Ruby master - Bug #18007
Help developers of C extensions meet requirements in "doc/extension.rdoc"
06/25/2021 05:54 AM - mdalessio (Mike Dalessio)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</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>
<tr>
<td>ruby -v:</td>
<td>Backport:</td>
</tr>
<tr>
<td></td>
<td>2.6: REQUIRED, 2.7: REQUIRED, 3.0: DONE</td>
</tr>
</tbody>
</table>

Description
A pull request for this feature has been submitted at https://github.com/ruby/ruby/pull/4604

Problem being solved
This option is intended to help developers of C extensions to check if their code meets the requirements explained in "doc/extension.rdoc". Specifically, I want to ensure that T_DATA object classes undefine or redefine the alloc function.

https://github.com/ruby/ruby/blob/6963f8f743b42f9004a0879cd66c550f18987352/doc/extension.rdoc#label-Write+the+C+Code says:

Since Object.allocate allocates an ordinary T_OBJECT type (instead of T_DATA), it's important to either use rb_define_alloc_func() to overwrite it or rb.UndefAlloc_func() to delete it.

(note: which matters when using TypedData_Make_Struct/TypedData_Wrap_Struct as the native pointer is supplied without calling the class alloc function).

There is currently no easy way for an author of a C extension to easily see where they have made the mistake of letting the default alloc function remain for their class (and therefore class.new creating a T_OBJECT instead of T_DATA and not setting the data pointer).

Description of the solution
Compiled with this option, Ruby will warn when a T_DATA object is created whose class has not undefined or redefined the alloc function.

A new function is defined, rb_data_object_check. That function is called from rb_data_object_wrap() and rb_data_typed_object_wrap() (which implement the DataWrap_Struct family of macros).

The warning, when emitted, looks like this:

warning: T_DATA class Nokogiri::XML::Document should undefine or redefine the alloc function, please see doc/extension.rdoc

Examples of this problem in the wild
Using this option, I found that many of Nokogiri's classes needed to redefine allocate.

This PR also updates these core Ruby classes by undefining allocate:

- ObjectSpace::InternalObjectWrapper
- Socket::Ifadd

Questions for reviewers
Does this check really need to be behind a configuration option? Performance impact is very small (see benchmarks below), but I put it behind a flag because I am worried that there may be a many C extensions that would emit warnings at runtime, and the users of those extensions cannot fix the problem and so would mostly just be annoyed.

Should this warning be emitted with the deprecated category?
Benchmarking

I benchmarked this code by allocating Nokogiri::XML::NodeSets in a loop. This is a class with a relatively simple allocate function.

The runs cover the four combinations of enabled/disabled, and warnings/no-warnings.

```ruby
ruby 3.1.0dev (2021-06-25T04:02:18Z flavorjones-extens.. de943189aa) [x86_64-linux]
Warming up --------------------------------------
disabled, warn=false 490.143k i/100ms
Calculating -------------------------------
disabled, warn=false 4.863M (± 1.5%) i/s - 49.014M in 10.081177s
```

```ruby
ruby 3.1.0dev (2021-06-25T04:02:18Z flavorjones-extens.. de943189aa) [x86_64-linux]
Warming up --------------------------------------
disabled, warn=true 483.070k i/100ms
Calculating -------------------------------
disabled, warn=true 4.839M (± 1.4%) i/s - 48.790M in 10.083899s
```

Comparison:
disabled, warn=false: 4863064.0 i/s
disabled, warn=true: 4839310.1 i/s - same-ish: difference falls within error

disabled, warn=false: 4840123.2 i/s - same-ish: difference falls within error
disabled, warn=true: 4839310.1 i/s - same-ish: difference falls within error

