Ruby - Feature #20152

mkmf / extconf: Add a proper way to not compile the extension

01/05/2024 10:53 AM - byroot (Jean Boussier)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Context

There are various gems that ship with a native extension as a way to speedup part of the gem, but also ship with a pure Ruby version of these methods as a fallback. So they only want to compile the extension if the platform supports it, and if not, just fallback to the slightly slower Ruby version.

Right now users rely on one of two hacks to do this. Either they create an empty Makefile, but then still depend on make being available, or publish platform specific packages without any extension in them.

Examples:

- bootsnap skip compilation if not on MRI or TruffleRuby
- erb has an extension for MRI but then need to publish a java version of the gem that doesn't actually contain Java code, just to skip compilation on JRuby
- hiredis-client skips the compilation for Windows and non-MRI rubies

Feature

It would be very useful to have some proper first class API to skip compiling the extension.

Something like:

```
require "mkmf"

if RUBY_ENGINE != "ruby" || RUBY_PLATFORM.match?(/mswin/)
  skip_compilation
else
  # ...
end
```

cc @k0kubun (Takashi Kokubun) @headius (Charles Nutter)

History

#1 - 01/05/2024 12:58 PM - nobu (Nobuyoshi Nakada)

byroot (Jean Boussier) wrote:

> It would be very useful to have some proper first class API to skip compiling the extension.
>
> Something like:
>
> ```
> require "mkmf"
>
> if RUBY_ENGINE != "ruby" || RUBY_PLATFORM.match?(/mswin/)
>   skip_compilation
> else
>   # ...
> end
> ```

Where will this be used?

#2 - 01/05/2024 01:29 PM - byroot (Jean Boussier)

> Where will this be used?

I'm not sure I understand the question.

This would be called in extconf.rb

#3 - 01/05/2024 04:03 PM - byroot (Jean Boussier)

In case you meant which gems would use this, then I think the 3 I listed as example would make a good use of it, and I'm certain there are some more that are in similar cases.

#4 - 01/05/2024 04:06 PM - kddnewton (Kevin Newton)

Prism would use this as well: https://github.com/ruby/prism/blob/829ac0ed3f449313584aae35db98fd7614eb9d63/ext/prism/extconf.rb#L47-L56

#5 - 01/06/2024 07:43 AM - nobu (Nobuyoshi Nakada)

byroot (Jean Boussier) wrote in #note-2:

> This would be called in extconf.rb

Does skip_compilation just make a Makefile does nothing?
Then it won't be able to fix the issue that make is needed, since make is called outside extconf.rb.

#6 - 01/06/2024 08:50 AM - byroot (Jean Boussier)

> Does skip_compilation just make a Makefile does nothing?

I don't know, I'm unfamiliar with the internals of mkmf and not well versed in all this tooling.

I'm just requesting the capability of skipping compilation from extconf.rb, not prescribing an implementation nor a specific API.

If you want a specific implementation plan, I can look into it.

#7 - 01/06/2024 12:00 PM - Eregon (Benoit Daloze)

This is not something mkmf can do on its own, it would need changes in RubyGems, and it seems not so easy for RubyGems to know.

When one uses spec.extensions = ["ext/mygem/extconf.rb"] in a .gemspec file, RubyGems will:

1. Run ruby -rmkmf ext/mygem/extconf.rb.
2. Run make in ext/mygem.

Maybe RubyGems could recognize dummy Makefiles like the one created with File.write("Makefile", dummy_makefile($srcdir).join("")) which is the most common pattern?
A bit hacky, but it would address the problem with very few changes.

File.write("Makefile", dummy_makefile($srcdir).join("")) is already used in many gems:
https://github.com/search?q=language%3ARuby+dummy_makefile%28%24srcdir%29.join&type=code
But a shorter variant could be nice, e.g. create_dummy_makefile that does the same as File.write("Makefile", dummy_makefile($srcdir).join("")).

