Ruby master - Feature #15861

Correctly parse `file:c:/path/to/file` URIs

05/18/2019 03:21 PM - deivid (David Rodríguez)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

Recently ruby has getting better at parsing URIs using the "file" scheme, with the addition of "URI::File". Still, some Windows edge cases are not implemented, and it would be nice to have them. For example, while the addressable gem can correct parse "file:c:/path/to/file", the builtin library is not that smart yet:

```
irb(main):001:0> URI.parse("file:c:/path/to/file").path
=> nil
irb(main):002:0> require 'addressable'
=> true
irb(main):003:0> Addressable::URI.parse("file:c:/path/to/file").path
=> "c:/path/to/file"
```

I think this would be a matter of implementing https://tools.ietf.org/html/rfc8089#appendix-E.2, which is not normative but it would be certainly nice to have.

History

#1 - 05/18/2019 03:37 PM - jeremyevans0 (Jeremy Evans)

Looking at RFC 8089, Appendix E (Nonstandard Syntax Variations) states:

```
   These variations may be encountered by existing usages of the file
   URI scheme but are not supported by the normative syntax of
   Section 2.

   This appendix is not normative.
```

Appendix E.2 says that the following form is already supported: file:///c:/path/to/file. Using that with the URI library:

```
URI.parse("file:///c:/path/to/file").path
=> "/c:/path/to/file"
```

I can see where the preceding / may present an issue. However, I am not sure whether we want to add support for nonstandard syntax variations to the file scheme.

#2 - 05/18/2019 04:02 PM - deivid (David Rodríguez)

Yep, that's correct.

The alternative is to use the alternative form, but as you point out, the preceding slash is certainly inconvenient.

Anyways, thanks for considering this.

#3 - 05/18/2019 04:05 PM - deivid (David Rodríguez)

This is definitely intentional behavior, by the way. This is how the current tests look:

```
  u = URI("file:///c:/path/to/file")
  assert_equal "/c:/path/to/file", u.path

  # this form is not supported
  u = URI("file:c:/path/to/file")
  assert_equal "c:/path/to/file", u.opaque
```

I just wished working with file URIs on Windows was a bit easier.

#4 - 05/18/2019 10:58 PM - phluid61 (Matthew Kerwin)

jeremyevans0 (Jeremy Evans) wrote:

> Looking at RFC 8089, Appendix E (Nonstandard Syntax Variations) states:
>
> ```
>    These variations may be encountered by existing usages of the file
>    URI scheme but are not supported by the normative syntax of
>    Section 2.
>
>    This appendix is not normative.
> ```

"may be encountered" here is meant to mean that old "poorly formed" URIs exist in the wild, and occasionally we encounter them. For example, if I were to write a Ruby script to update a bunch of historical documentation from 1997. It can also mean the kind of stuff humans input, which "worked for me when I did it in Internet Explorer".

> Appendix E.2 says that the following form is already supported: file:///c:/path/to/file. Using that with the URI library:
>
> ```
> URI.parse("file:///c:/path/to/file").path
> => "/c:/path/to/file"
> ```

In other words, implement a pre-parser that detects E.2-flavoured URIs and inserts the extra slash. FWIW there's actually a gem out there that does essentially this, however since it was written and published alongside RFC 8089 it predates the core File URI handling.

```
require 'file-uri/win'
URI.parse("file:c:/path/to/file").path
=> "/c:/path/to/file"
```

> I can see where the preceding / may present an issue. However, I am not sure whether we want to add support for nonstandard syntax variations to the file scheme.

It's a pain, but it's a pain we've lived with forever. FWIW, I added a #to_file_path method in that gem to see if it makes this particular paint point any easier. But I don't think this is the major driver, more it's having to detect and correct those other URIs.

All said; if all else fails there's always Addressable.