Ruby master - Misc #15347

Require C99

11/27/2018 03:53 PM - k0kubun (Takashi Kokubun)

Status:	Closed	
Priority:	Normal	
Assignee:		

Description

Problem

We've spent a lot of time for supporting C90 https://github.com/ruby/ruby/search?o=desc&q=c99&s=committer-date&type=Commits.
It is not only time-consuming but also sometimes sacrifices readability like r66034 (today's commit).

Suggestion

Ruby 3 is planned to be released around 2020. As 30th anniversary of C90, it may be a good time to re-consider dropping support of C90 on the major version upgrade.

Known issues

- Microsoft Visual Studio C++
  - We don't know which VC++ version supports which feature of C99: https://twitter.com/unak/status/1025937122161618944 (Japanese tweet)
- Solaris
  - Solaris Studio 12.4 uses C89 on Solaris 10 by default [Bug #14200]
  - Solaris 10's GCC uses C90 by default [Bug #14751]
  - We don't know how well C99 support is supported on Solaris 10 or 11

Please comment to this ticket if you know more.

Possible approaches

- Investigate C99 features supported by VC++, document allowed C99 features in https://bugs.ruby-lang.org/projects/ruby/wiki/DeveloperHowTo, and somehow prepare CI environment that monitors conformity of it on AppVeyor or Azure Pipelines.
- Solaris seems to support C99 but it's disabled by default. We might be able to enable that on configure and some features we want to use could work.
  - Notable information: Solaris 10 Extended Support ends Jan 2021 http://www.oracle.com/us/support/library/lifetime-support-hardware-301321.pdf. If we assume (of course, not decided yet) Ruby 3 is released in Dec 2020, it'll be just 1-month earlier of it.
- Print warnings for future breakage on Ruby 3 when building Ruby 2.6 or 2.7, for some EOL platforms.

Disclaimer

I'm NOT intending to disrupt users by this.

I just want to make it easier to maintain Ruby interpreter by allowing to use some useful and portable C99 features if and only if almost all Ruby users can compile Ruby interpreter with it by our efforts on build system.
I don't know if it's feasible or not. I want to collect information of it and possibly give up to have the hope for using C99 if it's turned out to be too difficult.

Related issues:

Related to Ruby master - Bug #14200: addr2line.c compile error with C89 compi...	**Closed**
Related to Ruby master - Bug #15519: addr2line compile error on RHEL7	**Closed**

Associated revisions

Revision ec336fb4 - 01/10/2019 08:04 AM - k0kubun (Takashi Kokubun)

configure.ac: Require C99

We already added AC_PROG_CC_C99 in r66605.

This commit stops warning C99 features.

[Misc #15347] [close https://github.com/ruby/ruby/pull/2064]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66771 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66771 - 01/10/2019 08:04 AM - k0kubun (Takashi Kokubun)

configure.ac: Require C99

We already added AC_PROG_CC_C99 in r66605.
This commit stops warning C99 features.

[Misc #15347] [close https://github.com/ruby/ruby/pull/2064]

Revision bfb684c0 - 01/10/2019 08:22 AM - k0kubun (Takashi Kokubun)

NEWS: announce C99 requirement [ci skip]

[Misc #15347]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@66773 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 66773 - 01/10/2019 08:22 AM - k0kubun (Takashi Kokubun)

NEWS: announce C99 requirement [ci skip]

[Misc #15347]

History

#1 - 11/27/2018 04:15 PM - k0kubun (Takashi Kokubun)

For VC++, I heard from usa (Usaku NAKAMURA) (mswin platform maintainer) that he assumes Microsoft does not declare EOL of old VC++s, so he sometimes drops support of old versions as he thinks okay, and he is thinking about requiring vc12 from Ruby 2.7.

