The next dev meeting

Date: 2021/08/19 13:00-17:00
Place/Sign-up/Agenda/Log: https://github.com/ruby/dev-meeting-log/blob/master/DevelopersMeeting20210819Japan.md

- Dev meeting IS NOT a decision-making place. All decisions should be done at the bug tracker.
- Dev meeting is a place we can ask Matz, nobu, nurse and other developers directly.
- Matz is a very busy person. Take this opportunity to ask him. If you can not attend, other attendees can ask instead of you (if attendees can understand your issue).
- We will write a log about the discussion to a file or to each ticket in English.
- All activities are best-effort (keep in mind that most of us are volunteer developers).
- The date, time and place are scheduled according to when/where we can reserve Matz's time.
- DO NOT discuss then on this ticket, please.

Call for agenda items

If you have a ticket that you want matz and committers to discuss, please post it into this ticket in the following format:

* [Ticket ref] Ticket title (your name)
  * Comment (A summary of the ticket, why you put this ticket here, what point should be discussed, etc.)

Example:

* [Feature #14609] `Kernel#p` without args shows the receiver (ko1)
  * I feel this feature is very useful and some people say :+1: so let discuss this feature.

- It is recommended to add a comment by 2021/08/16. We hold a preparatory meeting to create an agenda a few days before the dev-meeting.
- The format is strict. We'll use this script to automatically create a markdown-style agenda. We may ignore a comment that does not follow the format.
- Your comment is mandatory. We cannot read all discussion of the ticket in a limited time. We appreciate it if you could write a short summary and update from a previous discussion.

Related issues:

<table>
<thead>
<tr>
<th>Related to Ruby master - Misc #14770: [META] DevelopersMeeting</th>
<th>Open</th>
</tr>
</thead>
</table>

History

#1 - 07/16/2021 06:25 AM - mame (Yusuke Endoh)
- Related to Misc #14770: [META] DevelopersMeeting added

#2 - 08/03/2021 12:41 AM - nobu (Nobuyoshi Nakada)

- [Misc #18059] Which FL_USERx are open to extension libraries? (nobu)
  - (need comment)

#3 - 08/03/2021 01:48 AM - nobu (Nobuyoshi Nakada)

- [Feature #18051] Move symbols exported under internal (nobu)
  - Some symbols exported in headers under internal are declared and used in some extension libraries. These seem OK to be moved to
public headers.
- ruby_scan_digits, ruby_hexdigits, ruby_digit36_to_number_table
- rb_int_positive_pow
- rb_deprecate_constant

#4 - 08/03/2021 09:48 PM - alanwu (Alan Wu)

- [Bug #13392] TracePoint return event location is incorrect for methods defined with define_method (alanwu)
  - There are two PRs available, one more complex but opens the possibility for future optimizations. Which do you prefer?
- [Bug #18060] Fix infinite loop when b_return TracePoint throws (alanwu)
  - Does the fix make sense?
- [Bug #18031] Nested TracePoint#enable with define_method target crashes (alanwu)
  - Does my fix look good? It assumes that multiple TracePoints targeting the same bmethod should work similar to other targets.

#5 - 08/05/2021 12:33 AM - duerst (Martin Dürst)

- [Feature #18045] Variable Width Allocation Phase II (peterzhu2118)
  - Variable Width Allocation is a major change to Ruby internals and therefore should be discussed seriously.

#6 - 08/06/2021 11:20 AM - byroot (Jean Boussier)

- [Feature #10917] Add GC.stat :total_time
  - Time spent in GC is a very important metric for application performance monitoring.
  - Right now only GC::Profiler can be used, but it is ill suited when there are multiple callers because it is stateful. It also need to be gated by a Mutex.
  - A single monotonically increasing time metric would be much more usable.
  - If the getusage_time overhead is dimmed to big, it could be a an opt-in metrics through a flag on ::GC.
  - I don't ask to merge that particular patch, but wether the general idea is acceptable.
  - Eidgen (Benoit Daloze) suggested GC.stat[:time] in milliseconds, as that's what JRuby and TruffleRuby already expose.

#7 - 08/09/2021 07:56 PM - jeremyevans0 (Jeremy Evans)

- [Bug #18036] Pthread fibers become invalid on fork - different from normal fibers (jeremyevans0)
  - fork terminates other threads.
  - When using pthread coroutine, all fibers are threads, therefore fork terminates all other fibers.
  - Are we OK with this runtime difference with pthread coroutine? (I am.)
  - Do we want to terminate other fibers when forking always, or just when using the pthread coroutine?
  - Do we want to raise an exception if calling fork from a non-main fiber?
- [Feature #16889] TracePoint.enable { ... } also activates the TracePoint for other threads, even outside the block (jeremyevans0)
  - I think this behavior is currently expected. At least the tests expect it, and the behavior seems reasonable.
  - Ruby could benefit from a nicer way to pass a block and only allow tracing for the block.
  - I implemented this nicer way by supporting a target: :block option.
  - Is the current behavior of TracePoint.enable with a block expected?
  - Is it OK to add support for a target: :block option using the patch?
- [Bug #15404] Endless range has inconsistent chaining behaviour (jeremyevans0)
  - Do we want to make ranges of ranges a parse error (e.g. (1..2)..(1..3))?
  - I think we shouldn't. It's possible to define Range#<< and Range#succ such that a range of ranges could be useful.
- [Bug #15428] Refactor Proc#>> and #<< (jeremyevans0)
  - Do we want Proc#<< and Proc#>> to call to_proc on the argument given?
  - Doing so would allow usage of symbols and other things that implement to_proc but not call.
  - Doing so always would break usage with arguments that respond to call but do not respond to to_proc.
  - Doing so only if the argument responds to to_proc and not call may result in inconsistent behavior.
  - I see no problem with the current situation, but I'm not opposed to calling to_proc if the object does not respond to call.
- [Bug #14744] Refinements modules have a superclass (jeremyevans0)
  - Should we hide the superclass of a refinement module from reflection APIs such as ancestors?

#8 - 08/15/2021 03:08 AM - ioquatix (Samuel Williams)

- [Feature #16182] Should expr in a, b, c be allowed or not? (ktst)
  - How about allowing brackets/braces to be omitted in one-line pattern matching?

#9 - 08/16/2021 07:35 PM - tenderlovemaking (Aaron Patterson)

- [Feature #18007] C extension "allocation function" check
  - Should we add a config option and put in 3.1, or just wait for 3.2?

#10 - 08/18/2021 03:08 AM - ioquatix (Samuel Williams)
[Feature #18083] Capture error in ensure block.
  - Is it acceptable?

#11 - 08/19/2021 04:35 AM - nobu (Nobuyoshi Nakada)
  - [Bug #18077] Marshal.dump(closed_io) raises IOError instead of TypeError (nobu)
    - Marshal.dump is expected to raise a TypeError for unmarshallable objects. But closed streams raise an IOError

#12 - 08/20/2021 05:14 AM - mame (Yusuke Endoh)
  - Description updated

#13 - 11/24/2021 04:59 AM - hsbt (Hiroshi SHIBATA)
  - Status changed from Open to Closed