Deprecate `Random::DEFAULT` and introduce `Random.default()` method to provide Ractor-supported default random generator

11/12/2020 01:48 AM - ko1 (Koichi Sasada)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td></td>
</tr>
</tbody>
</table>

**Description**

`Random::DEFAULT` a default random generator used by `rand`, `srand`, `Array#shuffle` without a given random generator, and so on.

Random generators are not thread-safe, so they are not ractor safe, and they are not shareable. So a program refer to `Random::DEFAULT` on non-main ractor, it causes an error.

To provide per-ractor default random generator, this ticket propose the `Random.default()` method which returns per-ractor random generator. `Random::DEFAULT` is a result of `Random.default()` on main-ractor and it should be deprecated, or at least it should not be used on multi-ractor supporting apps and libraries.

**Related issues:**

- Related to Ruby master - Feature #17351: Deprecate Random::DEFAULT - Closed

**Associated revisions**

Revision 2db2fb9f - 11/27/2020 08:03 AM - ko1 (Koichi Sasada)

per-ractor Random::DEFAULT

Random generators are not Ractor-safe, so we need to prepare per-ractor default random generators. This patch set Random::DEFAULT = Randm (not a Random instance, but the Random class) and singleton methods like Random.rand() use a per-ractor random generator.

[Feature #17322]

**History**

#1 - 11/12/2020 01:58 AM - shyouhei (Shyouhei Urabe)

Why not introduce general-purpose per-ractor variables first, instead of magical method that interacts with the current ractor behind-the-scene.

#2 - 11/12/2020 02:15 AM - ko1 (Koichi Sasada)

Why not introduce general-purpose per-ractor variables first, instead of magical method that interacts with the current ractor behind-the-scene.

Do you mean new syntax or method?

Just now I'm writing the per-ractor storage class (not a syntax). However, it needs new constant name and method access, such as Random::DEFAULT2.value, and Random.default seems better, so I filed this ticket first.

#3 - 11/12/2020 02:24 AM - shyouhei (Shyouhei Urabe)

ko1 (Koichi Sasada) wrote in #note-2:

Why not introduce general-purpose per-ractor variables first, instead of magical method that interacts with the current ractor behind-the-scene.

Do you mean new syntax or method?

Both :(
But I understand that new syntax needs time (or maybe it's matz who needs time). A method would be a good starting point. Maybe Ractor#[] (like Thread#[])?

#4 - 11/12/2020 02:26 AM - ko1 (Koichi Sasada)

Maybe Ractor#?

yes, it is one option (but I don't like this, make a new ticket soon about it with reasons).

However, if we choose Ractor[], Ractor.current[:DEFAULT_RANDOM] seems not good idea (not impossible).

#5 - 11/12/2020 02:39 AM - shyouhei (Shyouhei Urabe)

It sounds rather intuitive to me that rand etc. are not sharable among ractors. It is their nature to mutate process-global state of executions.

If a programmer wants randomness inside of a ractor they needs special care to separate their random source from the rest of the world. This means API changes are inevitable. I'm +1 to the deprecation part.

#6 - 11/12/2020 09:31 PM - Eregon (Benoit Daloze)

In JRuby and in TruffleRuby, Random instances are thread-safe (i.e., they use synchronization internally). Also, rand, etc, use a per-Thread Random instance to avoid needless contention when calling rand in multiple threads concurrently.

So, I agree we should deprecate Random::DEFAULT.

I think rand, etc should use a thread-local Random instance (and thread-local implies Ractor-local, of course). Those thread-local Random instances should probably not be exposed to the Ruby level, so that way there is never a need to synchronize access to it.

So, I think we should deprecate Random::DEFAULT without replacement.

Is there any case where it's useful?

#7 - 11/13/2020 07:13 PM - Dan0042 (Daniel DeLorme)

Eregon (Benoit Daloze) wrote in #note-6:

I think rand, etc should use a thread-local Random instance (and thread-local implies Ractor-local, of course). Those thread-local Random instances should probably not be exposed to the Ruby level, so that way there is never a need to synchronize access to it.

Bypassing the need for synchronization is a very good idea.

Is there any case where it's useful?

