Ruby - Bug #20311

Struct.new("A") memory leak?

02/28/2024 12:03 PM - MaxLap (Maxime Lapointe)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	3.3.0	**Backport:**	3.0: WONTFIX, 3.1: REQUIRED, 3.2: DONE, 3.3: DONE

Description

The following code gives the impression of a memory leak.

```
10.times do
  5000.times do
    Struct.new("A")
    Struct.send(:remove_const, :A)
  end

  GC.start
  puts `ps -o rss= -p #{$$}`.to_i
end

27868
35324
43400
51472
58676
66144
73764
81196
88512
95752
```

Is there another location where the struct gets set that I need to clear up for the GC free the memory?

Happens in 3.2.2, 3.2.3, 3.3.0, 3.3-head, ruby-head.

Associated revisions

Revision e626da82eae3d437b84d4f9ead0164d436b08e1a - 03/01/2024 07:23 AM - byroot (Jean Boussier)

Don't pin named structs defined in Ruby

[Bug #20311]

rb_define_class_under assumes it's called from C and that the
reference might be held in a C global variable, so it adds the
class to the VM root.

In the case of Struct.new('Name') it's wasteful and make
the struct immortal.

Revision e626da82eae3d437b84d4f9ead0164d436b08e1a - 03/01/2024 07:23 AM - byroot (Jean Boussier)

Don't pin named structs defined in Ruby

[Bug #20311]

rb_define_class_under assumes it's called from C and that the
reference might be held in a C global variable, so it adds the
class to the VM root.

In the case of Struct.new('Name') it's wasteful and make
the struct immortal.

Revision e626da82 - 03/01/2024 07:23 AM - byroot (Jean Boussier)

Don't pin named structs defined in Ruby

[Bug #20311]

rb_define_class_under assumes it's called from C and that the
reference might be held in a C global variable, so it adds the
class to the VM root.

In the case of Struct.new('Name') it's wasteful and make
the struct immortal.

Revision f3af5ae7e6c1c096bbfe46d69de825a02b1696cf - 03/01/2024 03:33 PM - byroot (Jean Boussier)

Make Struct memory leak test faster

[Bug #20311]

It times out on some platform, so we can reduce iterations.
On my machine it completes in 250ms and RSS grows 8X.

Revision f3af5ae7e6c1c096bbfe46d69de825a02b1696cf - 03/01/2024 03:33 PM - byroot (Jean Boussier)

Make Struct memory leak test faster

[Bug #20311]

It times out on some platform, so we can reduce iterations.
On my machine it completes in 250ms and RSS grows 8X.

Revision f3af5ae7 - 03/01/2024 03:33 PM - byroot (Jean Boussier)

Make Struct memory leak test faster

[Bug #20311]

It times out on some platform, so we can reduce iterations.
On my machine it completes in 250ms and RSS grows 8X.

Revision f79b1d1ef1f7aa64d20f0eadbb3b0f8f7084deb3 - 03/21/2024 05:31 AM - NARUSE, Yui

merge revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a,f3af5ae7e6c1c096bbfe46d69de825a02b1696cf: [Backport #20311] (#10312)

Don't pin named structs defined in Ruby

```
    [Bug #20311]

    `rb_define_class_under` assumes it's called from C and that the
    reference might be held in a C global variable, so it adds the
    class to the VM root.

    In the case of `Struct.new('Name')` it's wasteful and make
    the struct immortal.

    Make Struct memory leak test faster

    [Bug #20311]

    It times out on some platform, so we can reduce iterations.
    On my machine it completes in 250ms and RSS grows 8X.
```

Revision f79b1d1ef1f7aa64d20f0eadbb3b0f8f7084deb3 - 03/21/2024 05:31 AM - NARUSE, Yui

merge revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a,f3af5ae7e6c1c096bbfe46d69de825a02b1696cf: [Backport #20311] (#10312)