OTOH, IMO it's not a big ask for devs/users to install make when they are installing gems.
If devs/users really want to cut dependencies in production they can and should install gems in a separate Docker image than the runtime one (already necessary e.g. for CRuby & TruffleRuby if ones wants to not ship a cc in the final image, example).
Also one might need make anyway, e.g. for gems used via FFI like sassc and prism, where make is used to build the C code (which does not include ruby.h).
So trying to avoid needing make on JRuby seems very niche to me.

#8 - 01/06/2024 01:51 PM - byroot (Jean Boussier)

> This is not something mkmf can do on its own, it would need changes in RubyGems

Thanks for the info, as I said, I'm quite unfamiliar with how these pieces fit together.

> Maybe RubyGems could recognize dummy Makefiles

Or look for a specially named file? This way mkmf could just create that file and move on?

> OTOH, IMO it's not a big ask for devs/users to install make when they are installing gems.

On UNIX systems it's not, but on Windows it start to be a bit more complicated.

> So trying to avoid needing make on JRuby seems very niche to me.

I honestly don't mind it that much, but it's true that if the end the Makefile does nothing, we might as well not require make, would be nicer.

#9 - 01/06/2024 09:38 PM - kou (Kouhei Sutou)

byroot (Jean Boussier) wrote in #note-8:

> > OTOH, IMO it's not a big ask for devs/users to install make when they are installing gems.
>
> On UNIX systems it's not, but on Windows it start to be a bit more complicated.

Really? https://rubyinstaller.org/ provides MSYS2 integration by "Devkit". It provides make.

#10 - 01/07/2024 12:10 AM - jeremyevans0 (Jeremy Evans)

nobu (Nobuyoshi Nakada) wrote in #note-5:

> byroot (Jean Boussier) wrote in #note-2:
>
> > This would be called in extconf.rb
>
> Does skip_compilation just make a Makefile does nothing?
> Then it won't be able to fix the issue that make is needed, since make is called outside extconf.rb.

I agree that this isn't a mkmf/extconf.rb issue. This is a rubygems issue. I think the easiest way to fix this would be for rubygems to check for a skip-compilation-RUBY_ENGINE file in the same directory as extconf.rb. If it exists, do not run extconf.rb or make. If it exists after calling extconf.rb, do not call make. This fixes cases where an extension is not needed for a certain engine, as well as cases where an extension is not needed in certain configurations, such as when a pure-ruby implementation is also shipped in the same gem, and a native library necessary for the native extension is not available.

#11 - 01/08/2024 01:30 PM - Eregon (Benoit Daloze)

jeremyevans0 (Jeremy Evans) wrote in #note-10:

> I agree that this isn't a mkmf/extconf.rb issue. This is a rubygems issue. I think the easiest way to fix this would be for rubygems to check for a skip-compilation-RUBY_ENGINE file in the same directory as extconf.rb. If it exists, do not run extconf.rb or make. If it exists after calling extconf.rb, do not call make. This fixes cases where an extension is not needed for a certain engine, as well as cases where an extension is not needed in certain configurations, such as when a pure-ruby implementation is also shipped in the same gem, and a native library necessary for the native extension is not available.

I think skip-compilation-RUBY_ENGINE is going too far and counter-productive.
For instance I have seen some C extensions in gems which realistically will only work on CRuby (e.g. depends on CRuby deep internals, on CRuby bytecode, optimization only profitable on CRuby, etc), in that case we'd want to avoid the extension on any RUBY_ENGINE != "ruby", so a skip-compilation-RUBY_ENGINE would not help.

There could be a skip-make/skip-compilation or so file which does not include RUBY_ENGINE in the filename and is created dynamically by the extconf.rb, but this will take a lot longer to get adopted, as it means each gem skipping compilation would need to use that.
But it is error-prone, because e.g. then if one builds (e.g. rake compile) a gem locally on JRuby and then on CRuby then compilation will be unexpectedly skipped on CRuby, unless the code also takes care of removing that file (I guess mkmf create_makefile could remove it, it seems too annoying if extconf.rb or Rakefile needs to remove it manually).