As we already have vc12 CI on AppVeyor, we may be able to achieve the following part from Ruby 2.7:

> nvestigate C99 features supported by VC++, document allowed C99 features in https://bugs.ruby-lang.org/projects/ruby/wiki/DeveloperHowTo, and somehow prepare CI environment that monitors conformity of it on AppVeyor or Azure Pipelines.

#2 - 11/27/2018 05:39 PM - alanwu (Alan Wu)

I would like to add that for Clang, -Werror=declaration-after-statement is ignored in c99 mode due to a bug:
https://bugs.llvm.org/show_bug.cgi?id=27493.
Because of this, it's easy to author code that isn't C90 compatible when using Clang.
As a newcomer and someone who uses Clang a lot, I've been bitten by this twice already:
https://bugs.ruby-lang.org/issues/15313#note-4
https://bugs.ruby-lang.org/issues/15331 (v3 of this patch)

So I'm definitely in favor of this because I think dropping C90 will help new contributors.

#3 - 11/27/2018 06:35 PM - shevegen (Robert A. Heiler)

Personally I am not affected either way, so this is for the ruby developers; and those
users who may be affected by the change away from C90 support (which does not include me).

I think it would be nice to drop C90 support, if we say that ruby 3.0 is to be released
(let's pick a date) in 2020 at the latest (well). So C90 ... to the year 2020 ... that is
30 years. Should be more than enough time to drop something old - I guess quite some
ruby hackers may not even be 30 years old. :) (I am really just using the years as a
rough calculation).

> I think dropping C90 will help new contributors.

I am not sure if this will lead to an influx of new contributors, it seems fairly
minor. But I think for devs it is more convenient to have C99, and less time could
be spent for C90 compatibility (I think there were some changes specifically because
of C90 by shyouhei and the C-language specification, but I may remember incorrectly).

> I'm NOT intending to disrupt users by this.

That reminds me of matz' presentation good change, bad change. :)

Sometimes in order to move forward, one has to leave something behind. I think it
depends how many changes are to be made at the same time. If these are just a few
changes then the impact may be very small. The change from ruby 1.8.x to, ultimately,
2.x, was quite a big one (compared to the other changes). Smaller changes are often
a lot easier to deal with than one huge version change + lots of incompatibilities;
so perhaps abandon C90 even before ruby 3.x (IF it is decided that way).

I guess abandoning C90 may come at some moment in time, even if not at ruby 3.0,
so at some other point in the future. If this is the case then matz may have to
decide on the when. Could also come after 3.x (but I guess Takashi may not be
the only one to express this opinion, so perhaps more would favour C90 being
abandoned; that one, in my opinion, is completely up to the ruby dev team).

> I don't know if it's feasible or not. I want to collect information of it and
> possibly give up to have the hope for using C99 if it's turned out to be
> too difficult.

I guess this would be best, no matter when it may come that C90 is no longer
used (e. g. when matz decides, in regards to ruby), to already start collecting
information how many users of ruby may be affected by the change; and how/why.

Perhaps they could even comment here such as Alan Wu commenting on using clang
(I usually use gcc but I also have used clang in the past before). I think
only people on ancient systems may be affected by this ... but I am not sure.
Old old distributions should not have a problem with gcc versions supporting
C99 right? All of GCC 5.x I guess has no problem here - but I have no data
to support my statement. Let's see how and if people use C90 + ruby.

If enough information can be carried forward by people who depend on C90,
we could see how many really need C90 and how many do not; and after that
a decision could be made either way too, so we don't need a decision either
way for now, only more information. If we know that almost nobody will really
be affected by this, then one could have a look at how much change would be
required within the ruby source code - but to be honest, I think the latter
should not be too difficult if we aim for ruby 3.x; ruby is not that huge.

PS: By the way, it may also help if someone puts this on twitter. It helped
a little when Takashi asked on twitter about people showing code and also
benchmarks for when an enabled JIT may slow something down and how it does
so (to then go forward and optimize on these seemingly slow parts).

