Ruby - Bug #20206

PTY.spawn seems to fail to capture the output of "echo foo" once in a while

01/23/2024 09:00 PM - lacostej (Jerome Lacoste)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.3.0 (2023-12-25 revision 5124f9ac75) [arm64-darwin23]	**Backport:**	3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

We use PTY.spawn to call "echo foo", and on Mac it seems to randomly fail, capturing an empty output every now and then.
On Linux, the failure doesn't seem to happen.

The following code contains 2 ways of capturing the output from PTY.spawn. Both seem to show the same issue (run_command and run_command2).

```
require 'pty'
require 'expect'

def run_command(command)
  output = []
  PTY.spawn(command) do |command_stdout, command_stdin, pid|
    begin
      command_stdout.each do |l|
        line = l.chomp
        output << line
      end
    rescue Errno::EIO
      # This is expected on some linux systems, that indicates that the subcommand finished
      # and we kept trying to read, ignore it
    ensure
      command_stdout.close
      command_stdin.close
      Process.wait(pid)
    end
  end
  raise "#{$?.exited?} #{$?.stopped?} #{$?.signaled?} - #{$?.stopsig} - #{$?.termsig} -" unless $?
.exitstatus == 0
  [$?.exitstatus, output.join("\n")]
end

def run_command2(command)
  output = []
  PTY.spawn(command) do |command_stdout, command_stdin, pid|
    output = ""
    begin
      a = command_stdout.expect(/foo.*/, 5)
      output = a[0] if a
    ensure
      command_stdout.close
      command_stdin.close
      Process.wait(pid)
    end
  end
  raise "#{$?.exited?} #{$?.stopped?} #{$?.signaled?} - #{$?.stopsig} - #{$?.termsig} -" unless $?
.exitstatus == 0
  [$?.exitstatus, output]
end

def test_spawn(command)
  status, output = run_command(command)
```

```
  errors = []
  errors << "status was '#{status}'" unless status == 0
  errors << "output was '#{output}'" unless output == "foo"
  raise errors.join(" - ") unless errors.empty?
end

command = "echo foo"

puts "Will run command: '#{command}'"

errors = 0
2000.times do |i|
  begin
    test_spawn(command)
  rescue => e
    puts "ERROR #{i}: #{e}"
    errors += 1
  end
end

raise "Failed #{errors} times" unless errors == 0
```

Here are some ways of reproducing the issue.

```
ruby test_pty.rb
```

Use stress -c 16 -t 99 in the background to trigger the issue more often.

Here's an example of how it fails on circleci.
https://app.circleci.com/pipelines/github/lacostej/cienvs/33/workflows/d6d8e604-8a0d-4ede-8c44-d154dde93111

Tested on ruby 2.6 to ruby 3.3.0 on Mac.

History

#1 - 01/23/2024 09:06 PM - lacostej (Jerome Lacoste)

- Description updated

#2 - 02/26/2024 03:34 PM - lacostej (Jerome Lacoste)

Changing the command to "echo 'foo'" or "stdbuf -i0 -o0 -e0 echo foo" doesn't reproduce the failure.

So this could very much be caused by buffering issues on the terminal side and not a bug in ruby. I think we can close this issue.

#3 - 02/26/2024 03:40 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed