Ruby master - Feature #16978

Ruby should not use realpath for __FILE__

06/23/2020 07:29 AM - vo.x (Vit Ondruch)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

This is the simplest test case:

```
$ mkdir a

$ echo "puts __FILE__" > a/test.rb

$ ln -s a b

$ ruby -Ib -e "require 'test'"
/builddir/a/test.rb
```

This behavior is problematic, because Ruby should not know nothing about the a directory. It was not instructed to use it. I should always refer to the file using the original path and do not dig into the underlying details, otherwise depending on file system setup, one might be forced to used File.realpath everywhere trying to use __FILE__.

History

#1 - 07/09/2020 09:10 PM - jeremyevans0 (Jeremy Evans)

- Backport deleted (2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN)

- ruby -v deleted (ruby 2.7.1p83 (2020-03-31 revision a0c7c23c9c) [x86_64-linux])

- Tracker changed from Bug to Feature

I don't think this is a bug. __FILE__ is documented as follows: The path to the current file. Which path (real, absolute, relative, expanded) is not specified.

Not using the real path would lead to behavior that depends on the first path used when requiring the file.

a/test.rb (b symlinked to a):

```
def a
  __FILE__
end

ruby -Ia -Ib -rtest -e 'a'
# /path/to/a/test.rb
ruby -Ib -Ia -rtest -e 'a'
# Current: /path/to/a/test.rb
# Your proposed: /path/to/b/test.rb
```

What actually happens is not the file path being converted to a real path, but the include directory being converted to a real path before the file is required (in rb_construct_expanded_load_path). Changing this to not use a real path would probably break the code that checks that a feature hasn't been require twice. For example, this code would change behavior:

```
$: << 'a'
require 'test'
$: << 'b'
require 'test'
# Current: not loaded again
# Your proposed: loaded again
```

If you symlink the file itself and not the include directory, Ruby will attempt to require it as a separate feature.

Note that if you provide a path when requiring, Ruby already operates the way you want:

```
ruby -r./a/test -e 'a' # /path/to/a/test.rb
ruby -r./b/test -e 'a' # /path/to/b/test.rb
```

I can certainly see pros and cons from changing the behavior, but I would consider this a feature request and not a bug.

#2 - 07/09/2020 09:18 PM - Eregon (Benoit Daloze)

The "main file" (the file passed to ruby myfile.rb is also special in that __FILE__ and $0 can both be relative paths (basically the same path as passed on the command line).

#3 - 07/09/2020 09:47 PM - vo.x (Vit Ondruch)

> this code would change behavior:

Absolutely, different $LOAD_PATH must result in different behavior.

```
$: << 'a'
require 'test'
$: << 'b'
require 'test'
# Current: not loaded again
# Your proposed: loaded again
```

This is actually where I am coming from. I am not 100 % sure what is the spec of require, but I would expect that require 'test' works just once and it does not matter what is the current $LOAD_PATH status. So my proposal is actually:

```
# Your proposed: not loaded again
```