Ruby master - Feature #2715

Optimization to avoid spawning shell in Kernel#system call should check for failure conditions

02/06/2010 06:54 AM - taw (Tomasz Wegrzanowski)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

=begin
This is an old issue, I think going all the way back to Perl's system().

Kernel#system is supposed to spawn shell and pass its argument to shell process just like C system(); except it's optimized for a common case of very simple argument, in which case Ruby (like Perl) simply does fork and exec of the string passed, split on whitespace.

This almost works, except shell error reporting is not duplicated. If command doesn't exist, shell would print an error message on stderr, while Ruby like Perl fails silently.

Problem is present in all versions of Ruby including 1.8.7 and 1.9.1.

To reproduce:
$ ruby -e 'system "fail"'
$ ruby -e 'system "\"\"fail"'
sh: fail: command not found

Output of both commands should be identical (or at least close enough, details of error message might differ by Unix flavor).

rb_f_system code for VMS and Windows uses different code paths, so I don't know if it's also affected or not.

I wrote a bit more about background of this issue on my blog:
http://t-a-w.blogspot.com/2010/02/what-is-all-this-perl-doing-in-my-ruby.html
=end

Related issues:

Related to Ruby master - Feature #3426: exec doesn't allow command lines whic...	**Rejected**	**06/11/2010**
Related to Ruby master - Feature #4477: Kernel:exec and backtick (`) don't wo...	**Closed**	**03/07/2011**

History

#1 - 02/06/2010 11:27 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

=begin

=end

#2 - 02/06/2010 11:30 AM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Sat, 6 Feb 2010 06:55:09 +0900,
Tomasz Wegrzanowski wrote in [ruby-core:28072]:

> This almost works, except shell error reporting is not
> duplicated. If command doesn't exist, shell would print an
> error message on stderr, while Ruby like Perl fails silently.
>
> Problem is present in all versions of Ruby including 1.8.7 and 1.9.1.
>
> To reproduce:
> $ ruby -e 'system "fail"'
> $ ruby -e 'system "\"\"fail"'
> sh: fail: command not found

> Output of both commands should be identical (or at least
> close enough, details of error message might differ by Unix
> flavor).

What's that you want to do? In 1.9, the former returns nil,
while the later returns false, and spawn and Process.spawn
raise an Errno::ENOENT.

In addtion, system and spawn in 1.9 have been improved very
much, so you don't need to use sh for redirection.

# run "fail" command with no output
system("fail", out:"/dev/null", err:[:child, :out])

> I wrote a bit more about background of this issue on my blog:
> http://t-a-w.blogspot.com/2010/02/what-is-all-this-perl-doing-in-my-ruby.html

Please describe the rationale briefly on the BTS, not separated
blog.

--
Nobu Nakada

=end

#3 - 02/12/2010 04:02 PM - taw (Tomasz Wegrzanowski)

=begin
Nobu: I want system "fail" and system ""fail" to behave identically - printing error message when command fail doesn't exist.

For Ruby's system() not to spawn shell should be just performance optimization, it should behave the same way regardless of it spawns shell (system ""fail") or not (system "fail").

That's all.
=end

#4 - 05/30/2010 06:11 PM - mame (Yusuke Endoh)

=begin
Hi,

> For Ruby's system() not to spawn shell should be just performance optimization

I don't think so. Indeed, it is a feature for optimization, but
it is also spec.

Process invocation may cause significant result such as breaking
system. So changing that may cause compatibility issue.

> Nobu: I want system "fail" and system ""fail" to behave identically - printing error message when command fail doesn't exist.

"Printing error message" is absolutely new feature.
I move this ticket to feature tracker, though the feature request
is really uncool for me...

--
Yusuke Endoh mame@tsg.ne.jp
=end

#5 - 06/26/2011 01:51 PM - akr (Akira Tanaka)

- Project changed from Ruby to Ruby master

- Category changed from core to core

#6 - 03/18/2012 03:22 PM - akr (Akira Tanaka)

- Description updated

I feel the optimization (avoid shell when no meta characters) can be removed.

We have many ways to avoid shell such as system(command, arg1, arg2).

And recent computers are fast enough to run command via shell.

#7 - 03/25/2012 01:49 PM - mame (Yusuke Endoh)

- Assignee set to akr (Akira Tanaka)

#8 - 06/04/2012 10:34 PM - akr (Akira Tanaka)

- File remove-shell-invocation-optimization.patch added

I wrote a patch to remove the optimization:
remove-shell-invocation-optimization.patch

However it was not good enough because it causes several test failures.

