.

• thread_win32.c (rb_divert_reserved_fd): always returns 0 because of no reserved FDs.

• internal.h (rb_divert_reserved_fd): prototype declaration. It is Ruby internal use only.

Revision 51268 - 07/16/2015 01:06 PM - ngoto (Naohisa Goto)

• process.c (redirect_dup2): when the new FD of dup2() conflicts with one of the timer thread FDs, the internal FD is diverted. [Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]

• process.c (dup2_with_divert): new function for the above purpose.

• thread_pthread.c (rb_divert_reserved_fd): new function for diverting reserved FD. If the given FD is the same as one of the reserved FDs, the reserved FD number is internally changed. It returns -1 when error. Otherwise, returns 0. It also returns 0 if there is no need to change reserved FD number.

• thread_win32.c (rb_divert_reserved_fd): always returns 0 because of no reserved FDs.

• internal.h (rb_divert_reserved_fd): prototype declaration. It is Ruby internal use only.

History

#1 - 07/07/2015 04:58 PM - normalperson (Eric Wong)

Was there any output in stderr from this test? Thanks.
Was there any output in stderr from this test?
nothing

Recently, the failure message is changed to below. It seems the SIGKILL is caused by timeout of the test.

TestProcess#test_exec_fd_3_redirect [/XXXXX-51186/test/ruby/test_process.rb:2038]:
assert_separately failed
pid 9248 killed by SIGKILL (signal 9)

Can you try a higher fd number instead of 3? Also, truss output would be useful. Thanks.

Sometimes ASYNC BUG occurs, but it cannot be reproduced with truss.

$ ruby -e 'a = IO.pipe; b = IO.pipe; p a; p b; pid = fork { exec("ruby", "-e", "print IO.for_fd(3).read(1)", 3 =>a[0],1=>b[1] ); b[1].close; a[0].close; a[1].write("."); p b[0].read(1); Process.wait(pid)'
[<IO:fd 7>, <IO:fd 8>]
[<IO:fd 9>, <IO:fd 10>]
[ASYNC BUG] consume_communication_pipe: read
EBADF

OK, I also hit the problem on a VM, too. The problem is the timer thread is still running when we are performing redirects for exec.

Can you try the following to stop the timer thread?

```c
--- a/process.c
+++ b/process.c
@@ -2566,7 +2566,7 @@ rb_f_exec(int argc, const VALUE *argv)
 #if defined(__APPLE__) || defined(__HAIKU__)
     before_exec_without_timer_thread(eargp, errmsg, sizeof(errmsg));
 #else
-    before_exec_async_signal_safe(); /* async-signal-safe */
+    before_exec(); /* NOT async-signal-safe */
     preserving_errno(after_exec_async_signal_safe()); /* async-signal-safe */
 #endif
```

I will have limited Internet access the next few days. Feel free to commit if it works for you; maybe the timer thread needs to
be restarted if execve fails, too (but the process will die).

Also, maybe the APPLE || HAIKU code above is suitable for all
OS, too.

In the coming weeks, we may also consider lazy spawning timer thread,
single-threaded scripts do not need it.

#6 - 07/08/2015 11:03 AM - ngoto (Naohisa Goto)

> Also, maybe the APPLE || HAIKU code above is suitable for all
OS, too.

I think so, and the following patch solves the failure of TestProcess#test_exec_fd_3_redirect on Solaris 10.

Index: process.c
===================================================================
--- process.c (revision 51194)
+++ process.c (working copy)
@@ -2473,7 +2473,7 @@
    RB_GC_GUARD(execarg_obj);
 }

-#if defined(__APPLE__) || defined(__HAIKU__)
+#if defined(__APPLE__) || defined(__HAIKU__) || defined(__sun)
    static int rb_exec_without_timer_thread(const struct rb_execarg *eargp, char *errmsg, size_t errmsg_buflen);
 #endif

