Ruby - Bug #19985

Confusing error message when nonexistent `Pathname` for `require`

11/02/2023 09:30 AM - vo.x (Vit Ondruch)

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

Description

It seems that RubyGems / Bundler require method overrides of accept Pathname as and argument

```
$ ruby -rpathname -e '
pa = Pathname.new("test")
require pa
'
<internal:/usr/share/rubygems/rubygems/core_ext/kernel_require.rb>:85:in `require': cannot load su
ch file -- test (LoadError)
 from <internal:/usr/share/rubygems/rubygems/core_ext/kernel_require.rb>:85:in `require'
 from -e:3:in `<main>'
```

while plain Ruby require does not:

```
$ ruby --disable-gems -rpathname -e '
pa = Pathname.new("test")
require pa
'
-e:3:in `require': no implicit conversion of Pathname into String (TypeError)
 from -e:3:in `<main>'
```

This inconsistency is surprising. It seems that RubyGems / Bundler developers think [1] that Ruby require should also accept Pathname.

```
$ ruby -v
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]
```

Associated revisions

Revision 4329554f171fdb483cafa672df5f2a08741940c5 - 11/06/2023 08:58 AM - nobu (Nobuyoshi Nakada)

[Bug #19985] Raise LoadError with the converted feature name

Kernel#require converts feature name objects that have the to_path
method such as Pathname, but had used the original object on error
and had resulted in an unexpected TypeError.

Revision 4329554f171fdb483cafa672df5f2a08741940c5 - 11/06/2023 08:58 AM - nobu (Nobuyoshi Nakada)

[Bug #19985] Raise LoadError with the converted feature name

Kernel#require converts feature name objects that have the to_path
method such as Pathname, but had used the original object on error
and had resulted in an unexpected TypeError.

Revision 4329554f - 11/06/2023 08:58 AM - nobu (Nobuyoshi Nakada)

[Bug #19985] Raise LoadError with the converted feature name

Kernel#require converts feature name objects that have the to_path
method such as Pathname, but had used the original object on error
and had resulted in an unexpected TypeError.

Revision 881088e06f092d20a361c9528b2927cdc2b1616c - 11/06/2023 11:22 AM - U.Nakamura

merge revision(s) 4329554f171fdb483cafa672df5f2a08741940c5: [Backport #19985]

```
    [Bug #19985] Raise LoadError with the converted feature name

    `Kernel#require` converts feature name objects that have the `to_path`
    method such as `Pathname`, but had used the original object on error
    and had resulted in an unexpected `TypeError`.
    ---
     load.c                     | 14 ++++++++++---
     test/ruby/test_require.rb | 26 ++++++++++++++++++++++-----
     2 files changed, 32 insertions(+), 8 deletions(-)
```

Revision 881088e06f092d20a361c9528b2927cdc2b1616c - 11/06/2023 11:22 AM - U.Nakamura

merge revision(s) 4329554f171fdb483cafa672df5f2a08741940c5: [Backport #19985]

```
    [Bug #19985] Raise LoadError with the converted feature name

    `Kernel#require` converts feature name objects that have the `to_path`
    method such as `Pathname`, but had used the original object on error
    and had resulted in an unexpected `TypeError`.
    ---
     load.c                     | 14 ++++++++++---
     test/ruby/test_require.rb | 26 ++++++++++++++++++++++-----
     2 files changed, 32 insertions(+), 8 deletions(-)
```

Revision 881088e0 - 11/06/2023 11:22 AM - U.Nakamura

merge revision(s) 4329554f171fdb483cafa672df5f2a08741940c5: [Backport #19985]

```
    [Bug #19985] Raise LoadError with the converted feature name

    `Kernel#require` converts feature name objects that have the `to_path`
    method such as `Pathname`, but had used the original object on error
    and had resulted in an unexpected `TypeError`.
    ---
     load.c                     | 14 ++++++++++---
     test/ruby/test_require.rb | 26 ++++++++++++++++++++++-----
     2 files changed, 32 insertions(+), 8 deletions(-)
```

