Ruby master - Bug #14372

Memory leak in require with Pathnames in the $LOAD_PATH in 2.3/2.4

01/18/2018 04:57 PM - jrafanie (Joe Rafaniello)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.3p205 (2017-12-14 revision 61247) [x86_64-darwin16]	**Backport:**	2.3: REQUIRED, 2.4: DONE, 2.5: DONTNEED

Description

There is a memory leak that we have found on ruby 2.3.6 and 2.4.3 that happens on Mac OSX and Linux. Ruby 2.2.6 and 2.5.0 do not leak. We have not tested other platforms.

If $LOAD_PATH contains one or more Pathname objects, require without a fully qualified path, such as require 'ostruct', will leak if you do this require many times.

For example, the following script will leak very quickly on ruby 2.3.6 and 2.4.3:

```
require 'pathname'

puts Process.pid
$LOAD_PATH.unshift(Pathname.new(__dir__))

dot      = "."
filename = "ostruct"
1000.times { 1000.times { require filename }; print dot; GC.start; }
```

From what we can understand, it appears that rb_require_internal calls rb_feature_p which ultimately calls rb_file_expand_path_fast and resizes a string. It doesn't seem like the memory is ever freed either in c or the garbage collector. This happens many times, perhaps because Pathname objects, unlike Strings, aren't cached in loaded features so they get expanded each time. See below:

https://github.com/ruby/ruby/blob/v2_3_6/load.c#L43-L47

We can workaround this problem by converting Pathname objects in $LOAD_PATH to Strings, but this leak should be fixed since this it's common to use Pathname objects in $LOAD_PATH.

We used the instruments tool on OSX to show one such leak and the callstack, see attached image.

Related issues:

Related to Ruby master - Bug #10222: require_relative and require should be c...	**Closed**

Associated revisions

Revision eaba9da1 - 02/16/2018 04:25 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 59983,59984: [Backport #10222] [Backport #14372] [Backport #14424]

```
    file.c: rb_check_realpath

    * file.c (rb_check_realpath): returns real path which has no
      symbolic links.  similar to rb_realpath except for returning
      Qnil if any parts did not exist.

    load.c: real path to load

    * load.c (rb_construct_expanded_load_path): expand load paths to
      real paths to get rid of duplicate loading from symbolic-linked
      directories.  [Feature #10222]
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@62440 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 62440 - 02/16/2018 04:25 PM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 59983,59984: [Backport #10222] [Backport #14372] [Backport #14424]

```
file.c: rb_check_realpath

* file.c (rb_check_realpath): returns real path which has no
  symbolic links.  similar to rb_realpath except for returning
  Qnil if any parts did not exist.

load.c: real path to load

* load.c (rb_construct_expanded_load_path): expand load paths to
  real paths to get rid of duplicate loading from symbolic-linked
  directories.  [Feature #10222]
```

History

#1 - 01/18/2018 05:16 PM - jrafanie (Joe Rafaniello)

It's worth mentioning that more Pathname objects in the $LOAD_PATH may make this leak worse as even on ruby 2.5.0, the time for require increases with each added Pathname to the $LOAD_PATH.

#2 - 01/18/2018 09:34 PM - jrafanie (Joe Rafaniello)

I did a small change to see how the number of Pathnames in the $LOAD_PATH changes the leak amount at the script's completion.

It looks like the memory leak is linear:
1 74.6 MB
2 149.5 MB
3 214 MB
4 290 MB
5 353.6 MB
9 575.4 MB
10 650.6 MB

Here's the script:

```
require 'pathname'

puts Process.pid

puts ARGV[0]
(ARGV[0] || 1).to_i.times { $LOAD_PATH.unshift(Pathname.new(__dir__) ) }

dot      = "."
filename = "ostruct"
1000.times { 1000.times { require filename }; print dot; GC.start; }

STDOUT.puts "exit?"
STDIN.gets
```

Additionally, I measured the time to do the requires only changing the number of Pathnames in the $LOAD_PATH:

1 Pathname, 6.1s
2 Pathname, 8s
3 Pathname, 10.1s
4 Pathname, 12.2s
5 Pathname, 13.4s
9 Pathname, 20.3s
10 Pathname, 22.2s

```
require 'pathname'

puts Process.pid

puts ARGV[0]
(ARGV[0] || 1).to_i.times { $LOAD_PATH.unshift(Pathname.new(__dir__) ) }

dot      = "."
filename = "ostruct"
1000.times { 1000.times { require filename }; print dot; GC.start; }
```

#3 - 01/26/2018 05:19 PM - jrafanie (Joe Rafaniello)

Because Rails.root is a Pathname, it's a fairly common for developers to use Rails.root.join("lib") or something similar in their autoload_paths or eager_load_paths, both of which end up in the $LOAD_PATH and lead to a leak on each call to require.

This memory has been found in various open source projects and workarounds provided (convert Pathname objects destined for the $LOAD_PATH to strings):

https://github.com/lobsters/lobsters/pull/449
https://github.com/opf/openproject/pull/6148
https://github.com/errbit/errbit/pull/1257
https://github.com/ManageIQ/manageiq/pull/16837
https://github.com/ManageIQ/manageiq-api/pull/288
https://github.com/ManageIQ/manageiq-automation_engine/pull/146
https://github.com/ManageIQ/manageiq-ui-classic/pull/3266
https://github.com/ManageIQ/manageiq-graphql/pull/34

#4 - 01/28/2018 06:40 PM - oliverguenther (Oliver Günther)

jrafanie (Joe Rafaniello) wrote:

> it's a fairly common for developers to use Rails.root.join("lib") or something similar in their autoload_paths or eager_load_paths, both of which end up in the $LOAD_PATH and lead to a leak on each call to require.

As linked by Joe above, exactly this happened to us at OpenProject and had a noticeable impact. I assume this will affect all larger Rails applications. As such, we endorse addressing of this bug.

Best,
Oliver

#5 - 01/30/2018 01:42 AM - wanabe (_ wanabe)

According to git bisect, this leak had been introduced at r51360 [Bug #11386] and this was fixed at r59984 [Feature #10222] on trunk.

#6 - 01/30/2018 08:27 AM - nobu (Nobuyoshi Nakada)

- Related to Bug #10222: require_relative and require should be compatible with each other when symlinks are used added

#7 - 01/30/2018 08:27 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

#8 - 01/30/2018 06:56 PM - jrafanie (Joe Rafaniello)

I opened a backport bug so we can have this memory leak fixed in ruby 2.3 and 2.4:
https://bugs.ruby-lang.org/issues/14424

#9 - 02/16/2018 04:26 PM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN to 2.3: REQUIRED, 2.4: DONE, 2.5: DONTNEED

ruby_2_4 r62440 merged revision(s) 59983,59984.

Files

Instruments2 2018-01-18 11-43-32.png	181 KB	01/18/2018	jrafanie (Joe Rafaniello)