Ruby - Feature #2351

system() hardlinked to /bin/sh

11/09/2009 11:41 PM - Comradin (Marcus Franke)

Status:	Rejected	
Priority:	Normal	
Assignee:		
Target version:		

Description

=begin
I encountered this problem while using a feature of the very powerful zsh(ell).

In zsh one can use regular expressions in shell commands, like accessing files in
different locations with one command:

cat -la /mnt/server(1|2)/some/path/and/file_log

The default /bin/sh does not as it is a bash in most cases.

Example:
% irb

```
        ENV['SHELL']
        => "/usr/bin/zsh"
        system('cat /mnt/server(1|2|3)/file | sed, awk, etc.')
        sh: Syntax error: "(" unexpected
        => false
```

I can update the ENV['SHELL'] to every setting I'd like, but system() will always refer to /bin/sh.

Thus my script will always fail as long as I do not circumvent this feature with something like:

system('/usr/bin/zsh -c "cat /mnt/server(1|2|3)/file | sed, awk, etc."')

I tested this with different ruby versions installed by different Linux distributions:
ruby 1.8.7 (2008-08-11 patchlevel 72) [i486-linux] Ubuntu 9.04
ruby 1.8.7 (2008-08-11 patchlevel 72) [x86_64-linux] openSuSE 11.1
ruby 1.8.7 (2009-04-08 patchlevel 160) [i386-freebsd7]
ruby 1.8.5 (2006-08-25) [i386-linux] CentOS release 5.3 (Final)
=end

History

#1 - 11/25/2010 01:59 PM - shyouhei (Shyouhei Urabe)

=begin
Moving this to the feature request tracker.

Kernel#system's invoking /bin/sh is not 1.8.7 specific.
=end

#2 - 11/26/2010 10:37 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

SUS says

> The environment of the executed command shall be as if a child process were created using fork(), and the child process invoked the sh utility using execl() as follows:
>
> execl(, "sh", "-c", command, (char *)0);

> where is an unspecified pathname for the sh utility.

In other word, it doesn't refer SHELL environemnt. Now ruby does so too because it is easy understandable.
Moreover you can invoke zsh easily by you wrote or similar way.

=end

#3 - 11/26/2010 11:18 PM - kosaki (Motohiro KOSAKI)

=begin
And if we accept this proposal, system() will lose user enviroment portability. e.g. csh has completely different syntax from /bin/sh. Thus, distributed ruby scripts will not be able to use system().

=end

#4 - 11/27/2010 08:58 AM - meta (mathew murphy)

=begin
On Fri, Nov 26, 2010 at 08:18, Motohiro KOSAKI redmine@ruby-lang.orgwrote:

> And if we accept this proposal, system() will lose user enviroment
> portability. e.g. csh has completely different syntax from /bin/sh. Thus,
> distributed ruby scripts will not be able to use system().

Surely anyone using system() to execute a shell script will be using a
shebang (#!) line to specify the shell the script is supposed to be executed
by?

If you don't, you're likely to discover that ksh is the standard POSIX
shell, not sh; and then there's the whole issue of bash vs ash vs sh.

mathew

On Fri, Nov 26, 2010 at 08:18, Motohiro KOSAKI <redmine@ruby-lang.org> wrote:

> And if we accept this proposal, system() will lose user enviroment portability. e.g. csh has completely different syntax from /bin/sh. Thus, distributed ruby scripts will not be able to use system().

Surely anyone using system() to execute a shell script will be using a shebang (#!) line to specify the shell the script is supposed to be executed by?

If you don't, you're likely to discover that ksh is the standard POSIX shell, not sh; and then there's the whole issue of bash vs ash vs sh.

mathew

=end

#5 - 11/27/2010 09:36 PM - kosaki (Motohiro KOSAKI)

=begin
2010/11/27 mathew meta@pobox.com:

> On Fri, Nov 26, 2010 at 08:18, Motohiro KOSAKI redmine@ruby-lang.org
> wrote:
>
>> And if we accept this proposal, system() will lose user enviroment
>> portability. e.g. csh has completely different syntax from /bin/sh. Thus,
>> distributed ruby scripts will not be able to use system().
>
> Surely anyone using system() to execute a shell script will be using a
> shebang (#!) line to specify the shell the script is supposed to be executed
> by?

> If you don't, you're likely to discover that ksh is the standard POSIX
> shell, not sh; and then there's the whole issue of bash vs ash vs sh.

In practical world, peope who want distribute their own scripts are only using
ash subset. It works enough, I think.

=end

#6 - 03/18/2012 02:50 PM - nahi (Hiroshi Nakamura)

- Status changed from Open to Rejected

- Description updated

I close this as 'Rejected' since no update given long time.