@@ -2563,7 +2563,7 @@
    fail_str = eargp->use_shell ? eargp->invoke.sh.shell_script : eargp->invoke.cmd.command_name;
-#if defined(__APPLE__) || defined(__HAIKU__)
+#if defined(__APPLE__) || defined(__HAIKU__) || defined(__sun)
    rb_exec_without_timer_thread(eargp, errmsg, sizeof(errmsg));
 #else
    before_exec_async_signal_safe(); /* async-signal-safe */
@@ -3077,7 +3077,7 @@
    return -1;
 }

-#if defined(__APPLE__) || defined(__HAIKU__)
+#if defined(__APPLE__) || defined(__HAIKU__) || defined(__sun)
    static int
    rb_exec_without_timer_thread(const struct rb_execarg *eargp, char *errmsg, size_t errmsg_buflen)
    { 

#7 - 07/10/2015 09:46 AM - ngoto (Naohisa Goto)

The following example can reproduce the problem that the timer thread reads fd=3 after the redirection to fd=3 is set and before execve() is called, and then the script waits forever in b[0].read(1) and/or IO.for_fd(3).read(1).

ruby -e 'a = IO.pipe; b = IO.pipe; p a; p b; opt = { 3=>a[0], 1=>b[1] }; (5..50).each { |i| opt[i] = "/dev/zero" }; pid = fork { exec("ruby", "-e", "print IO.for_fd(3).read(1)", opt); }[b[1].close; a[0].close; a[1].write( "."); p b[0].read(1); Process.wait(pid) }

So, I also think stopping timer thread before execve is needed on all OS.

#8 - 07/10/2015 09:52 AM - ngoto (Naohisa Goto)

I forgot to mention that this was reproduced on x86_64 Linux.

#9 - 07/10/2015 11:28 AM - ngoto (Naohisa Goto)

- Status changed from Open to Closed

Applied in changeset r51209.

- process.c (rb_f_exec): rb_exec_without_timer_thread should be used on every OS, not only on Mac OS X or Haiku, to prevent timer thread communications using file descriptor 3 after setting redirection of the fd 3 before calling exec. [Bug #11336] [ruby-core:69886]
#10 - 07/14/2015 04:28 AM - ngoto (Naohisa Goto)
- Related to Bug #11350: When Process.exec failed, redirections were still changed and not restored added

#11 - 07/15/2015 02:24 PM - ngoto (Naohisa Goto)
- Related to Bug #11353: ASYNC BUG after failure of Process.exec when closing FD 3 (or 4 or 5) added

#12 - 07/16/2015 12:11 PM - ngoto (Naohisa Goto)
- Status changed from Closed to Open

In r51265, I reverted r51209 because signals after stopping timer thread before entering exec(2) system call might be lost.

On Mac OS X and Haiku, the loss of signal still might occur.

#13 - 07/16/2015 01:06 PM - ngoto (Naohisa Goto)
- Status changed from Open to Closed

Applied in changeset r51268.

- process.c (redirect_dup2): when the new FD of dup2() conflicts with one of the timer thread FDs, the internal FD is diverted.
  [Bug #11336] [ruby-core:69886] [Bug #11350] [ruby-core:69961]

- process.c (dup2_with_divert): new function for the above purpose.

- thread_pthread.c (rb_divert_reserved_fd): new function for diverting reserved FD. If the given FD is the same as one of the reserved FDs, the reserved FD number is internally changed. It returns -1 when error. Otherwise, returns 0. It also returns 0 if there is no need to change reserved FD number.

- thread_win32.c (rb_divert_reserved_fd): always returns 0 because of no reserved FDs.

- internal.h (rb_divert_reserved_fd): prototype declaration. It is Ruby internal use only.

#14 - 07/16/2015 09:28 PM - normalperson (Eric Wong)
Sorry, but I think rb_divert_reserved_fd seems a racy fix. I think the correct fix is to allow single-threaded Ruby to process signals without timer thread; I will try to make it happen now.

#15 - 07/17/2015 06:27 AM - ngoto (Naohisa Goto)

Sorry, but I think rb_divert_reserved_fd seems a racy fix.