Don't pin named structs defined in Ruby

```
    [Bug #20311]

    `rb_define_class_under` assumes it's called from C and that the
    reference might be held in a C global variable, so it adds the
    class to the VM root.

    In the case of `Struct.new('Name')` it's wasteful and make
    the struct immortal.
```

```
    Make Struct memory leak test faster

    [Bug #20311]

    It times out on some platform, so we can reduce iterations.
    On my machine it completes in 250ms and RSS grows 8X.
```

Revision f79b1d1e - 03/21/2024 05:31 AM - NARUSE, Yui

merge revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a,f3af5ae7e6c1c096bbfe46d69de825a02b1696cf: [Backport #20311] (#10312)

Don't pin named structs defined in Ruby

```
    [Bug #20311]

    `rb_define_class_under` assumes it's called from C and that the
    reference might be held in a C global variable, so it adds the
    class to the VM root.

    In the case of `Struct.new('Name')` it's wasteful and make
    the struct immortal.

    Make Struct memory leak test faster

    [Bug #20311]

    It times out on some platform, so we can reduce iterations.
    On my machine it completes in 250ms and RSS grows 8X.
```

Revision bd5df1693c89d389471d145fc19b487c708912b1 - 07/07/2024 07:44 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a, f3af5ae7e6c1c096bbfe46d69de825a02b1696cf: [Backport #20311]

```
    Don't pin named structs defined in Ruby

    [Bug #20311]

    `rb_define_class_under` assumes it's called from C and that the
    reference might be held in a C global variable, so it adds the
    class to the VM root.

    In the case of `Struct.new('Name')` it's wasteful and make
    the struct immortal.

    Make Struct memory leak test faster

    [Bug #20311]

    It times out on some platform, so we can reduce iterations.
    On my machine it completes in 250ms and RSS grows 8X.
```

Revision bd5df1693c89d389471d145fc19b487c708912b1 - 07/07/2024 07:44 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a, f3af5ae7e6c1c096bbfe46d69de825a02b1696cf: [Backport #20311]

```
    Don't pin named structs defined in Ruby

    [Bug #20311]

    `rb_define_class_under` assumes it's called from C and that the
    reference might be held in a C global variable, so it adds the
    class to the VM root.

    In the case of `Struct.new('Name')` it's wasteful and make
    the struct immortal.

    Make Struct memory leak test faster

    [Bug #20311]

    It times out on some platform, so we can reduce iterations.
    On my machine it completes in 250ms and RSS grows 8X.
```

Revision bd5df169 - 07/07/2024 07:44 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a, f3af5ae7e6c1c096bbfe46d69de825a02b1696cf: [Backport #20311]

```
    Don't pin named structs defined in Ruby

    [Bug #20311]

    `rb_define_class_under` assumes it's called from C and that the
    reference might be held in a C global variable, so it adds the
    class to the VM root.

    In the case of `Struct.new('Name')` it's wasteful and make
    the struct immortal.

    Make Struct memory leak test faster

    [Bug #20311]

    It times out on some platform, so we can reduce iterations.
    On my machine it completes in 250ms and RSS grows 8X.
```

History

#1 - 02/28/2024 01:28 PM - byroot (Jean Boussier)

I had a quick look at this using ObjectSpace.dump_all, and it appears that these classes are being retained in the VM root:

```
require 'objspace'

3.times do
  puts ObjectSpace.dump(Struct.new("A"))
  Struct.send(:remove_const, :A)
end

GC.start
ObjectSpace.dump_all(output: File.open("/tmp/heap.json", "w+"))

{"address":"0x102bc33f8", "type":"CLASS", ....
{"address":"0x102bc3218", "type":"CLASS", ....
{"address":"0x102bc3038", "type":"CLASS", ....

$ rg -F '"0x102bc3218"' /tmp/heap.json
1:{"type":"ROOT", "root":"vm", "references":[..., "0x102bc3218"
$ rg -F '"0x102bc33f8"' /tmp/heap.json
1:{"type":"ROOT", "root":"vm", "references":[..., "0x102bc33f8"
```

#2 - 02/28/2024 04:36 PM - nobu (Nobuyoshi Nakada)

Seems referred from defined_module_hash.
rb_const_remove needs to take care of it when the removed constant is a class or module?