OTOH if we detect the dummy_makefile pattern, it's immediately picked up as soon as that RubyGems change is in, and JRuby could even cherry-pick it sooner if they care enough about it.
That is why I think this is the best solution as it solves it with very few changes and faster.

Another possibility would be if there is no Makefile created by extconf.rb, then RubyGems would not call make (make in a dir with no Makefile is an error so we cannot just use that).
It's very simple and I think rather intuitive.
And extconf.rb could just e.g. return if RUBY_ENGINE == "jruby"/RUBY_ENGINE != "ruby"/... at the top.
A concern is if one builds (e.g. rake compile) a gem locally on CRuby and then on JRuby.
But rake clean should already remove the Makefile and so that seems no problem.

It is already expected and necessary that one needs to rake clean when switching Rubies before rake compile.

kou (Kouhei Sutou) wrote in #note-9:

> Really? https://rubyinstaller.org/ provides MSYS2 integration by "Devkit". It provides make.

It's mostly relevant for JRuby, it's basically not relevant for CRuby (where it's very common to have non-precompiled native extensions, like erb).

#12 - 01/08/2024 02:11 PM - byroot (Jean Boussier)

> Another possibility would be if there is no Makefile created by extconf.rb, then RubyGems would not call make

I quite like that one.

#13 - 01/09/2024 02:40 PM - vo.x (Vit Ondruch)

I believe that if something complex is needed, RakeBuilder could be used instead:

https://github.com/rubygems/rubygems/blob/master/lib/rubygems/ext/rake_builder.rb

Just FTR, these are currently supported builders:

https://github.com/rubygems/rubygems/blob/8ffa73df3a250623b13c5bf7ab48ed8b8e84e46f/lib/rubygems/ext/builder.rb#L145-L161

#14 - 01/10/2024 09:22 AM - Iloeki (Loic Nageleisen)

> I agree that this isn't a mkmf/extconf.rb issue. This is a rubygems issue.

Sounds like so to me as well (although it may need a bit of support from the ruby side to be solved cleanly)

> Maybe RubyGems could recognize dummy Makefiles

Not too fond of that.

> > Another possibility would be if there is no Makefile created by extconf.rb, then RubyGems would not call make
>
> I quite like that one.

I too, sounds better than the one above.

> Also one might need make anyway, e.g. for gems used via FFI like sassc and prism, where make is used to build the C code

Not necessarily, when platform specific binary gems are available (e.g mini_racer / libv8-node).

> It's mostly relevant for JRuby, it's basically not relevant for CRuby

I would disagree with that, although it seems relevant to CRuby only in a reduced number of cases.

The solutions we found in our case was:

- dependent gem has a hard add_runtime_dependency on the dependency. This is a prerequisite of being able to specify the dependency version requirement at gem install/bundle install time.
- provide binary gems, which then inform of feature availability at runtime (something like MyNamespace.available? returning true)
- the ruby platform gem is either one of:
  - a shim, and provides no implementation whatsoever, thus MyNamespace.available? returns false, which is a bit of a lie since the ruby platform is actually not supported
  - sidestep rubygems platform resolution and attempt fetching binary for that binary platform at gem install time. if it succeeds MyNamespace.available? returns true and succeeds MyNamespace.available? returns false, which is sad as it breaks the common "install does network but does not execute" and "build executes but does not do network" expectation.
  - able to compile from C/Rust source, but limit to a set list of platforms in extconf.rb and otherwise produce a dummy Makefile
  - able to compile from C/Rust source, if compilation succeeds MyNamespace.available? will return true but if compilation fails the result is swallowed, a warning is printed, and MyNamespace.available? will return false

This is all mandated by ruby gems being unable to declare optional dependencies in gemspecs:

```
  s.add_runtime_dependency, foo, '~> 1.1', optional: true
```

Which would mean that if foo is present in the bundle it must be ['>= 1.1', '< 2.0.a'] but if it's not in GEM_HOME / in Gemfile / does not resolve that's okay, for which there are two sides to the coin: dependency is not installed at all or dependency failed to install.

