Ruby - Bug #12159

Thread::Backtrace::Location#path returns absolute path for files loaded by require_relative

03/09/2016 01:27 AM - tagomoris (Satoshi Tagomori)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:			
ruby -v:	ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-darwin14]	**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

I expected that Thread::Backtrace::Location#path always returns base filename, but returns absolute path for files loaded by require_relative.
Is it intentional? or a bug?

```
$ ruby -v
ruby 2.3.0p0 (2015-12-25 revision 53290) [x86_64-darwin14]
$ cat > x.rb
def a
  caller_locations
end

p a.first.path
$ cat > y.rb
require_relative "x"
$ ruby x.rb
"x.rb"
$ ruby y.rb
"/Users/tagomoris/x.rb"
```

History

#1 - 03/09/2016 01:40 AM - usa (Usaku NAKAMURA)

- Status changed from Open to Assigned

- Assignee set to ko1 (Koichi Sasada)

I guess that it's intentional.
absolute_path guarantees to contain the absolute path, but path does not guarantee so.
It may contain the absolute path or may not.

note: not only in require_relative but also in require.

#2 - 03/09/2016 01:43 AM - tagomoris (Satoshi Tagomori)

Usaku NAKAMURA wrote:

> I guess that it's intentional.
> absolute_path guarantees to contain the absolute path, but path does not guarantee so.
> It may contain the absolute path or may not.
>
> note: not only in require_relative but also in require.

If so, I think it's better to write "Returns the file name or absolute path of this frame" in document.
http://docs.ruby-lang.org/en/2.3.0/Thread/Backtrace/Location.html#method-i-path

#3 - 03/16/2016 10:17 AM - shevegen (Robert A. Heiler)

Agreed - should be more clearly written to reflect the current behaviour.

#4 - 04/11/2016 04:57 AM - ko1 (Koichi Sasada)

Actually, I'm not sure the policy of path representation.
For example, we can normalize every path entities with absolute path.

I'll ask Matz at next dev meeting (next Wed 13:00-, JST).

#5 - 11/05/2016 04:14 PM - ko1 (Koichi Sasada)

Sorry for late response.

I'll ask matz again.
I and Nobu talked about this topic and we agree with:

(1) to be obsolete absolute_path method and alias with path method.
(2) path method returns absolute path, even if main script (which is specified for ruby command).

Disadvantage is backtrace will be long for main script.

#6 - 11/05/2016 04:16 PM - matz (Yukihiro Matsumoto)

Agreed.

Matz.

#7 - 05/30/2017 06:10 AM - ko1 (Koichi Sasada)

After consideration, I changed my proposal.

- Rename #absolute_path to #real_path (or #realpath) and make #aboluste_path as alias of #real_path.
- change absolute_path (real_path) on eval with given file name.

summary of current behavior

On MRI, "path" is used several ways.

- __FILE__
- caller(_locations), backtrace
- requre_relative (base directory)
- $0

And ISeq has path and absolute_path. I use this terminology.

- path: given path.
- absolute_path: realpath(path) if path is exist. If not, it is nil.

Above usages are implemented with path and absolute_path.

- __FILE__ # path
- caller(_locations), backtrace # path
- requre_relative (base directory) # absolute_path
- $0 # path

Most of case, path and absolute_path is same. However, the following case they are not same.

- path and realpath(path) is different because of symlink (absolute_path is realpath).
- script name is given by command parameter (ruby x.rb) (absolute_path will be /path/to/x.rb).
- eval() without file name (path will be "(eval)" and absolute_path will be nil).

Note that eval(script, binding, "x.rb") makes path and absolute_path return "x.rb" even if given file name is not realpath.

```
eval('caller_locations(0, 1).each{|e| p [e.path, e.absolute_path]}')
eval('caller_locations(0, 1).each{|e| p [e.path, e.absolute_path]}', binding, 'x.rb')
["(eval)", nil]
["x.rb", "x.rb"]
```

proposal

Checking current behavior, #absolute_path is used as realpath (check the existing and resolve symlink). So I want to add #realpath or #real_path. I'm not sure which is better because there is File#realpath and absolute_path include _.

Also I want to check realpass for file name given at eval(). If file name is not existing, #realpath should be nil. In this case, require_relative should fail because MRI can't infer the base directory.

How about it?

#8 - 05/30/2017 09:58 AM - Eregon (Benoit Daloze)

ko1 (Koichi Sasada) wrote:

> Disadvantage is backtrace will be long for main script.

This only applies to the main script, and none of the other files so I think it's worth the gain in consistency.
Backtraces in anything but small script include more than one file anyway.

#realpath sounds OK, although it seems to mix two different things at once.
Is #realpath essentially (in terms of current methods):

```
def realpath
  absolute_path && File.realpath(absolute_path) rescue nil
end
```

If so I think it's not needed and the simpler proposal is better.

#9 - 02/25/2020 04:30 AM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

I talked about this issue with nobu and ko1.

> Is it intentional? or a bug?

As @usa (Usaku NAKAMURA) said, it is intentional according to nobu and ko1. Please use #absolute_path.

> If so, I think it's better to write "Returns the file name or absolute path of this frame" in document.

It may return relative path, too. IMO, "file name" includes relative and absolute paths, so I don't think we need to change anything.

ko1 said he withdraws his proposal. So closing this ticket.