Make Pathname to embedded class of Ruby

pathname is one of most useful utility class of Ruby. I'm happy to use Pathname without require it.

Any thought?

History

#1 - 12/26/2020 12:45 PM - Eregon (Benoit Daloze)
If we do so, could we actually define most of Pathname in Ruby, and not in C?
Right now, https://github.com/ruby/ruby/blob/3fc53de5c961cc8fa2b6acbd63874b69fe709520/ext/pathname/pathname.c is essentially just a bunch of rb_funcall() calls, which are no faster than Ruby code, but makes it harder to reuse on other implementations, and less readable and harder to maintain.

#2 - 12/27/2020 10:57 AM - mame (Yusuke Endoh)
- Assignee set to akr (Akira Tanaka)
The proposal lacks one background, so I'd like to add it. Rubygems cannot allow users to choose the version of a gem that rubygems itself are using. So, we want to make Rubygems independent with any gems. According to hsbt (Hiroshi SHIBATA), Rubygems is using Pathname, FileUtils, Tsort, etc. Though FileUtils and Tsort are relatively easy to be removed from the dependencies, Pathname looks difficult.
In addition, of course, Pathname is widely used, and the API of Pathname looks stable. So I also think it deserves built-in.

#3 - 12/27/2020 04:16 PM - Eregon (Benoit Daloze)
I forgot to mention, +1 from me for Pathname in core, if it's written mostly in Ruby.

#4 - 12/27/2020 07:19 PM - Dan0042 (Daniel DeLorme)
mame (Yusuke Endoh) wrote in #note-2:
Though FileUtils and Tsort are relatively easy to be removed from the dependencies, Pathname looks difficult.
Isn't Pathname dependent on FileUtils though?

#5 - 12/28/2020 12:30 AM - mame (Yusuke Endoh)
Dan0042 (Daniel DeLorme) wrote in #note-4:
 Isn't Pathname dependent on FileUtils though?
Yes. It uses only FileUtils.mkpath and rm_r. Rubygems also uses them, so hsbt (Hiroshi SHIBATA) is preparing to propose making a very limited set of FileUtils methods built-in, too.

#6 - 12/28/2020 12:38 PM - Eregon (Benoit Daloze)
I thought about FileUtils too, but it's required lazily for these two methods. So I think it might be fine to move only Pathname to core, and accept that Pathname#(mkpath, rmtree) requires fileutils when used.
I think having part of FileUtils defined in core would be confusing.

#7 - 12/30/2020 03:05 PM - shyouhei (Shyouhei Urabe)
JFYI pathname was born as a pure-ruby library, then was eventually translated into C (in 4bf3cb5ba9c0242bd5a6d556b7d89f837c09edf). Don't know the reason behind that move though. akr (Akira Tanaka) do you remember?
JFYI pathname was born as a pure-ruby library, then was eventually translated into C (in 4bf3cb5ba9c0242bd5a6d0d55b7db9f837c09edf). Don't know the reason behind that move though. akr (Akira Tanaka) do you remember?

I know, I'd like to undo that change (I can make a PR), except for the parts which are significantly faster in C (I think only Pathname#<=> / path_cmp). Then, we could share more of Pathname between Ruby implementations, and avoid maintaining both a Ruby and C version. There is a copy in TruffleRuby, in JRuby and in Rubinius right now due to that commit :/

akr (Akira Tanaka) Would that be OK?

Then, we could share more of Pathname between Ruby implementations, and avoid maintaining both a Ruby and C version.

I'm not sure how embedded pure-Ruby implementation to core classes. The above request is the different request.

I support making Pathname part of core Ruby. It's an extremely convenient library that I use very often.

Just for the record, String#unicode_normalize is part of a core class but is implemented in pure Ruby. It's a single case, and includes loading code on demand. If we do more pure-Ruby implementations for core classes, we should think again about the best way to do it.

It's also related, if the decision was to move all of Pathname to C (due to some reason of moving it to core), I would be strongly against it. I think concretely we can just have pathname.rb at the root, similar to e.g. dir.rb.

I had a plan that Pathname contains a FD optionally as root directory and use openat() and other "at system call.
No progress after translation to C, though.