Ruby master - Feature #6507

File Literal

05/28/2012 02:03 AM - trans (Thomas Sawyer)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

=begin
One of the features of the Rebol programming language that I has always liked is its direct support for files via the (({%filename})) notation (See http://www.rebol.com/r3/docs/datatypes/file.html). I've often wondered how Ruby might support the same, but finding a suitable and available notation proved rather difficult.

Today it occurred to me that perhaps the / symbol could do the trick:

file = /README.rdoc

For absolute paths it could be //:

file = //etc/fstab

Exactly what class of object (({file})) should be is up for debate. Probably it would be a (({Pathname})) instance, but I suppose it could a different "Path" class basically a wrapper round (({File})) and (({Dir})) classes.

The benefit of this is fairly obvious I think, but I'll give one clear usecase just the same:

```
class Foo
def initialize(source)
case source
when String
parse(source)
when Pathname  # assuming this to be the instance
parse(source.read)
end
end
end

# from string
Foo.new "content of foo"

# from file
Foo.new /foo.txt
```

There is the ambiguity of x /a for division, but I think expecting x/a or x / a for that is okay. After all, the same limitation holds for other unary operators too.

Actually, while I like the concise notation, it may be more flexible to require a string:

/'foo.txt'

Then / could actually be a unary operator.

In anycase, whether this notation works or not, I hope this spurs some debate so that ultimately something along these lines will come of it. I truly tire of typing things like (({File.read(File.join(File.dirname(**FILE**), fname))})).
=end

History

#1 - 05/28/2012 02:39 AM - mame (Yusuke Endoh)

- Status changed from Open to Feedback

trans (Thomas Sawyer) wrote:

> Today it occurred to me that perhaps the / symbol could do the trick:
>
> file = /README.rdoc

Consider a file named "i" in a directory named "foo":

regexp = /foo/i

> In anycase, whether this notation works or not, I hope this spurs some debate so that ultimately something along these lines will come of it. I truly tire of typing things like (({File.read(File.join(File.dirname(**FILE**), fname))})).

I understand your motivation completely.
But please propose a possible concrete syntax first.

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 05/28/2012 03:15 AM - trans (Thomas Sawyer)

Ah frigets! I totally spaced on regexp notation. Well maybe the string would have to be used:

file = /'foo'/'i'

But I can see how that could still be confusing with regex.

#3 - 05/28/2012 09:23 PM - prijutme4ty (Ilya Vorontsov)

May be %f{foo.bar}.open{}, %f{file*.txt}.each{}

Syntax with open bracket and without closing makes trouble when method call expected: \foo.rb.open{}

#4 - 05/28/2012 09:49 PM - trans (Thomas Sawyer)

=begin
I thought about this a bit more and there is really no way to avoid a certain amount of regex look-alike b/c that's simply how paths are notated -- with slashes.

One thought I had was using ./ and // (for root) as initial markers.

file = //'etc/fstab'

file = ./'README.rdoc'

ilya (Ilya Boltnev) You are right, but parenthesis could be used if need be. Given your example:

(\foo.rb).open{}

In any case we will want to support variables so it's probably not a good option for that reason either, e.g. we would want to do things like:

dir = File.dirname(**FILE**)

//dir/'foo.txt'

The use of (({%f})) would work. The only shortcoming there is having to use interpolation for variables, e.g. the example above would be:

%f{#{dir}/foo.txt}

Not quite as nice. But, I'd still take that over what we have now.
=end

#5 - 10/25/2012 05:50 PM - yhara (Yutaka HARA)

- Target version changed from 2.0.0 to 3.0

#6 - 10/25/2012 05:54 PM - yhara (Yutaka HARA)

- Target version changed from 3.0 to 2.6

#7 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)