Good question. Found this. Doesn't look so useful.

```ruby
faker-2.2.2/lib/faker.rb
35: @random ||= Random::DEFAULT
```

```ruby
sentry-raven-2.11.1/lib/raven/configuration.rb
445: if Random::DEFAULT.rand >= sample_rate
```

That said, even though using Random::DEFAULT seems like an anti-pattern, there's no real benefit to removing it. Maybe just convert it to an instance not used by anything? (except gems/apps)

#8 - 11/25/2020 07:32 PM - ko1 (Koichi Sasada)

At the last dev-meeting, there is an idea to replace Random::DEFAULT with an object which delegates operations to the per-ractor random generator. Fortunately, Random class object can be a good placeholder because it already has Random.rand, Random.bytes and so on.

```ruby
Random::DEFAULT = Random
p Random::DEFAULT.rand(2) #=> 0 or 1
```

The specification changes are:

- Random::DEFAULT isn't Random instance, but Random class object.
- Random.rand(), Random.bytes() and so on use per-ractor random generator. Maybe this is implementation dependent. JRuby and so on can use thread-local random generators.
- rand(), srand() use Random.rand(), Random.srand() respectively (but it doesn't trace Random::DEFAULT replacement as current implementation doesn't)
Random.seed() is added to return current seed of per-ractor default random generator.

Advantages:

- We don't need to change most of code.
- We can introduce per-ractor (per-thread and so on) semantics naturally.

Disadvantages => It can break the compatibility because Random::DEFAULT is a Class instance.

- can not save the state of Random::DEFAULT
  - We can not use dup for it (there is a test using Random::DEFAULT.dup in test/ruby/test_random.rb).
  - We can not use marshal protocol to save the random generator states of Random::DEFAULT.
  - I'm not sure how it is important.
- can not use instance methods of Random class
  - If user defines Random#foo, but Random::DEFAULT.foo is not available.

I checked Random::DEFAULT usage with gem-codesearch and most of case it uses Random::DEFAULT.rand() and so on.

I can't trace how to use Random::DEFAULT for the following 4 cases.

```
/srv/gems/kmat-0.0.3/lib/kmat/random.rb: Random::DEFAULT.randn(*args)
/srv/gems/kmat-0.0.3/lib/kmat/random.rb: Random::DEFAULT.randn(*args)
# I can't find the 'Random#randn' definition.
/srv/gems/prop_check-0.14.1/lib/prop_check/property.rb: rng = Random::DEFAULT
# I can't find how to use 'rng'
/srv/gems/util-0.4.0/lib/util/args.rb: Random => Random::DEFAULT,
# I can't find how to use this information.
```

This is implementation: https://github.com/ruby/ruby/pull/3813

#9 - 11/26/2020 05:56 AM - matz (Yukihiro Matsumoto)
I vote for replacing Random::DEFAULT with ractor-safe object (Random class). We don't need Random.default() then.

Matz.

#10 - 11/26/2020 02:53 PM - Eregon (Benoit Daloze)

Sounds great, except:

ko1 (Koichi Sasada) wrote in #note-8:

- Random.seed() is added to return current seed of per-ractor default random generator.

Is this needed? It sounds very bad practice from a security point of view to ever be able to read the seed directly
(without also changing it at the same time like srand).

#11 - 11/26/2020 03:14 PM - Eregon (Benoit Daloze)

I see, it's to be compatible for Random::DEFAULT.seed, unfortunate.

I think we should deprecate the Random::DEFAULT constant, it doesn't make sense anymore and it's longer than using methods on Random or Kernel directly.

Also, people might expect it to be global.

#12 - 11/26/2020 10:26 PM - shyouhei (Shyouhei Urabe)

Despite the title of this issue, ko1's intention is to make Ractor usable, not to make Random usable.

I think there still is a room to depreciate Random::DEFAULT. But if you want that I guess you can have a separate ticket. I guess that should be smoother than discussing it here.

#13 - 11/27/2020 02:37 AM - nobu (Nobuyoshi Nakada)

Eregon (Benoit Daloze) wrote in #note-10:

Is this needed? It sounds very bad practice from a security point of view to ever be able to read the seed directly
(without also changing it at the same time like srand).
As you **must not** use the default RNG for security purpose, it doesn't matter.

#14 - 11/27/2020 08:03 AM - ko1 (Koichi Sasada)
- Status changed from Open to Closed

Applied in changeset a122b2f2f96c742d5bd0019ccd11c7a375eb12b0b.

per-ractor Random::DEFAULT

Random generators are not Ractor-safe, so we need to prepare per-ractor default random generators. This patch set Random::DEFAULT = Random (not a Random instance, but the Random class) and singleton methods like Random.rand() use a per-ractor random generator.

[Feature #17322]

#15 - 11/27/2020 11:16 AM - Eregon (Benoit Daloze)
- Related to Feature #17351: Deprecate Random::DEFAULT added