Ruby - Bug #14196

Signal.trap overrides pre-existing "SIG_IGN" handler on a process

12/18/2017 05:26 AM - shayonj (Shayon Mukherjee)

Status:	Third Party's Issue		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.3.3p222 (2016-11-21 revision 56859) [x86_64-darwin16]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

Came across while debugging an issue with bundler https://github.com/bundler/bundler/issues/6150

The issue is, if a process already has a SIG_IGN / IGNORE present on it, then doing a Signal.trap overrides the handler on the same process. I believe, if a process already has a SIG_IGN then the same should be respected/restore.

This is how I am able to replicate the issue (from a linux machine where /proc is mounted).

Sample ruby file for testing (**test.rb**)

```
p Signal.trap("SIGHUP", "IGNORE")
p Signal.trap("SIGHUP", "SYSTEM_DEFAULT")
# This returns the correct string, `IGNORE`, but does not update the signal handler.

sleep 100
```

Here we are replicating a scenario where a process first gets a trap from SIG_IGN and then a SYSTEM_DEFAULT / DEFAULT.
Next, run it

```
ruby test.rb 2>&2 &

[1] 23156
```

Now, lets see what signals are reserved on this status:

```
cat /proc/23156/status
Name: ruby
Umask: 0022
State: S (sleeping)
Tgid: 23156
Ngid: 0
Pid: 23156
PPid: 52
TracerPid: 0
Uid: 0 0 0 0
Gid: 0 0 0 0
FDSize: 256
Groups:
NStgid: 23156
NSpid: 23156
NSpgid: 23156
NSsid: 52
VmPeak:    40508 kB
VmSize:    40508 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:     8036 kB
VmRSS:     8036 kB
RssAnon:      3596 kB
RssFile:      4440 kB
RssShmem:        0 kB
VmData:     5192 kB
```

```
VmStk:     8188 kB
VmExe:        4 kB
VmLib:     6624 kB
VmPTE:       88 kB
VmPMD:       12 kB
VmSwap:       0 kB
HugetlbPages:        0 kB
Threads: 2
SigQ: 0/7753
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 00000001c2007e4e
CapInh: 00000000a80425fb
CapPrm: 00000000a80425fb
CapEff: 00000000a80425fb
CapBnd: 00000000a80425fb
CapAmb: 0000000000000000
Seccomp: 2
Cpus_allowed: f
Cpus_allowed_list: 0-3
Mems_allowed: 1
Mems_allowed_list: 0
voluntary_ctxt_switches: 1
nonvoluntary_ctxt_switches: 8
```

You'll notice that SigIgn is 0000000000000000 which a bitmask and is indicating no IGNORE handlers on the process is reserved.

I have a fix/patch with specs, which I will share soon on Github for further insight. I would love to learn thoughts on this :).

History

#1 - 12/18/2017 05:28 AM - shayonj (Shayon Mukherjee)

- Description updated

#2 - 12/18/2017 05:53 AM - shyouhei (Shyouhei Urabe)

shayonj (Shayon Mukherjee) wrote:

> I believe, if a process already has a SIG_IGN then the same should be respected/restore.

I don't think so. Signal handlers are per-prorcess global resources. If a program decides to set one, that should be honored, not ignored only because its previous state is SIG_IGN. This is how a signal handler works, AFAIK as in other languages like C.

PS. The return value of Signal.trap is what was previously set, not the current handler. So when it returns "IGNORE", that means the previous handler was it, not now.

#3 - 12/18/2017 06:08 AM - shayonj (Shayon Mukherjee)

shyouhei (Shyouhei Urabe) wrote:

> shayonj (Shayon Mukherjee) wrote:
>
> > I believe, if a process already has a SIG_IGN then the same should be respected/restore.
>
> I don't think so. Signal handlers are per-prorcess global resources. If a program decides to set one, that should be honored, not ignored only because its previous state is SIG_IGN. This is how a signal handler works, AFAIK as in other languages like C.
>
> PS. The return value of Signal.trap is what was previously set, not the current handler. So when it returns "IGNORE", that means the previous handler was it, not now.

Thanks! I realize that Signal.trap returns the previously set handler, though I didn't realize, we don't wish to preserve the previously set handler. I guess it makes sense for a program to handle this case-by-case basis.

Feel free to close it out, if you think this is not actionable :).

#4 - 12/18/2017 06:15 AM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Third Party's Issue

Closing. Thanks anyway!