#3 - 02/29/2024 11:45 AM - byroot (Jean Boussier)

> rb_const_remove needs to take care of it when the removed constant is a class or module?

The problem is that it's a set:

```
st_insert(vm->defined_module_hash, (st_data_t)module, (st_data_t)module);
```

So if a module is added twice, you need to remove it from the set if it's removed twice... I'll see if I can do that.

But I also wonder why we have this root in the first place, I'd think these get marked from being constants in ::Object.

#4 - 02/29/2024 11:47 AM - byroot (Jean Boussier)

> But I also wonder why we have this root in the first place, I'd think these get marked from being constants in ::Object.

Nevermind, if I understand correctly, it's not to mark them, it's to pin them.

#5 - 02/29/2024 12:07 PM - byroot (Jean Boussier)

I think https://github.com/ruby/ruby/pull/10143 will do. But a good followup would also be to not pin Struct.new("A").

#6 - 02/29/2024 12:08 PM - byroot (Jean Boussier)

- Backport changed from 3.0: UNKNOWN, 3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN to 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: REQUIRED

#7 - 02/29/2024 02:59 PM - byroot (Jean Boussier)

So https://github.com/ruby/ruby/pull/10143 is turning into a huge yak-shave and will be hard to backport. The reason is a lot of code end up relying on classes defined in C becoming immortal (in addition to being pinned).

So maybe a better short term solution that is easy to backport is to just not use these API for naming structs: https://github.com/ruby/ruby/pull/10144

#8 - 03/01/2024 06:03 AM - nobu (Nobuyoshi Nakada)

byroot (Jean Boussier) wrote in #note-3:

> So if a module is added twice, you need to remove it from the set if it's removed twice... I'll see if I can do that.

Is there any chance for the same module to be added twice?

#9 - 03/01/2024 06:04 AM - nobu (Nobuyoshi Nakada)

- Description updated

#10 - 03/01/2024 07:01 AM - nobu (Nobuyoshi Nakada)

byroot (Jean Boussier) wrote in #note-7:

> So maybe a better short term solution that is easy to backport is to just not use these API for naming structs:
> https://github.com/ruby/ruby/pull/10144

Seems fine, but I'm afraid what can happen when an extension library stores rb_path2class("Struct::A") result then remove that constant?
Probably this is same for any Ruby-level defined classes/modules.

#11 - 03/01/2024 07:23 AM - byroot (Jean Boussier)

> Is there any chance for the same module to be added twice?

Yes, aliasing isn't uncommon.

> Seems fine, but I'm afraid what can happen when an extension library stores rb_path2class("Struct::A") result then remove that constant?
> Probably this is same for any Ruby-level defined classes/modules.

Yes, my reasoning is that it's not reasonable to expect classes created in Ruby to be immortal and immovable like the ones created in C.

#12 - 03/01/2024 07:23 AM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset git|e626da82eae3d437b84d4f9ead0164d436b08e1a.

Don't pin named structs defined in Ruby

[Bug #20311]

rb_define_class_under assumes it's called from C and that the
reference might be held in a C global variable, so it adds the
class to the VM root.

In the case of Struct.new('Name') it's wasteful and make
the struct immortal.

#13 - 03/21/2024 06:58 AM - naruse (Yui NARUSE)

- Backport changed from 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: REQUIRED to 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: DONE

ruby_3_3 f79b1d1ef1f7aa64d20f0eadbb3b0f8f7084deb3 merged revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a,f3af5ae7e6c1c096bbfe46d69de825a02b1696cf.

#14 - 07/07/2024 11:26 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 3.0: WONTFIX, 3.1: REQUIRED, 3.2: REQUIRED, 3.3: DONE to 3.0: WONTFIX, 3.1: REQUIRED, 3.2: DONE, 3.3: DONE

ruby_3_2 bd5df1693c89d389471d145fc19b487c708912b1 merged revision(s) e626da82eae3d437b84d4f9ead0164d436b08e1a, f3af5ae7e6c1c096bbfe46d69de825a02b1696cf.