Ruby master - Feature #17760
Where we should install a header file when `gem install --user`?
03/30/2021 12:50 AM - mrkn (Kenta Murata)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Third Party's Issue</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**

As digest have been made a default gem at Ruby 3.0, it can be installed by gem install command.

When we install digest, digest.h is installed at the same directory as ruby.h. But when we use gem install --user for installing it, where should digest.h is installed in?

Now, the location of digest.h is always the same directory as ruby.h regardless of whether we use gem install --user. It occurs permission error when non-root user uses gem install --user for installing digest on the system-ruby.

**Related issues:**

- Related to Ruby master - Bug #17761: Install location of header files in exte...
- Closed

**History**

#1 - 03/30/2021 12:51 AM - mrkn (Kenta Murata)
The permission error was reported at https://github.com/ruby/digest/issues/14.

#2 - 03/30/2021 01:06 PM - nobu (Nobuyoshi Nakada)
- Backport set to 2.5: REQUIRED, 2.6: REQUIRED, 2.7: REQUIRED, 3.0: REQUIRED
- Tracker changed from Feature to Bug

#3 - 03/30/2021 02:11 PM - nobu (Nobuyoshi Nakada)
- Related to Bug #17761: Install location of header files in extension libraries added

#4 - 03/30/2021 02:20 PM - nobu (Nobuyoshi Nakada)
- Backport deleted (2.5: REQUIRED, 2.6: REQUIRED, 2.7: REQUIRED, 3.0: REQUIRED)
- Status changed from Open to Third Party's Issue
- Tracker changed from Bug to Feature

This is because rubygems doesn't manage header files installed from gems, since no gems had installed headers until digest has been gemified. Now rubygems needs to support such gems.

#5 - 10/14/2021 03:40 PM - Eregon (Benoit Daloze)
Is there a RubyGems issue tracking this?

#6 - 10/15/2021 07:44 AM - byroot (Jean Boussier)
deivid (David Rodríguez) are you aware of this issue? Any idea how we could fix this or what decision would need to be taken?

I fear that the digest extraction might cause major problem to people trying to upgrade to 3.1 next year, or trying to test 3.1.0-preview1 soon.

#7 - 10/15/2021 08:05 AM - deivid (David Rodríguez)
Hi, no I wasn't, and I'm not sure how it should be fixed. If I understand correctly, this is about cases where rubygems does not have write access to the location where ruby itself is installed, while it does have write access to the location where gems are installed, correct?

#8 - 10/15/2021 11:01 AM - byroot (Jean Boussier)
Kind of yes. From my understanding the problem is that $INSTALL_FILES always install files in site_ruby
https://github.com/ruby/digest/blob/1cb6bac1a1b732a1415276b94faaf966e695477dec/ext/digest/extconf.rb#L7-L9

So from my limited understanding I see two ways out of this:

- Require that site_ruby is user writable (sounds weird)
● Have a location defined by rubygems that acts as "user" site ruby.

But then I suppose them that compile against digest.h would also need to know where to look, I'm really unsure how it's supposed to work. If I have 5 different versions of digest, which one should I compile against?

e.g. I have app1 with digest 3.0.0 and app2 with digest 3.2.0, both have somegem-that-link-to-digest, this means we'd need to compile that gem twice and link against different headers?

Seems complicated. Maybe I should put this to the next developers meeting agenda.

#9 - 10/15/2021 04:25 PM - Eregon (Benoit Daloze)
byroot (Jean Boussier) wrote in #note-8:

But then I suppose them that compile against digest.h would also need to know where to look, I'm really unsure how it's supposed to work. If I have 5 different versions of digest, which one should I compile against?

Because digest is a default gem, I think there is a simple answer here: latest version wins.
That's the behavior of default gems, if you gem install some_default_gem, RubyGems will then always use that version (instead of the version shipped with Ruby).

When using bundler, I'd think it's possible to choose an explicit version, and bundler would resolve which one is used.
Probably RubyGems and/or Bundler need to set CPATH to the header is found (seems better), or we need some explicit API in RubyGems and/or mkmf for a gem to ask headers for another gem (seems bad as then we have the version problem).

#10 - 10/15/2021 04:30 PM - Eregon (Benoit Daloze)
byroot (Jean Boussier) wrote in #note-8:

e.g. I have app1 with digest 3.0.0 and app2 with digest 3.2.0, both have somegem-that-link-to-digest, this means we'd need to compile that gem twice and link against different headers?

This is a good point.
Maybe it can be resolved by somegem-that-link-to-digest depending on an exact version (or small range of versions) of digest which have the same header and fully compatible (both forward and backward between all those versions)?
I.e., the header is also part of API (or technically ABI) so if digest changes the header in some incompatible way (including what gets generated by gems using it), then it should bump the major version (or some other well-documented scheme).

Seems complicated. Maybe I should put this to the next developers meeting agenda.

Yes, I think that would be useful.

#11 - 10/15/2021 05:16 PM - byroot (Jean Boussier)

then it should bump the major version (or some other well-documented scheme).

I can hardly see something like this working in practice. Maybe for digest it would work as realistically it won't change much in the future. But if RubyGems introduce this header file management, other gems will start using it, and I can foresee tons of problems.

#12 - 10/21/2021 09:41 AM - knu (Akinori MUSHA)
We have discussed this in today's developer meeting.

● As a tentative workaround, I'll fix the digest gem not to try to install digest.h to an unwritable directory so gem install --user digest succeeds.
● It is (technically) possible for RubyGems to specify where to install header files by passing HDRDIR=destination to make install for example, and to expose the installed header files of activated gems to a newly built gem by passing CPPFLAGS to extconf.rb that lists the header directories.