The only way to work around that is at runtime, well after dependency resolution / gem install time:

- have the hard dependency as above, ignore compilation failure / have a ruby platform shim gem, and check for availability at runtime
- not have the hard dependency, and hopefully do the optional gem version check at runtime, duplicating rubygems/bundler dependency resolution (badly)

Note that I'm not positing the above as a solution, merely that it seems to me that it hints at the requirement on creating dummy Makefiles that do nothing might indeed be a deeper rubygems limitation moving a dependency resolution problem to be solved by either build time or runtime hacks.

#15 - 01/10/2024 09:26 AM - ivoanjo (Ivo Anjo)

+1 this would be useful for the ddtrace gem as well, we also do the "if (a bunch of conditions) generate an empty makefile" dance.

#16 - 01/10/2024 10:04 AM - byroot (Jean Boussier)

I tried implementing this idea in rubygems https://github.com/rubygems/rubygems/pull/7372.

I still think it would be nice for mkmf to provide a skip_compilation helper that would generate the dummy makefile (for backward compatibility?) and create the skip file.

#17 - 01/10/2024 10:19 AM - Eregon (Benoit Daloze)

> Another possibility would be if there is no Makefile created by extconf.rb, then RubyGems would not call make

One problem with that though is it would not work on older RubyGems, so if any gem uses that it would fail on existing Ruby releases, which kinda makes it useless unfortunately (at least short term).

That's why I like detecting the dummy makefile approach so much, because it's harmless on existing Ruby releases, it helps immediately for newer Ruby releases/newer RubyGems and even requires no changes in gems (which takes a while and might not work for older RubyGems).

#18 - 01/10/2024 11:39 AM - byroot (Jean Boussier)

I don't think it makes it useless. For now gems can do both (dummy makefile and skip-build file).

And in a few years, as older versions become deprecated, we'll be able to only do the skip-build.

#19 - 01/10/2024 12:04 PM - lloeki (Loic Nageleisen)

There's spec.required_rubygems_version to enforce compatibility. On rubygems versions before the feature a dependency would then resolve to the last one non-compatible with compilation skipping.

Agreed that Benoit's approach could be a transitional solution complementing the above by maintaining backwards compatibility while introducing the feature in short order.

#20 - 01/24/2024 06:18 AM - headius (Charles Nutter)

Finally jumping in here. I think this was initiated by me filing an issue on erb to move the C extension out to a separate gem.

Background: JRuby users traditionally have not needed any build tools **whatsoever** to install JRuby and build and run Rails applications. Everything is pre-built for JVM (though that has changed a bit with sassc).

As it existed a few versions ago, the erb gem would unconditionally try to build the C extension it ships with, which meant that JRuby users could not install it. The first fix was to just emit a dummy Makefile on JRuby, which only half fixes it; JRuby users would still have to have make installed, even though the build would produce **nothing at all**.

Let's be honest, the empty-Makefile trick is a hack at best. There should be a way in RubyGems to indicate that the current OS, OS version, ARCH, RUBY_ENGINE, RUBY_VERSION, etc does not need an extension built, so don't try to build it. That obviously applies in this case (Ruby fallback works fine), but it also applies to any gems that want to ship pre-built native libraries with more customization than just OS and ARCH.

I don't have a strong opinion on "skip dummy makefiles" vs "don't try to build" but the former would integrate more easily. There *are* places where we accept that make will be needed, like sassc (to build the native library bound in Ruby using FFI), and we would not want to disable make for all gems. We just want to fix the ones that really don't need to build, like erb.