#4 - 11/27/2018 09:16 PM - duerst (Martin Dürst)

k0kubun (Takashi Kokubun) wrote:

> We've spent a lot of time for supporting C90 https://github.com/ruby/ruby/search?o=desc&q=c99&s=committer-date&type=Commits.
> It is not only time-consuming but also sometimes sacrifices readability like r66034 (today's commit).

First, thanks for r66034. In this case, it ultimately led to better code (see r66043). But I very much agree that at some point, and better sooner than later, we should leave dinosaur compilers behind.

I'm also wondering what other languages similar to Ruby are doing in this respect.

#5 - 11/27/2018 09:32 PM - normalperson (Eric Wong)

> We've spent a lot of time for supporting C90 https://github.com/ruby/ruby/search?o=desc&q=c99&s=committer-date&type=Commits.
> It is not only time-consuming but also sometimes sacrifices readability like r66034 (today's commit).

Agree, C99 struct initializer/labels are a huge benefit to
readability and we can maybe avoid some "init" functions as a
result.

Fwiw, gnulib (GNU portability lib) started using C99 last year;
and GNU programs probably supports more platforms than we do:

```
https://public-inbox.org/bug-gnulib/71472264-68fe-ca47-cee6-4693682db6cb@cs.ucla.edu/
```

recent search results about C99 reveals some issues, but I haven't dug deeper:

```
https://public-inbox.org/bug-gnulib/?q=c99
```

We can start using "restrict" keyword (make it no-op for old
compilers) and see if that leads to performance improvements,
too.

I'm glad we already support most C99 int types.
"bool" would be nice for documentation/legibility, too...

I don't care much for declaration-after-statement, I think it
makes code messier in some cases and encourages big functions;
but IMHO it is nice for loop iterators:

```
for (size_t i = 0; i < limit; i++)
```

> ## Suggestion
>
> Ruby 3 is planned to be released around 2020. As 30th anniversary of C90, it may be a good time to re-consider dropping support of C90 on the major version upgrade.

Anyways, it's unlikely I'll be around in 2020 (or even 2019) so
I won't be around to benefit from it; but I support the change
regardless.

#6 - 11/29/2018 12:22 AM - normalperson (Eric Wong)

edchick@gmail.com wrote:

> This is sad we loss another contributor with deep low level knowledge. I read in another thread you won't be hacking in Ruby again, why is that?

There's thousands of potential contributors more knowledgeable
and skilled than I. Most of them are also willing to use
proprietary communications tools and able to work in
non-text-only environments with tiny fonts.

#7 - 11/30/2018 02:03 PM - cremno (cremno phobia)

Requiring a more modern version of MSVC should be a given anyway. Currently CRuby still tries to support 6.0 (_MSC_VER=1200) which was originally released in 1998 and has seen its last update in 2004. Additional code like in numeric.c and vm_insnhelper.c to support a buggy and obsolete compiler is nothing more than clutter. It just makes the codebase harder to read. Less scrolling, smaller file sizes - what's not to like?

There are quite a few other accommodations CRuby should give up. Missing stuff that's required by C89/C90. For example, functions like strchr, memcmp or fmod, or headers like <float.h> or <limits.h> - just call or include them without any fallback.

Back to Visual Studio: 2012 is the last version capable of running on Windows XP. However newer versions still can target it. It has only little C99 support though (//, long long, flexible array member, etc.).

Requiring a sensible C99 subset would at least require 2013 which added support for: _Bool, compound literals, designed initializes, mixed declarations and code. _Complex and VLAs are still missing (and optional since C11). restrict is also missing but there's __restrict.

The standard library is almost complete in 2013. Functions still missing like snprintf and incomplete format specifier support (also for strftime) aren't an issue though for CRuby (see strftime.c). 2015 provides them but it might be too new.

Btw. CPython 3.6/3.7 require at least 2015 (see PCbuild/readme.txt) and use certain C99 features (see PEP 7). CRuby should too.