Several failures are caused by pid change which the invoked command is not
direct child process of ruby process.
(This depends on /bin/sh. If /bin/sh optimizes tail invocations,
invoked command process will be a direct child process.)

I think this is incompatible enough.

Now, I feel printing a error message suggested by taw is better solution.

#9 - 06/06/2012 01:40 AM - kosaki (Motohiro KOSAKI)

Hmm...

As akr-san described, some shells have tail invocation optimization. Thus, even if ruby doesn't have shell invocation optimization, caller can't assume that spawned process is neither child nor grand child.

Therefore, if some tests were fail, it is certainly test mistake. I still think we can remove the shell invocation optimization. It is a source of confusion and I dislike it.

Of cource, it shouldn't be backported 1.9.x.

#10 - 06/06/2012 04:55 AM - akr (Akira Tanaka)

> As akr-san described, some shells have tail invocation optimization. Thus, even if ruby doesn't have shell invocation optimization, caller can't assume that spawned process is neither child nor grand child.

If we have shell invocation optimization (as now) and the command line is
simple, the command is spawned as a child process.

So removing the optimization breaks a program which assumes this behavior.

For example, IO.popen("foo bar") {|io| ... Process.kill sig, io.pid ... }
will be broken by removing the optimization.

I feel this is incompatible enough.

#11 - 06/07/2012 05:23 AM - kosaki (Motohiro KOSAKI)

(6/5/12 3:55 PM), akr (Akira Tanaka) wrote:

> Issue #2715 has been updated by akr (Akira Tanaka).
>
> > As akr-san described, some shells have tail invocation optimization. Thus, even if ruby doesn't have shell invocation optimization, caller can't assume that spawned process is neither child nor grand child.
>
> If we have shell invocation optimization (as now) and the command line is
> simple, the command is spawned as a child process.
>
> So removing the optimization breaks a program which assumes this behavior.
>
> For example, IO.popen("foo bar") {|io| ... Process.kill sig, io.pid ... }
> will be broken by removing the optimization.

> I feel this is incompatible enough.

.... I realized IO.pid is crazy mistaken feature. It wouldn't work when using
complex cmdline even if we will not remove the optimization. Sigh. (+_+

#12 - 06/08/2012 04:21 PM - akr (Akira Tanaka)

> .... I realized IO.pid is crazy mistaken feature. It wouldn't work when using complex cmdline even if we will not remove the optimization. Sigh. (+_+

Killing a complex command line, which invokes many processes, is a different problem.
IO#pid is not a origin of the problem.
spawn also have same problem as follows.

pid = spawn("foo & bar"); Process.kill(:TERM, pid)

Do you say that returning pid feature of spawn is crazy mistaken?

Killing a complex command line needs process group,
as shell job control does.

To return to this issue, I'm considering adding a new spawn option, :shell.
This option disables the optimization.

system("simple command", :shell=>true)

The default of the option is disabled as now.

This option solves this issue as:

- shell error messages are printed if :shell => true.
- doesn't break programs which expects simple commands are direct child process.
- don't need to implement mimic of shell error message printing feature in ruby.

#13 - 06/08/2012 04:29 PM - kosaki (Motohiro KOSAKI)

On Fri, Jun 8, 2012 at 3:21 AM, akr (Akira Tanaka) akr@fsij.org wrote:

> Issue #2715 has been updated by akr (Akira Tanaka).
>
> > .... I realized IO.pid is crazy mistaken feature. It wouldn't work when using complex cmdline even if we will not remove the optimization. Sigh. (+_+
>
> Killing a complex command line, which invokes many processes, is a different problem.
> IO#pid is not a origin of the problem.
> spawn also have same problem as follows.
>
> pid = spawn("foo & bar"); Process.kill(:TERM, pid)
>
> Do you say that returning pid feature of spawn is crazy mistaken?
>
> Killing a complex command line needs process group,
> as shell job control does.
>
> To return to this issue, I'm considering adding a new spawn option, :shell.
> This option disables the optimization.
>
> system("simple command", :shell=>true)
>
> The default of the option is disabled as now.
>
> This option solves this issue as:
>
> - shell error messages are printed if :shell => true.
> - doesn't break programs which expects simple commands are direct child process.
> - don't need to implement mimic of shell error message printing feature in ruby.

Seems make sense.

#14 - 11/20/2012 11:00 PM - mame (Yusuke Endoh)

- Target version set to 2.6

#15 - 12/25/2017 06:00 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

- Assignee deleted (akr (Akira Tanaka))

Files

remove-shell-invocation-optimization.patch	5.07 KB	06/04/2012	akr (Akira Tanaka)