Revision 2aaa9af75989bb0993a44e9690ed2ca890b2ff91 - 11/09/2023 08:36 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 4329554f171fdb483cafa672df5f2a08741940c5,b5c74d548872388921402ff2db36be15e924a89b: [Backport #19985]

```
    [Bug #19985] Raise LoadError with the converted feature name

    `Kernel#require` converts feature name objects that have the `to_path`
    method such as `Pathname`, but had used the original object on error
    and had resulted in an unexpected `TypeError`.
    ---
     load.c                     | 14 ++++++++++---
     test/ruby/test_require.rb | 26 ++++++++++++++++++++++-----
     2 files changed, 32 insertions(+), 8 deletions(-)

    Ease the `Encoding::CompatibilityError` test failure

    At the time this test first started using `assert_raise_with_message`,
    it did not touch `Encoding.default_internal`.
    ---
     test/ruby/test_require.rb | 3 ++-
     1 file changed, 2 insertions(+), 1 deletion(-)
```

Revision 2aaa9af75989bb0993a44e9690ed2ca890b2ff91 - 11/09/2023 08:36 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 4329554f171fdb483cafa672df5f2a08741940c5,b5c74d548872388921402ff2db36be15e924a89b: [Backport #19985]

```
    [Bug #19985] Raise LoadError with the converted feature name

    `Kernel#require` converts feature name objects that have the `to_path`
    method such as `Pathname`, but had used the original object on error
    and had resulted in an unexpected `TypeError`.
    ---
     load.c                     | 14 ++++++++++---
     test/ruby/test_require.rb | 26 ++++++++++++++++++++++-----
     2 files changed, 32 insertions(+), 8 deletions(-)

    Ease the `Encoding::CompatibilityError` test failure
```

```
    At the time this test first started using `assert_raise_with_message`,
    it did not touch `Encoding.default_internal`.
    ---
     test/ruby/test_require.rb | 3 ++-
     1 file changed, 2 insertions(+), 1 deletion(-)
```

Revision 2aaa9af7 - 11/09/2023 08:36 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 4329554f171fdb483cafa672df5f2a08741940c5,b5c74d548872388921402ff2db36be15e924a89b: [Backport #19985]

```
    [Bug #19985] Raise LoadError with the converted feature name

    `Kernel#require` converts feature name objects that have the `to_path`
    method such as `Pathname`, but had used the original object on error
    and had resulted in an unexpected `TypeError`.
    ---
     load.c                    | 14 ++++++++++---
     test/ruby/test_require.rb | 26 ++++++++++++++++++++++++++-----
     2 files changed, 32 insertions(+), 8 deletions(-)

    Ease the `Encoding::CompatibilityError` test failure

    At the time this test first started using `assert_raise_with_message`,
    it did not touch `Encoding.default_internal`.
    ---
     test/ruby/test_require.rb | 3 ++-
     1 file changed, 2 insertions(+), 1 deletion(-)
```

History

#1 - 11/02/2023 10:54 AM - zverok (Victor Shepelev)

Actually, it seems kinda weird that Pathname doesn't have an implicit conversion to a String (#to_str).
OTOH, Pathname#to_path docs seem to claim that having #to_path is an agreement to represent "path-like" things, which, say File.open respects, but bare require ignores.

Rubygems respect the agreement through calling File.path. The funny thing though is there is no place where the agreement is documented! Even File.path which implements it, never documents how exactly it evaluates "string representation of the path" :)

#2 - 11/02/2023 11:00 AM - zverok (Victor Shepelev)

Even funnier that bare load does support the convention:

```
$ ruby --disable-gems -r pathname -e "load Pathname.new('test')"
-e:1:in `load': cannot load such file -- test (LoadError)
```

#3 - 11/02/2023 02:37 PM - jeremyevans0 (Jeremy Evans)

zverok (Victor Shepelev) wrote in #note-2:

