Ruby master - Bug #5556

SIGHUP no longer ignored when sent to process group from a subprocess

11/03/2011 08:07 AM - brixen (Brian Shirai)

Status:	Closed		
Priority:	Normal		
Assignee:	kosaki (Motohiro KOSAKI)		
Target version:	2.0.0		
ruby -v:	-	**Backport:**	

Description

Hi,

Prior to 2.0.0dev, this script:

```
sasha:rubinius brian$ cat process.rb
puts "before system"
system("ruby -e 'Process.kill(:HUP, 0)'")
puts "after system"
```

would print the following:

```
sasha:rubinius brian$ ruby1.9.2 -v process.rb
ruby 1.9.2p290 (2011-07-09 revision 32553) [x86_64-darwin10.8.0]
before system
after system
```

Basically, SIGHUP was ignored when sent from a subprocess. Presently, this is the result:

```
sasha:rubinius brian$ ruby -v process.rb
ruby 2.0.0dev (2011-11-01 trunk 33596) [x86_64-darwin10.8.0]
before system
Hangup
```

The following issue may be related, but the explanation is in Japanese, so I cannot follow it: http://redmine.ruby-lang.org/issues/4765

Is this change intentional? I discovered it running RubySpec, where there are specs for the behavior of sending SIGHUP to the process group.

Thanks,
Brian

History

#1 - 11/03/2011 09:02 AM - kosaki (Motohiro KOSAKI)

- Category set to core

- Status changed from Open to Assigned

- Assignee set to kosaki (Motohiro KOSAKI)

- Target version set to 2.0.0

#2 - 11/10/2011 02:19 AM - kosaki (Motohiro KOSAKI)

> Is this change intentional? I discovered it running RubySpec, where there are specs for the behavior of sending SIGHUP to the process group.

In short. Yes, intentional. SIGHUP ignorance is mimic of csh. but current almost modern shell nor other scripting language (e.g. perl) don't have such feature. and then, any userland programs don't depend on that a parent process ignore SIGHUP.

Do you have any usecase of SIGHUP? If you have real world usecause, you might be able to persuade us.

Thank you.

#3 - 11/10/2011 02:20 AM - kosaki (Motohiro KOSAKI)

- Status changed from Assigned to Feedback

#4 - 11/10/2011 02:21 AM - kosaki (Motohiro KOSAKI)

In addition, signal handler is per-process resource. iow, changing sighandler is thread unsafe. then, we don't want unnecessary sighandler change. that's why I ask you usecase.

#5 - 11/10/2011 03:27 AM - brixen (Brian Shirai)

The use case is any reason someone would send a signal to the process group but not want the parent process to abort. I'm quite sure there is existing code that depends on this behavior even by accident. The fact that it changed will affect some programs that have previously not ignored HUP, so people will likely get puzzling process exits in cases such as I discovered here by accident, ie on process subprocessing worker processes.

I really don't care what the behavior is as long as it is documented and tested. If you can point me to tests for this, I'll adapt them for RubySpec.

I have no idea what you mean by "signal handler is per-process resource", how would it not be a per-process resource? Or do you have a definition of per-process where process is not an OS process? How does per-process and thread-safety relate? Seems like we should have a glossary of terms as MRI defines them to avoid some confusion here.

#6 - 11/10/2011 11:23 AM - kosaki (Motohiro KOSAKI)

- ruby -v changed from ruby 2.0.0dev (2011-11-01 trunk 33596) [x86_64-darwin10.8.0] to -

> The use case is any reason someone would send a signal to the process group but not want the parent process to abort. I'm quite sure there is existing code that depends on this behavior even by accident. The fact that it changed will affect some programs that have previously not ignored HUP, so people will likely get puzzling process exits in cases such as I discovered here by accident, ie on process subprocessing worker processes.

Guys, I asked you practical and real world usecase.

> I really don't care what the behavior is as long as it is documented and tested. If you can point me to tests for this, I'll adapt them for RubySpec.
>
> I have no idea what you mean by "signal handler is per-process resource", how would it not be a per-process resource? Or do you have a definition of per-process where process is not an OS process? How does per-process and thread-safety relate? Seems like we should have a glossary of terms as MRI defines them to avoid some confusion here.

man sigaction.

#7 - 11/12/2011 09:47 AM - brixen (Brian Shirai)

Apparently, I don't have a "practical and real world usecase" as a change in the behavior of running RubySpec doesn't qualify.

This change will potentially affect any program that subprocesses workers. If that's fine with you, great.

I'm simply asking for the change to be documented with tests. Do you have those?

> man sigaction.

That is unhelpful. I know what sigaction does. What is your definition of a "process" and why is changing the sighandler not thread-safe?

#8 - 12/03/2012 12:02 AM - kosaki (Motohiro KOSAKI)

- Assignee deleted (kosaki (Motohiro KOSAKI))

#9 - 02/17/2013 01:53 PM - ko1 (Koichi Sasada)

- Assignee set to mame (Yusuke Endoh)

#10 - 02/17/2013 02:48 PM - mame (Yusuke Endoh)

- Assignee changed from mame (Yusuke Endoh) to kosaki (Motohiro KOSAKI)

I don't think that this issue is ready for determining a backport to 2.0.0.
Kosaki-san, what do you think?

--
Yusuke Endoh mame@tsg.ne.jp

#11 - 02/17/2013 02:58 PM - kosaki (Motohiro KOSAKI)

- Status changed from Feedback to Closed

No feedback. Close it.