Use -Werror=implicit-function-declaration by default for building C extensions

03/08/2022 01:01 PM - Eregon (Benoit Daloze)

Status: Open
Priority: Normal
Assignee: 
Target version: 

Description
Currently, if a C extension refers a non-existing function it will continue to compile and only emit a warning. And compilation warnings are hidden by default for both gem install and bundle install (gem install -V somegem shows them).

A concrete example is the sqlite3 gem, if we use version 1.3.13 it fails only at runtime:

```
$ gem install sqlite3:1.3.13
Fetching sqlite3-1.3.13.gem
Building native extensions. This could take a while...
Successfully installed sqlite3-1.3.13
1 gem installed
```

```
$ ruby -rsqlite3 -e 'db = SQLite3::Database.new "test.db"; p db'
ruby: symbol lookup error: /home/eregon/.rubies/ruby-3.0.2/lib/ruby/gems/3.0.0/gems/sqlite3-1.3.13/lib/sqlite3/sqlite3_native.so: undefined symbol: rb_check_safe_obj
```

This is not nice, it should have failed clearly at compile time, saying the function does not exist.

There is a compiler warning, which can only be seen with (and so most users would miss it):

```
$ gem install -V sqlite3:1.3.13
...
```

```
database.c: In function ‘initialize’;
database.c:60:3: warning: implicit declaration of function ‘rb_check_safe_obj’; did you mean ‘rb_check_safe_str’? [-Wimplicit-function-declaration]
   60 |   rb_check_safe_obj(file);
      | ^~~~~~~~~~~~~~~~~
      | rb_check_safe_str
```

Also multiple CRuby releases are broken on macOS which seems to enable -Werror=implicit-function-declaration by default (e.g., #17777).

EDIT: -Werror=implicit-function-declaration is now default for building CRuby, but not for C extensions.

How about we just always enable -Werror=implicit-function-declaration for all C extensions? (builtin or not).

It:

- shows clear errors early on and shows where the missing function is called (explained just above), instead of delaying them to runtime
- never compiles a call in C with the wrong type due to a missing include

From https://github.com/oracle/truffleruby/issues/2618

History

#1 - 03/08/2022 01:02 PM - Eregon (Benoit Daloze)

- Subject changed from Use -Werror=implicit-function-declaration by default for building C extensions to Use -Werror=implicit-function-declaration by default for building C extensions

#2 - 03/08/2022 01:11 PM - Eregon (Benoit Daloze)

Actually compiling CRuby itself already uses -Werror=implicit-function-declaration (https://github.com/ruby/ruby/blob/1adc7ea630d43e04d5e75b8bbbcf46b72a6e8c45/configure.ac#L650), but that's downgraded to -Wimplicit-function-declaration for C extensions.
It should be -Werror=implicit-function-declaration for C extensions too.

#3 - 03/08/2022 01:12 PM - Eregon (Benoit Daloze)
- Description updated

#4 - 03/08/2022 01:13 PM - Eregon (Benoit Daloze)
- Description updated

#5 - 03/08/2022 01:13 PM - Eregon (Benoit Daloze)
- Description updated

#6 - 03/09/2022 02:55 AM - shyouhei (Shyouhei Urabe)
Mmm... Rubygems' hiding compiler warnings is the root cause of this problem, meseems. Implicit function declaration is a bad idea today, but worked nonetheless before. I doubt if disabling it could break old extension libraries.

#7 - 03/09/2022 12:48 PM - Eregon (Benoit Daloze)
shyouhei (Shyouhei Urabe) wrote in #note-6:

Mmm... Rubygems' hiding compiler warnings is the root cause of this problem, meseems.

I think it is not the root cause, because even if RubyGems showed compiler warnings the relevant warning could easily be buried in the middle of many other warnings, or missed by Ruby users not so familiar with C warnings. Also I imagine most people doing bundle install wouldn't want to see some very minor C linting/no-big-deal warnings as it'd just be noise. It also seems hard to only show C compiler warnings and not pages of compilation output as well.

So the only real fix IMHO is to fail at compilation time for implicit-function-declaration, otherwise it'd still fail at runtime (even if the warning is printed). Also in practice I think it would be hard for most users to make the connection between the C warning to the runtime error (compilation output might be in some other terminal or done way before, etc).

Implicit function declaration is a bad idea today, but worked nonetheless before.

Yes, but it seems the perception on this evolved, notably with macOS or clang making it an error by default in recent versions.

I doubt if disabling it could break old extension libraries.

Maybe, but I think such old extension libraries would have very little chance to work as any function call using a non existing function would abort the whole process.

#8 - 03/10/2022 02:02 AM - shyouhei (Shyouhei Urabe)
There is nothing that a Ruby user can do for this situation. Gem authors should be aware of their bugs. Hiding compiler warnings from their eyes is the problem. bundle install is irrelevant here. I'm not against bundler to remain silent.

#9 - 03/10/2022 02:53 AM - shyouhei (Shyouhei Urabe)
What I have learned so far:
- Authors of sqlite gem have already fixed their code and released a new one.
- The link symbol lookup error occurs when trying to use an older sqlite gem with a newer ruby.
- That had worked before because we basically did not delete deprecated functions, but e91c39f1c0f7d5e670266d9593d533fd4444957f6 changed that.

jeremyevans0 (Jeremy Evans) Do you have any suggestions for this kind of situations?

#10 - 03/10/2022 11:37 AM - Eregon (Benoit Daloze)
shyouhei (Shyouhei Urabe) wrote in #note-8:

There is nothing that a Ruby user can do for this situation. Gem authors should be aware of their bugs. Hiding compiler warnings from their eyes is the problem. bundle install is irrelevant here. I'm not against bundler to remain silent.

Good point, and I think it also makes sense gem install does the same (but anyway probably very few use gem install directly). For gem authors, rake compile (or manually cd ext/... & ruby extconf.rb & make) does show C warnings, so that's fine.

However, gem authors can't predict which functions will be removed in future versions of Ruby (e.g., if the gem is released 2 years before, or if deprecated warnings are hidden by default ...).
IMHO it makes sense to remove deprecated functions after they've been emitting deprecation warnings.

My conclusion is the same: `-Werror=implicit-function-declaration makes for a better and clearer error for everyone, and there seems no value to let the C extension compile but fail at runtime.

#11 - 03/10/2022 12:46 PM - shyouhei (Shyouhei Urabe)
How about adding `rb_check_safe_obj` entry which is `@attribute __error__` to our headers? This is granular than all-or-nothing compiler flag.

#12 - 03/10/2022 01:08 PM - Eregon (Benoit Daloze)
That would maybe help for that specific function and nothing else.

It seems fairly straightforward that in 2022 `-Werror=implicit-function-declaration should be the default for all C programs (and clang/macOS seem to think the same).
It's a clear bug in the source code and should never compile: it can lead to bugs or I think even security issues like reading random memory (due to passing the wrong argument type, when the function does exist).
The fact it was not an error from the start is probably legacy C compiler behavior, which seems no longer relevant today.

This also helps gem authors, now instead of potentially missing a warning (e.g., because there are other warnings, some maybe not in their control from 3rd party sources) which can cause their gem to fail at runtime, it fails at compile time and tell them their gem won't work until they fix that issue. I believe we are helping no one by making such a bug fail at runtime. It seems just sanity when developing C code, really.