Yes, it might be. Comparison and replacement of the fd is not atomic there, and calling dup2 after the rb_divert_reserved_fd is also a source of race. Indeed, I implicitly expected that rb_divert_reserved_fd and dup2_with_divert were executed with GVL, but this might not be true in some cases.

#16 - 07/18/2015 03:48 AM - normalperson (Eric Wong)
I started working on this patch which causes pipes to always be closed during timer thread shutdown.

http://80x24.org/spew/m/839ba256acbf1a5b17ad60953fc3ecf1df6ce50.txt

However, I may also eliminate timer thread entirely (on POSIX) by using timer_create+timer_settime to deliver SIGVTALRM.

I also eliminated malloc from signal thread list (but not FGLOCK, yet), but eliminating malloc may not be necessary.

http://80x24.org/spew/m/1437190754-29067-1-git-send-email-e@80x24.org.txt
With the patch, when exec system call fails, signals that are sent after stopping the timer thread before re-starting the timer thread may be ignored.

Currently, except Mac OS X and Haiku, the signals could be invoked after the failure of exec with no loss.

I'm not seeing it, did you notice/reproduce this failure?

I see:

1) rb_thread_stop_timer_thread()

2) signal received:
   sighandler
   -> signal_enqueue -> ATOMIC_INC(signal_buff.size) (++)
   -> rb_thread_wakeup_timer_thread -> noop since TT not running

3) exec fails

4) rb_thread_start_timer_thread()
   -> pthread_create(..., thread_timer)
   -> thread_timer
   -> timer_thread_function
   -> rb_threadptr_check_signal
   -> if (rb_signal_buff_size() > 0) { /* true from (++) /
      / continue processing signals */

In other words, I'm not seeing a problem based on the (++) steps.

I started working on this patch which causes pipes to always be closed during timer thread shutdown.

http://80x24.org/spew/m/839ba256acbf1a5b17ad60953fc3ecf1df6ec50.txt

OK, need to check FD validity, too. Still tricky:
http://80x24.org/spew/m/1437379741-6844-1-git-send-email-e@80x24.org.txt

However, I may also eliminate timer thread entirely on POSIX) by using timer_create+timer_settime to deliver SIGVTALRM.

Maybe...

I'm sorry too late.

On Solaris 10, the patch works fine, with no error/failure during make test-all (with r51450).

I'm not seeing it, did you notice/reproduce this failure?

No.

It seems the patch works correctly as you described.

I'm wondering whether the TODO in the following comment is still valid or not.

It seems it was first added in r32244 and was modified in r32276.

/* close communication pipe */
if (close_anyway) {
    /* TODO: Uninstall all signal handlers or mask all signals. */
    This pass is cleaning phase (terminate ruby process).
To avoid such race, we skip to close communication pipe. OS will close it at process termination.
It may not good practice, but pragmatic.
We remain it is TODO.

*/
/* close.communication.pipe(); */

#21 - 07/31/2015 09:08 PM - normalperson (Eric Wong)
ngotogenome@gmail.com wrote:
I'm sorry too late.

No worries, I don't have much time for ruby these few weeks, either.

On Solaris 10, the patch works fine, with no error/failure during make test-all (with r51450).
I'm not seeing it, did you notice/reproduce this failure?

No.
It seems the patch works correctly as you described.
I'm wondering whether the TODO in the following comment is still valid or not.
It seems it was first added in r32244 and was modified in r32276.

Not valid, I removed it in this patch [ruby-core:70050]:
http://80x24.org/spew/m/1437379741-6844-1-git-send-email-e@80x24.org.txt

#22 - 08/14/2015 08:33 AM - ngoto (Naohisa Goto)
- Status changed from Closed to Open

I think the patch http://80x24.org/spew/m/1437379741-6844-1-git-send-email-e@80x24.org.txt is better not only for Solaris but also for other OSs.

#23 - 08/14/2015 03:27 PM - ngoto (Naohisa Goto)
- Status changed from Open to Closed

memo:
r51576 fixes the bug (+ r51578 suppresses warning message)
r51268 is reverted by r51576