Ruby master - Feature #4969

Subtle issue with require

07/04/2011 04:19 AM - trans (Thomas Sawyer)

Status:	Rejected
Priority:	Normal
Assignee:	
Target version:	

Description

If I have a library with same name as Ruby standard library in load path (as an example):

lib/abbrev.rb

There is conflict with loading. Ok, I work around:

require 'rbconfig'

# Notice that rubylibdir takes precendence.
LOCATIONS = ::RbConfig::CONFIG.values_at(
'rubylibdir', 'archdir', 'sitelibdir', 'sitearchdir'
)

#
def require_ruby(file)
LOCATIONS.each do |loadpath|
if path = lib_find(loadpath, file)
return path
end
end

```
raise LoadError, "no such file to load -- #{fname}"
```

end

private

```
SUFFIXES = ['.rb', '.rbw', '.so', '.bundle', '.dll', '.sl', '.jar']

# Given a +loadpath+, a file's relative path, +relname+, and
# options hash, determine a matching file exists. Unless +:load+
# option is +true+, this will check for each viable Ruby suffix.
# If a match is found the full path to the file is returned,
# otherwise +nil+.
def lib_find(loadpath, relname)
  if SUFFIXES.include?(File.extname(relname))
    abspath = File.join(loadpath, relname)
    File.exist?(abspath) ? abspath : nil
  else
    SUFFIXES.each do |ext|
      abspath = File.join(loadpath, relname + ext)
      return abspath if File.exist?(abspath)
    end
  end
  nil
end
```

Now I can do:

require 'abbrev'
require_ruby 'abbrev'

And it works fine. But, if I do:

require_ruby 'abbrev'
require 'abbrev'

The second is not loaded because somehow it seems to confuse it for the first in $LOADED_FEATURES.

I realize this is a very subtle issue and not likely to effect most people, but it presents a big problem for some of my work (e.g. wedge gem)

How is Ruby confusing the two? Can it be fixed? Or is there at least a work around?

History

#1 - 07/04/2011 04:21 AM - trans (Thomas Sawyer)

Note: ignore the lib_find documentation about the load option, I removed the option hash to simplify this example.

#2 - 07/06/2011 05:59 AM - tenderlovemaking (Aaron Patterson)

- ruby -v changed from ruby 1.9.3dev (2011-07-03 trunk 32372) [x86_64-linux] to -

On Mon, Jul 04, 2011 at 04:19:42AM +0900, Thomas Sawyer wrote:

[snip]

> How is Ruby confusing the two? Can it be fixed? Or is there at least a work around?

I'm not sure about the code you presented above, but you should be able
to ensure your gem is consulted before stdlib by doing "gem 'whatever'".

Possibly you could just do:

gem 'wedge'
require 'abbrev'

Though, I would just avoid conflicting filenames. Maybe have a
'wedge/abbrev'.

--
Aaron Patterson
http://tenderlovemaking.com/

#3 - 07/07/2011 04:36 AM - trans (Thomas Sawyer)

@Aaron Yea, the problem isn't with loading a file *of* wedge. It has to do with what wedge does. The code I presented is a slightly simplified "wedge" in the project itself. The wedge gem is a lot like polyglot, but works a bit differently. And was originally created to handle the issue of loading ruby/gem files while by-passing any possible name conflicts. Of course it can be use for other things too, but that's what my current use case is.

I am going to do some more in-depth research on this so hopefully I can come back with more specific details on what's causing the problem. Please let me know if you have any ideas. Thanks.

#4 - 07/25/2011 07:55 PM - naruse (Yui NARUSE)

- File deleted (noname)

#5 - 07/25/2011 07:57 PM - naruse (Yui NARUSE)

- Status changed from Open to Feedback

#6 - 07/26/2011 10:10 PM - nobu (Nobuyoshi Nakada)

Your require_ruby seems to load nothing. Missed to paste?

Anyway, the behavior does not seem a bug.
It might be a feature request, though I'm not sure what you are suggesting.