#8 - 11/30/2018 09:03 PM - normalperson (Eric Wong)

cremno@mail.ru wrote:

> Back to Visual Studio:

What is the feasability of requiring GCC or clang for Windows builds
instead of supporting VS?

I know GCC can cross-compile for win32, at least; and I got
miniruby to run under WINE, even (100% Free-as-in-freedom stack).

Nobody can say they can't afford to purchase a compiler license
with GCC/clang (anti-GPL people can use clang, because Ruby is BSDL
anyways)

#9 - 12/05/2018 02:27 PM - k0kubun (Takashi Kokubun)

- Related to Bug #14200: addr2line.c compile error with C89 compilers on Solaris 10 added

#10 - 12/05/2018 03:37 PM - naruse (Yui NARUSE)

normalperson (Eric Wong) wrote:

> cremno@mail.ru wrote:
>
> > Back to Visual Studio:
>
> What is the feasability of requiring GCC or clang for Windows builds
> instead of supporting VS?
>
> I know GCC can cross-compile for win32, at least; and I got
> miniruby to run under WINE, even (100% Free-as-in-freedom stack).
>
> Nobody can say they can't afford to purchase a compiler license
> with GCC/clang (anti-GPL people can use clang, because Ruby is BSDL
> anyways)

To link non msvcrt.dll (msvcrXX and vcruntime), it requires related version of Visual C++ compiler.

Anyway as far as I understand, it already doesn't work with VC6,
And dropping VC9 is also acceptable (I already talked with usa about his).

Note that VC 2012's EOL is 2020 but it supports essential part of C99 for CRuby as cremno says.

Oracle Solaris Studio 12 also supports C99; it needs just add -xc99.
https://docs.oracle.com/cd/E18659_01/html/821-1384/bjayy.html

#11 - 12/10/2018 06:18 AM - duerst (Martin Dürst)

duerst (Martin Dürst) wrote:

> k0kubun (Takashi Kokubun) wrote:
>
> > It is not only time-consuming but also sometimes sacrifices readability like r66034 (today's commit).
>
> I very much agree that at some point, and better sooner than later, we should leave dinosaur compilers behind.

I was bitten again a little bit later (r66135, fixed by Koichi Sasada at r66137).

As long as we keep the policy of not using new C features, I wonder why the ./configure script (or whatever else is responsible) doesn't set the relevant warnings. It's okay to check on the continuous integration systems, but it would be much better if every committer by default got the relevant warnings.

If I should create a new issue for changing warning settings, please tell me.

#12 - 12/10/2018 07:04 AM - shyouhei (Shyouhei Urabe)

duerst (Martin Dürst) wrote:

> As long as we keep the policy of not using new C features, I wonder why the ./configure script (or whatever else is responsible) doesn't set the relevant warnings. It's okay to check on the continuous integration systems, but it would be much better if every committer by default got the relevant warnings.
>
> If I should create a new issue for changing warning settings, please tell me.

I did this, 6 years ago, in r36038. Then reverted by naruse (Yui NARUSE) in r54895.

#13 - 12/10/2018 08:07 AM - duerst (Martin Dürst)

shyouhei (Shyouhei Urabe) wrote:

> duerst (Martin Dürst) wrote:
>
> > As long as we keep the policy of not using new C features, I wonder why the ./configure script (or whatever else is responsible) doesn't set the relevant warnings. It's okay to check on the continuous integration systems, but it would be much better if every committer by default got the relevant warnings.

> > If I should create a new issue for changing warning settings, please tell me.
>
> I did this, 6 years ago, in r36038. Then reverted by naruse (Yui NARUSE) in r54895.

Both what was there before r54895 and after, and what's discussed in issue #12336, is all about C99. But as I understand, we are still on C90. So what I'm proposing is to use an option that says that we are on C90, and gives a warning for // comments and for declarations after statements and so no. The gcc man page I'm looking at here mentions an option of -std=C90 that is able to do this.