disabled, warn=true: 4866434.8 i/s
```

My conclusion is that the performance impact is very small, and we could omit the option if the Ruby core maintainers decide this behavior should be on by default.

Associated revisions

Revision e8e3b7a0 - 08/19/2021 11:30 PM - mdalessio (Mike Dalessio)

Undefine the alloc function for T_DATA classes which have not undefined or redefined it.

When a T_DATA object is created whose class has not undefined or redefined the alloc function, the alloc function now gets undefined by Data_Wrap_Struct et al. Optionally, a future release may also warn that this being done.

This should help developers of C extensions to meet the requirements explained in "doc/extension.rdoc". Without a check like this, there is no easy way for an author of a C extension to see where they have made a mistake.
undefine alloc functions for C extensions

per guidance in doc/extension.rdoc, these classes now undefine their alloc functions:

- ObjectSpace::InternalObjectWrapper
- Socket::Ifaddr

merge revision(s) c0f4e4ca6d0f76985bca79314b232b787c8f008e: [Backport #18007]

---

ext/objspace/objspace.c | 1 +
ext/socket/ifaddr.c | 1 +
2 files changed, 2 insertions(+)

History

#1 - 06/25/2021 07:25 AM - nobu (Nobuyoshi Nakada)
I don't think the configuration option is needed. And Check_Type(klass, T_CLASS); can be in rb_data_object_check.

#2 - 06/25/2021 01:50 PM - mdalessio (Mike Dalessio)
nobu (Nobuyoshi Nakada) wrote in #note-1:
I don't think the configuration option is needed. And Check_Type(klass, T_CLASS); can be in rb_data_object_check.

Thank you for your review. I've updated the pull request.

#3 - 07/15/2021 09:07 AM - naruse (Yui NARUSE)
I want Ruby 3.1 to be compatible with 3.0. Therefore even if it is accepted, you can merge this in 3.2.

#4 - 07/15/2021 11:50 AM - mdalessio (Mike Dalessio)
Naruse, thank you for your time.

Would you consider for 3.1 if this change was behind a configuration option that defaulted to disabled?

#5 - 08/17/2021 06:01 AM - mame (Yusuke Endoh)
How about raising an exception when attempting to allocate a T_DATA object with the default allocator?

#6 - 08/17/2021 06:44 AM - nobu (Nobuyoshi Nakada)
mame (Yusuke Endoh) wrote in #note-5:
How about raising an exception when attempting to allocate a T_DATA object with the default allocator?

Though I agreed it once, no way to tell if the given class is T_DATA in the default allocator.

#7 - 08/17/2021 08:25 AM - Eregon (Benoit Daloze)
I'm confused, why does Class#allocate not simply use the alloc function set with rb_define_alloc_func? Then there would be no need to undefine/redefine allocate, isn't it?

(IMHO users should never call allocate directly because it returns an uninitialized object which is likely to break when used, but that probably needs documentation, more thoughts, etc)

#8 - 08/17/2021 09:29 AM - nobu (Nobuyoshi Nakada)
Eregon (Benoit Daloze) wrote in #note-7:

I'm confused, why does Class#allocate not simply use the alloc function set with rb_define_alloc_func?

You're confused by Class#allocate which calls the allocator set for each T_DATA and rb_class_allocate which implements the default allocator for T_OBJECT. The former method now calls the latter function set with rb_define_alloc_func.

#9 - 08/19/2021 05:40 AM - matz (Yukihiro Matsumoto)
In my opinion, it should be merged, and it should cause errors (not warnings) in the future. But we need a migration path to ease the pain of the change.

Matz.

#10 - 08/19/2021 06:26 AM - nobu (Nobuyoshi Nakada)
naruse (Yui NARUSE) objects to that warning appearing every time that instance is created. So I propose to undefine the allocator in that check, so an exception will be raised when new or allocate are called later.

```c
static inline void
rb_data_object_check(VALUE klass)
{
    if (klass != rb_cObject && (rb_get_alloc_func(klass) == rb_class_allocate_instance)) {
        rb_undef_alloc_func(klass);
        #if 0 // TODO: enable at the next release
        rb_warn("allocator of T_DATA class "PRIzVALUE" got undefined", klass);
        #endif
    }
}
```

#11 - 08/19/2021 01:51 PM - mdalessio (Mike Dalessio)
Thank you for your time, Matz, Naruse, and Nobu.

I've made the requested change in the pull request to undefine the allocator in the check, and the warning is hidden by #if 0.

#12 - 08/19/2021 02:41 PM - Eregon (Benoit Daloze)
- Description updated

#13 - 08/19/2021 02:43 PM - Eregon (Benoit Daloze)
- Description updated

#14 - 08/19/2021 02:44 PM - Eregon (Benoit Daloze)
- Description updated

#15 - 08/19/2021 02:47 PM - Eregon (Benoit Daloze)
My confusion was that the description of this issue talks about the allocate method, but actually it's about the alloc function. In CRuby there is typically only one allocate method, Class#allocate, and that uses the alloc function associated with that class and goes up superclasses until it find one or finds UNDEFALLOC_FUNC (which causes TypeError "allocator undefined for Foo"). rb_define_alloc_func() and rb_undef_alloc_func() set the alloc function, they don't change any method (but of course they affect the behavior of Class#allocate which looks at the alloc function).

#16 - 08/19/2021 02:50 PM - Eregon (Benoit Daloze)
- Description updated

#17 - 08/19/2021 05:32 PM - mdalessio (Mike Dalessio)
Benoit, thanks for helping clarify the language in the description. I've also updated the PR commit log to use this language.

#18 - 08/19/2021 11:56 PM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed

#19 - 08/19/2021 11:58 PM - nobu (Nobuyoshi Nakada)
- Backport set to 2.6: REQUIRED, 2.7: REQUIRED, 3.0: REQUIRED
- Tracker changed from Feature to Bug
At least c0f4e4ca needs to be backported, since p ObjectSpace::InternalObjectWrapper.new segfaults.

#20 - 09/05/2021 05:56 AM - nagachika (Tomoyuki Chikanaga)
- Backport changed from 2.6: REQUIRED, 2.7: REQUIRED, 3.0: REQUIRED to 2.6: REQUIRED, 2.7: REQUIRED, 3.0: DONE

ruby_3_0 c42208f8e24402fe1aa8747901fba275bf0d56b merged revision(s) c0f4e4ca6d0f76985bca79314b232b787c8f008e.