Ruby master - Bug #14198

Error forwarding standard input to subprocess

12/18/2017 01:33 PM - betabandido (Victor Jimenez)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.2p198 (2017-09-14 revision 59899) [x64-mingw32]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

I am developing a wrapper for Terraform (https://www.terraform.io/), which at some point during its execution, it may request user input. So, my application must forward everything typed on its stdin to the subprocess' stdin. The following solution works on Linux, but on Windows the subprocess (Terraform) seems to never receive the input.

A similar approach works for wrappers implemented in Python or Go, so I believe this may actually be a bug in Ruby itself.

The file test.rb contains the code to reproduce the issue. First, Terraform is downloaded in the working directory. Then, popen3 is used to run Terraform.

History

#1 - 12/18/2017 01:46 PM - shyouhei (Shyouhei Urabe)

I din't have a windows machine right now so not sure this is the solution for you. But JFYI you can reroute the IOs by specifying arguments to spawn like this:

```
def exec(cmd)
  pid = Process.spawn(cmd, in: STDIN, out: STDOUT, err: STDERR)
  Process.waitpid pid
end
```

#2 - 12/18/2017 06:01 PM - betabandido (Victor Jimenez)

Using Process.spawn indeed works :) Unfortunately, I cannot figure out a way to forward the standard output and standard error to the console (i.e, STDOUT and STDERR) as well as storing that output in a variable.

The example in test.rb is a minimal example that just reproduces the issue where the subprocess does not receive any data on its standard input. But, the real code is a little bit more complex as it needs to store the output generated by the subprocess too.

Is there any easy way to do so? Is fork + exec and using pipes the only way to go?

#3 - 12/19/2017 12:12 AM - nobu (Nobuyoshi Nakada)

system also accepts redirections.
And child processes inherit STDIN/STDOUT/STDERR from their parent process by the default, you don't need such arguments at all usually.
If you want to capture the outputs and the errors, IO.popen(%w'terraform destroy', err: [:child, :out], &:read).

#4 - 12/19/2017 12:15 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

#5 - 01/04/2018 06:26 PM - betabandido (Victor Jimenez)

Thanks for the suggestions.

The following code snippet based on IO.popen seems to work. It executes a command, and it returns its output as an array containing the output lines. Optionally, the output is written to stdout too.

```
def run(cmd, directory: Dir.pwd, print_output: true)
  out = IO.popen(cmd, err: %i[child out], chdir: directory) do |io|
    begin
      out = ''
      loop do
        chunk = io.readpartial(4096)
        print chunk if print_output
        out += chunk
      end
    rescue EOFError; end
    out
  end

  $?.exitstatus.zero? || (raise "Error running command #{cmd}")

  out.split("\n")
     .map { |line| line.tr("\r\n", '') }
end
```

I am not sure whether there is an easier way to accomplish this, but this seems good enough.

#6 - 07/26/2019 10:46 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Feedback to Closed

Files

Gemfile	46 Bytes	12/18/2017	betabandido (Victor Jimenez)
test.rb	1.27 KB	12/18/2017	betabandido (Victor Jimenez)