#14 - 12/12/2018 09:17 AM - shyouhei (Shyouhei Urabe)

duerst (Martin Dürst) wrote:

> So what I'm proposing is to use an option that says that we are on C90, and gives a warning for // comments and for declarations after statements and so no.

This effectively kills 64 bit integer capabilities because there was no such thing like a 64 bit integer back in 1990. This can be okay for CI but not a realistic usage nowadays. Please do not do this by default.

So the whole idea of this issue is not to stick to that old standards. Everybody in this thread are agreeing to move forward. I would suggest just abandoning C90.

#15 - 12/13/2018 08:08 AM - duerst (Martin Dürst)

shyouhei (Shyouhei Urabe) wrote:

> duerst (Martin Dürst) wrote:
>
> > So what I'm proposing is to use an option that says that we are on C90, and gives a warning for // comments and for declarations after statements and so no.
>
> This effectively kills 64 bit integer capabilities because there was no such thing like a 64 bit integer back in 1990.

I agree we don't want to do this.

> This can be okay for CI but not a realistic usage nowadays. Please do not do this by default.

Then what about by default activating options that warn for // comments or for declarations after statements one by one? The list of warnings e.g. in gcc is very long, but I just found -Wdeclaration-after-statement. So I would definitely suggest that this warning is switched on as long as we require commits to conform to this and check this in CI. This warning alone would have avoided problems in two of my recent commits (one of them https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/66034, which was the reason this thread started).

Unfortunately, I haven't found a warning specifically for // comments.

> So the whole idea of this issue is not to stick to that old standards. Everybody in this thread are agreeing to move forward. I would suggest just abandoning C90.

I am as much as you for moving ahead. But as long as we stay on C90 (for certain parts of the syntax), I want to make it easy for committers, both those who occasionally forget to avoid declarations after statements (like me) and those who help with cleanup (such as ko1 and k0kubun).

#16 - 12/13/2018 10:35 AM - duerst (Martin Dürst)

duerst (Martin Dürst) wrote:

> I just found -Wdeclaration-after-statement. So I would definitely suggest that this warning is switched on as long as we require commits to conform to this and check this in CI. This warning alone would have avoided problems in two of my recent commits (one of them https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/66034, which was the reason this thread started).

Sorry, I was wrong. https://bugs.ruby-lang.org/projects/ruby-trunk/repository/revisions/66034 was a different C90/C99 issue, "initializer for aggregate is not a compile-time constant". I'm not sure there is a warning specifically for that, but if there is, we should also activate it in the general make process, not only on ci.

#17 - 01/09/2019 04:16 PM - vo.x (Vit Ondruch)

- Related to Bug #15519: addr2line compile error on RHEL7 added

#18 - 01/10/2019 07:55 AM - k0kubun (Takashi Kokubun)

This is accepted for Ruby 2.7 at DevelopersMeeting20190110Japan [Misc #15462]. I'll take following actions:

- Disable change C90 checking on Travis to C99 https://github.com/ruby/ruby/pull/2064
- Fix configure as needed
    - -std=c99 or the like is already passed. Still we would need to stop passing a flag that warns C99 feature.
- Announce C99 usage somewhere
    - I'm planning to use NEWS
    - Also I'll post the description in https://github.com/ruby/ruby/pull/2064 to Wiki of bugs.ruby-lang.org

#19 - 01/10/2019 08:04 AM - k0kubun (Takashi Kokubun)

- Status changed from Open to Closed

Applied in changeset trunk|r66771.

---

configure.ac: Require C99

We already added AC_PROG_CC_C99 in r66605.
This commit stops warning C99 features.

[Misc #15347] [close https://github.com/ruby/ruby/pull/2064]

#20 - 01/10/2019 08:16 AM - k0kubun (Takashi Kokubun)

For developer's reference, I created https://bugs.ruby-lang.org/projects/ruby-trunk/wiki/C99 and linked it from https://bugs.ruby-lang.org/projects/ruby/wiki/DeveloperHowto.