#7 - 10/16/2011 01:36 AM - trans (Thomas Sawyer)

Okay, I finally got around to digging into this a bit more. The issue can be seen from this simple example.

Given a local directory containing:

fixture/
abbrev.rb

Then:

$ irb

```
$LOAD_PATH.unshift('./fixture')
=> ["./fixture", "/usr/local/lib/ruby/gems/1.9.1/gems/wirble-0.1.3/lib", "/usr/local/lib/ruby/site_ruby/1.9.1",
"/usr/local/lib/ruby/site_ruby/1.9.1/x86_64-linux", "/usr/local/lib/ruby/site_ruby", "/usr/local/lib/ruby/vendor_ruby/1.9.1",
"/usr/local/lib/ruby/vendor_ruby/1.9.1/x86_64-linux", "/usr/local/lib/ruby/vendor_ruby", "/usr/local/lib/ruby/1.9.1",
"/usr/local/lib/ruby/1.9.1/x86_64-linux"]
require '/usr/local/lib/ruby/1.9.1/abbrev.rb'
=> true
Abbrev
=> Abbrev
require 'abbrev'
=> false
```

Even though we loaded the standard abbrev.rb file using an absolute path, Ruby thinks that the subsequent require is for the same file. But it is not b/c the files in the 'fixture' directory should be taking precedence over the other location since it is earlier in the $LOAD_PATH.

#8 - 10/16/2011 02:23 AM - jonforums (Jon Forums)

what happens when you put $LOADED_FEATURES.reject! { |i| i =~/abbrev/ } before the last require

#9 - 10/16/2011 04:06 AM - trans (Thomas Sawyer)

Then it works.

```
$LOADED_FEATURES.reject! { |i| i =~/abbrev/ }
require 'abbrev'
true
```

#10 - 10/20/2011 08:40 PM - trans (Thomas Sawyer)

I've gone ahead and released the *loadable* gem. You can read about it at http://github.com/rubyworks/loadable. I know there has been some talk here about creating a more flexible load path. loadable works by adding load hook objects to the $LOADERS global variable. Load hooks are any object that responds to #call, which loads/requires the file, and #each, that iterates over all loadable files. It also extends #require and #load to accept an options hash to allow loader configurations.

The Ruby Loader that it comes with can be used to isolate loading from Ruby's standard library. But this issue (#4969) prevents it form fully working.

So, what's the word on this?

#11 - 11/01/2011 02:24 PM - funny_falcon (Yura Sokolov)

Why not put abbrev.rb into lib/wedge, and then call require 'wedge/abbrev' ? I thought it is standard way.

#12 - 11/01/2011 02:55 PM - trans (Thomas Sawyer)

hi, abbrev.rb is not part of wedge. I just used 'abbrev.rb' as an easy to understand example of the potential problem. Wedge (which has been renamed to 'Loadable' in the latest release), has a "load wedge" that makes it possible to circumvents any possible name clashes. Now it would be nice if everyone followed proper practice and thus avoided all possible clashes, but that is often not the case, like it or not. This can be easily seen from this very partial list, http://github.com/rubyworks/loadable/blob/master/INFRACTIONS.md

Even so, the issue I've run into isn't this per se.* I wrote Loadable to deal with it. The problem is that I can't make Loadable work as it should b/c of the way in which Ruby is handling feature caching. See point #7 of this discussion for what I mean. Thanks.

*Though IMO it would be a good idea for Ruby to deal with this.

#13 - 04/13/2012 09:25 AM - nobu (Nobuyoshi Nakada)

- Tracker changed from Bug to Feature

- Status changed from Feedback to Rejected

If you still need this, please make the issue clear and refine the proposal, then reopen this or file a new ticket.

#14 - 12/22/2012 01:45 AM - trans (Thomas Sawyer)

I just tried this out on Ruby v1.9.3-p327. And it seems to have been fixed! Yea!

So you can change the status of this from Rejected to Closed (if you'd like to be precise).

Thanks!