Ruby master - Feature #15017

Provide extended information about Signal

08/21/2018 11:26 AM - jreidinger (Josef Reidinger)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Hi,
I see that ruby already use sigaction for signal handling on linux. It would be really nice to extend it to provide also signal details in siginfo_t and then provide such details in ruby via Signal module (as additional param beside signal number for block). Use case is quite simple. Our ruby application is killed by sigint and we do not know who send this signal. We already catching signal and logging as much as possible, but without siginfo we are very limited. We do not know ff it is OOMKiller due to low memory, systemd or something else. This additional info in siginfo allows us to log a much more details when such signal appear and inspect process that send us this signal. This is useful especially on customer side where we do not have direct control and we get only logs from failed run.

If you need more info or help with example usage, do not hesitate to ask.

Thanks
Josef

History

#1 - 08/21/2018 05:02 PM - shevegen (Robert A. Heiler)

Some comments (personal opinion here):

1) I think the described use case is understandable, but if you can, I think it may
help matz and the ruby core team decide on; for example, what is the specific API
you had in mind? Could you give a more explicit code example here? You mentioned it
here already: "additional param beside signal number for block" but I think it may
help if you could give a specific example in ruby code, how it looks and perhaps
an example sentence for documentation (may help whoever will write the code on the
C level side, I think, and also provide an example to the on-line documentation
for ruby+signal).

2) I assume you use Linux, like many folks here do; matz uses debian as far as I
know. I myself use mostly slackware, but compile literally almost everything from
source using ruby scripts. Many other people use Windows, OSX/Mac variant or even
less common operating systems (Solaris, the various BSD flavours, HaikuOS and so
forth). There are examples where features have been rejected because they may
only run on some systems but not others. I do not know whether this is the case
in your example; perhaps there are agnostic implementations possible that would
work on every platform. That would be best.

I mention this in particular because you gave the example of systemd. And I
believe if functionality would be only limited to one OS, then it may be better
to have it available as a gem. Note that even in this case, I am sure the ruby
core team and others may be able to help here, e. g. what may be required to
offer the desired functionality.

I would suggest to you to provide a bit more context, if you feel like it; for
example perhaps you may have an idea how to best implement it too, but probably
describing your use case is the single best thing to do; matz and the core team
often said that they prioritize on real problems that ruby users have and the
corresponding use cases (although you may already have described it enough, I
just mention it again because I think it is one of the best ways to get a
change into ruby if you have a valid use case - there are various examples in
the past where this has helped a lot).

And consider adding it for discussion at the next developer meeting:

https://bugs.ruby-lang.org/issues/14981

Matz is very busy, coming to europe soon too \o/ so the developer meetings
are great to help "get the process going". But of course it is your issue,

please feel free to do whatever you want to do. :)

#2 - 08/22/2018 02:57 AM - nobu (Nobuyoshi Nakada)

shevegen (Robert A. Heiler) wrote:

> Some comments (personal opinion here):
>
> 1) I think the described use case is understandable, but if you can, I think it may
> help matz and the ruby core team decide on; for example, what is the specific API
> you had in mind? Could you give a more explicit code example here? You mentioned it
> here already: "additional param beside signal number for block" but I think it may
> help if you could give a specific example in ruby code, how it looks and perhaps
> an example sentence for documentation (may help whoever will write the code on the
> C level side, I think, and also provide an example to the on-line documentation
> for ruby+signal).

Currently, it is:

```
Signal.trap("INT") {|signo| }
```

where signo is an integer.

What I can think of is:

```
Signal.trap("INT") {|siginfo| }
```

where siginfo is an object which has #to_int method to return the signal number,
and other accessors.

Another idea is:

```
Signal.trap("INT") {signo, siginfo| }
```

but this can cause compatibility issues on the block arity.

> 2) I assume you use Linux, like many folks here do; matz uses debian as far as I
> know. I myself use mostly slackware, but compile literally almost everything from
> source using ruby scripts. Many other people use Windows, OSX/Mac variant or even
> less common operating systems (Solaris, the various BSD flavours, HaikuOS and so
> forth). There are examples where features have been rejected because they may
> only run on some systems but not others. I do not know whether this is the case
> in your example; perhaps there are agnostic implementations possible that would
> work on every platform. That would be best.

As far as rubyci.org, UNIX-like platforms have sigaction all, including AIX, FreeBSD, Solaris and macOS.

#3 - 08/22/2018 05:42 AM - normalperson (Eric Wong)

> Feature #15017: Provide extended information about Signal
> https://bugs.ruby-lang.org/issues/15017#change-73659

Implementation would be tricky, since we defer signals to run
Ruby code. (Deferring allows us to use malloc and other
non-async-safe C stdlib functions).

To implement, we'd have to reject Misc #15011[1] and
write "struct siginfo" to the pipe.

Or, write a async-signal-safe bump allocator...

[1] Too many full-sized pipes cause resource problems requiring ugly
workarounds like r64478.