> Even funnier that bare load does support the convention:
>
> ```
> $ ruby --disable-gems -r pathname -e "load Pathname.new('test')"
> -e:1:in `load': cannot load such file -- test (LoadError)
> ```

That's not funny, it's expected, as load deals exclusively with paths, and require does not generally deal with paths. Example:

```
File.write("x.foo", "p 1")
load("x.foo") # prints 1
load("./x.foo") # prints 1
require("x.foo") # LoadError
require("./x.foo") # LoadError
```

Now, if you have a path ending in .rb/.so/etc., you can get require to accept it directly. But in general, require does not deal with paths. I don't think we should add Pathname support to require.

#4 - 11/02/2023 03:44 PM - zverok (Victor Shepelev)

> That's not funny, it's expected, as load deals exclusively with paths, and require does not generally deal with paths.

Yeah, makes sense.

> But in general, require does not deal with paths. I don't think we should add Pathname support to require.

Makes sense. Then, probably, it is up to RubyGems to remove this false conversion.

#5 - 11/03/2023 05:19 AM - martinemde (Martin Emde)

It is funny to me that you would say it's not a path just because not all paths are valid. What else could it possibly be?

Ruby docs for require:

> If the filename neither resolves to an absolute path nor starts with ‘./’ or ‘../’, the file will be searched for in the library directories listed in $LOAD_PATH ($:). If the filename starts with ‘./’ or ‘../’, resolution is based on Dir.pwd.

https://ruby-doc.org/3.2.2/Kernel.html#method-i-require

Walks like a duck, quacks like a duck, looks like a duck, referred to as a duck, lays duck eggs, but certain duck species not allowed => not duck.

Now I'm not saying require should behave differently, your examples should continue to LoadError. It's just that the string "./x.foo" is a path. Pathname is a string manipulation class for dealing with these special strings that we separate with slashes and give special meaning to dots. That's a Pathname.

#6 - 11/03/2023 05:44 AM - jeremyevans0 (Jeremy Evans)

martinemde (Martin Emde) wrote in #note-5:

> It is funny to me that you would say it's not a path just because not all paths are valid. What else could it possibly be?

What the argument to require actually is is a feature name. That's why the argument to the method in the call-seq is name (not filename or path), and the first line of the require documentation mentions that it returns true or false depending on whether "the *feature* is already loaded".

> Ruby docs for require:
>
> > If the filename neither resolves to an absolute path nor starts with ‘./’ or ‘../’, the file will be searched for in the library directories listed in $LOAD_PATH ($:). If the filename starts with ‘./’ or ‘../’, resolution is based on Dir.pwd.
>
> https://ruby-doc.org/3.2.2/Kernel.html#method-i-require

Certainly the documentation for the require could be improved, because it implies that a filename that resolves to a path will be used, which is not generally the case as I showed previously. It's only the case for absolute paths or paths starting with ./ or ../, and only if the path ends with a recognized file extension.

In Ruby code, all valid load calls use paths. 99.9%+ of valid require calls do not use paths. I would guess that the majority of times that an argument starting with /, ./, or ../ is passed to require, it probably still leaves off the file extension.

> Walks like a duck, quacks like a duck, looks like a duck, referred to as a duck, lays duck eggs, but certain duck species not allowed => not duck.

It's a platypus, you shouldn't call it a duck just because it has a bill. :)

#7 - 11/03/2023 11:50 AM - vo.x (Vit Ondruch)

Should I go back to RubyGems / Bundler and ask them to "break" my first example? Or is there any reason why RubyGems / Bundler should behave differently?

#8 - 11/04/2023 04:04 PM - martinemde (Martin Emde)

It seems like you're suggesting that required features don't use a path name based hierarchy for organization.

I'm guessing you're thinking of pathnames as something strictly tied to the filesystem, but Pathname is a useful class for manipulating any string that follows the slash separated, tree based navigation. We usually think of these as files on a file system, but ruby features are organized this way too and even if we broke them free of the file system, we'd still use this hierarchy. This is how you require specific subsets of features from Active Support, or how you require a feature that is not in load path by referring to where it can be found with relative pathname conventions that translate to filesystem locations.

