Ruby - Feature #20034

[mkmf] Support creating a compilation database for C language tooling

12/01/2023 01:59 PM - pounce (Calvin Lee)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Abstract

Ruby projects are often developed with C extensions. These can be easily compiled by ruby by using create_makefile in mkmf.rb. Unfortunately, most C tooling is unable to interpret the ruby build system for C files, and thus cannot provide useful diagnostics to users about their C extensions.

This request entails generating a compilation database (as discussed below), which would allow C tooling to understand how the extensions are compiled.

Background

A compilation database is a standard format used to describe how C and C++ translation units are compiled. This is primarily used for a build system to communicate to C language tooling (such as a language server). Language servers need this information in order to determine how symbols, definitions and declarations are related in different files; and allows them to show descriptive warnings that may occur during compilation. This also allows users to perform quick and efficient refactors in their text editor.

It is supported by many build systems by default, such as CMake and Meson. These build systems often place a compile_commands.json in the build/ directory, which is automatically detected by most language servers. It is also supported by language servers such as clangd and ccls.

Proposal

When generating a makefile in create_makefile, a compilation database should also be created for all targets that are to be compiled.

Implementation

The location of the compilation database should be configurable, both by library and environment variable, so that a user may redirect the compilation database to a specific location if they desire.

Evaluation

This feature should be evaluated as successful if it is compatible with common language servers such as clangd and ccls. For example, if a user first compiles a project and then edits a C file that is compiled, then clangd should work with little to no configuration.

Discussion

For more information about compilation databases, please see clangd's informational site as well as [the ccls wiki])(https://github.com/MaskRay/ccls/wiki/Project-Setup). There is also an article by Guillaume Papin which discusses compilation databases and software which supports them.

As an alternative, users may use Bear to generate their compilation database. However, this is not optimal, as it only works on Linux. Furthermore, Bear is only able to capture flags for C files which are compiled during a single execution of the build system. Thus, is not able to record flags for C files which have been skipped.

History

#1 - 12/01/2023 04:28 PM - kjtsanaktsidis (KJ Tsanaktsidis)

Oh, I wrote a gem for this a while ago! https://rubygems.org/gems/extconf_compile_commands_json

Does this do what you want? (I haven’t used it in a year or so, so it could just be broken at the moment - if so let me know and I’ll try and fix it).

#2 - 12/01/2023 10:11 PM - pounce (Calvin Lee)

kjtsanaktsidis (KJ Tsanaktsidis) wrote in #note-1:

> Does this do what you want? (I haven’t used it in a year or so, so it could just be broken at the moment - if so let me know and I’ll try and fix it).

I found this during my search! I apologize for not linking to it. It did not work for me (insomuch that I could not find the generated compile-commands.json). I believe the RUBYOPT functionality is quite useful, and is what I needed since I am working on an open source project that I do not own.
I ended up using bear, as described in my request.

However, I still think it would be nice if this functionality were added to upstream ruby. This would make ruby C compilation feature-compatible to other common build systems, and it would make C tooling work automatically in Ruby projects.

#3 - 12/02/2023 12:13 AM - kjtsanaktsidis (KJ Tsanaktsidis)

> It did not work for me (insomuch that I could not find the generated compile-commands.json)

I just had another play around with it - It did work (spat out ./ext/nio4r/compile_commands.json when I tried to compile nio4r, for example, and my clangd setup could go-to-definition). But I *did* realise the snippet in the README about how to wrap rake compile with RUBYOPT was wrong (it was missing an -r). If you do give it another go and it still doesn't work for you, please do open an issue about it over there.

> However, I still think it would be nice if this functionality were added to upstream ruby

mkmf is the thing which has the information about how to generate such a file properly; everything else (like my gem) is just hacking at its implementation details from the outside. So, from a "where is the architecturally correct place for this functionality" perspective, I think I agree.

I wonder if a middle ground between "in cruby" and "in a gem that ~nobody knows about" would be to include this functionality in the rake-compiler gem. In my experience that's how the vast majority of people interact with building C extensions whilst developing them so perhaps that might be a good place to prove out the need for this feature & mature the implementation a bit.

#4 - 12/02/2023 07:17 AM - rubyFeedback (robert heiler)

kjtsanaktsidis wrote:

> I wonder if a middle ground between "in cruby" and "in a gem that ~nobody knows about" would be to include this functionality in the rake-compiler gem.

This may be useful in that the functionality could become part of rake or rake-addons. The total number of people searching, and finding, gems such as the one mentioned above (even more so if it is a rare gem or disjoint from other more common ruby gems/tools), is probably very small (or may never find it, considering how even Google search does not always yield the best results these days), so exposing or "bundling" this via rake (or any other popular tool/gem) would surely be helpful.

#5 - 12/02/2023 01:17 PM - pounce (Calvin Lee)

kjtsanaktsidis (KJ Tsanaktsidis) wrote in #note-3:

> I wonder if a middle ground between "in cruby" and "in a gem that ~nobody knows about" would be to include this functionality in the rake-compiler gem.

This was my initial idea too. I opened an issue in the rake-compiler repository before I opened this issue, and the rake-compiler maintainer directed me here.

#6 - 12/27/2023 12:20 AM - kou (Kouhei Sutou)

I'm the current rake-compiler maintainer.

rake-compiler just runs extconf.rb and uses generated Makefile. If we do this by rake-compiler, rake-compiler needs to parse Makefile or something to generate compile_commands.json. It's not straightforward and optimal.

I think that mkmf.rb not rake-compiler should generate compile_commands.json. Or we may want to develop a new modern mkmf.rb alternative.