Paths, and therefore Pathnames, show how to get somewhere. You could use them for url paths, cache keys, filesystems or ruby features. The fact is that we use paths for ruby features. That string, no matter what you call it, is a path even if it is not a strict filesystem filename.

Almost everyone thinks it's a duck (pathname) and rubygems has been accepting ducks for years for 99% of Rubyists. In this case things that make it unique are so slight that most people are more confused that it doesn't work than enlightened when they learn it's not supposed to work (for some reason). This is not following the principal of least surprise.

#9 - 11/04/2023 05:01 PM - vo.x (Vit Ondruch)

martinemde (Martin Emde) wrote in #note-8:

> rubygems has been accepting ducks for years for 99% of Rubyists.

During the years, I had already enough fights where Ruby upstream claimed that RubyGems are integral parts of Ruby and using something like --disable-gems is a sin. Looking at the issue from this perspective, the only logical conclusion is that Ruby should accept what RubyGems does (what probably really covers more then 99% of use cases) and let require accept Pathname.

Nevertheless, I don't really have a preference, I am looking more for consistency.

#10 - 11/06/2023 07:35 AM - nobu (Nobuyoshi Nakada)

- Subject changed from Support `Pathname` for `require` to Confusing error message when nonexistent `Pathname` for `require`

- Backport set to 3.0: REQUIRED, 3.1: REQUIRED, 3.2: REQUIRED

- Tracker changed from Feature to Bug

#11 - 11/06/2023 09:07 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset git|4329554f171fdb483cafa672df5f2a08741940c5.

[Bug #19985] Raise LoadError with the converted feature name

Kernel#require converts feature name objects that have the to_path
method such as Pathname, but had used the original object on error
and had resulted in an unexpected TypeError.

#12 - 11/06/2023 09:15 AM - mame (Yusuke Endoh)

A little explanation.

Actually, the builtin Kernel#require has accepted a Pathname object by calling #to_path.

```
$ cat test.rb
puts "Hello"

$ ruby -v --disable-gems -rpathname -e 'require Pathname("./test")'
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]
Hello
```

However, if the file did not exist, it would unintentionally result in a TypeError.

```
$ ruby -v --disable-gems -rpathname -e 'require Pathname("./test2")'
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]
-e:1:in `require': no implicit conversion of Pathname into String (TypeError)
        from -e:1:in `<main>'
```

Therefore, nobu has fixed the bug by 4329554f171fdb483cafa672df5f2a08741940c5.

```
$ ./local/bin/ruby -v --disable-gems -rpathname -e 'require Pathname("./test")'
ruby 3.3.0dev (2023-11-06T08:58:47Z master 4329554f17) [x86_64-linux]
Hello

$ ./local/bin/ruby -v --disable-gems -rpathname -e 'require Pathname("./test2")'
ruby 3.3.0dev (2023-11-06T08:58:47Z master 4329554f17) [x86_64-linux]
-e:1:in `require': cannot load such file -- ./test2 (LoadError)
        from -e:1:in `<main>'
```

#13 - 11/06/2023 11:22 AM - usa (Usaku NAKAMURA)

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

ruby_3_1 881088e06f092d20a361c9528b2927cdc2b1616c merged revision(s) 4329554f171fdb483cafa672df5f2a08741940c5.

#14 - 11/07/2023 08:09 AM - usa (Usaku NAKAMURA)

note: also needs backporting b5c74d54887

#15 - 11/09/2023 08:36 AM - nagachika (Tomoyuki Chikanaga)

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

ruby_3_2 2aaa9af75989bb0993a44e9690ed2ca890b2ff91 merged revision(s)
4329554f171fdb483cafa672df5f2a08741940c5,b5c74d548872388921402ff2db36be15e924a89b.