Ruby - Bug #14891

Pathname#join has different behaviour to File.join

07/03/2018 02:37 AM - robotdana (Dana Sherson)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.6.0-preview2, and before	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

```
Pathname.new('/a').join('/b').to_s
# => "/b"
File.join(Pathname.new('/a'), '/b').to_s
# => "/a/b"
```

in my case '/b' was in a variable and it wasn't immediately obvious why it wasn't working when I moved to use Pathname

This seems to not be desired behaviour as it's different to File.join, and this case isn't document anywhere.

Can we either change the behaviour to treat the "other" of Pathname#+ as always relative (possibly just removing a leading slash), or add this case to the documentation?

History

#1 - 07/03/2018 02:38 AM - robotdana (Dana Sherson)

- ruby -v set to 2.6.0-preview2, and before

#2 - 07/03/2018 07:24 AM - shevegen (Robert A. Heiler)

This behaviour surprised me too. Is it expected that the information from "/a"
is lost? I have no idea but the documentation does not mention this; perhaps
the above could be added as an example OR the behaviour changed (or both).

I have little to none experience with pathname these days as I seem to handle
paths ... without pathname. :D

Documentation to Pathname .join is at:

https://ruby-doc.org/stdlib-2.5.1/libdoc/pathname/rdoc/Pathname.html#method-i-join

#3 - 07/03/2018 08:05 AM - Hanmac (Hans Mackowiak)

More examples:

```
Pathname.new('/a').join('c', 'b').to_s #=> "/a/c/b"
Pathname.new('/a').join('/c', 'b').to_s #=> "/c/b"
Pathname.new('/a').join('/c', '/b').to_s #=> "/b"
```

Why it does this?
because "/c" means start of an absolute path there

#4 - 07/09/2018 12:05 PM - znz (Kazuhiro NISHIYAMA)

I think it is expected behavior.

https://github.com/ruby/ruby/blob/38e05ff3e194268fd2f38ac7c9530298e464f07b/test/pathname/test_pathname.rb#L211

```
  defassert(:plus, '/b', 'a', '/b')
```

#5 - 07/13/2018 04:58 AM - funny_falcon (Yura Sokolov)

I'd rather say that File.join is currently broken, and it should behave like Pathname.join. But probably I'm missing something.

#6 - 02/18/2020 07:58 AM - jnchito (Junichi Ito)

I am wondering about the current behavior of Pathname#join, too. Are there any useful use cases for Pathname.new('/a').join('/c', '/b').to_s #=> "/b"? I think it should be Pathname.new('/a').join('/c', '/b').to_s #=> "/a/c/b like File#join. I'd like to know the basic idea behind this design.

#7 - 02/18/2020 10:09 AM - zverok (Victor Shepelev)

> I am wondering about the current behavior of Pathname#join, too. Are there any useful use cases for Pathname.new('/a').join('/c', '/b').to_s #=> "/b"

I believe that Pathname#join acts kinda like cd in the shell: cd x is "go deeper to x level" while cd /x si "go to the root, then into /x folder".

I believe it can be useful and desired behavior when working with configs for some authoring systems and devops tools, where you take from config "do 'action' by path 'path'", and config can specify lot of actions relative to app folder (like touch: "tmp/restart.txt"), but eventually want to specify something to do in relative-from-root folder (like read: "/etc/myapp/settings.ini"). Without following the specification "/ at the beginning means go to root" it becomes rather ugly.

In other words, this:

```
# I am just trying to merge pathes explicitly, and receive "unexpected" result:
Pathname.new('foo').merge('/bar')
```

— might seem "weird", while this:

```
@app_path = Path.new(__dir__)
# ...
@app_path.join(action.target_path) # when target_path is "/foo/bar", it allows to act outside of base dir
```

— is desirable (and is NOT achievable if Pathname's behavior will be changed)

I assume that those finding the behavior less logical think about paths about "just some abstract strings" and Pathname#join as a fancy way to write Array#join(SYSTEM_PATH_DELIMITER). But Pathname tries to represent "filesystem path object" in a meaningful way, consistent with filesystem's intuitions.

#8 - 02/18/2020 01:39 PM - jnchito (Junichi Ito)

zverok (Victor Shepelev) wrote in #note-7:

> > I am wondering about the current behavior of Pathname#join, too. Are there any useful use cases for Pathname.new('/a').join('/c', '/b').to_s #=> "/b"
>
> I believe that Pathname#join acts kinda like cd in the shell: cd x is "go deeper to x level" while cd /x si "go to the root, then into /x folder".

Thank you for your explanation. It is easy to understand the behavior if I think Pathname.new('/a').join('/c', '/b') means cd /a then cd /c then cd /b. (params can be both directories and files, though.)

> I assume that those finding the behavior less logical think about paths about "just some abstract strings" and Pathname#join as a fancy way to write Array#join(SYSTEM_PATH_DELIMITER). But Pathname tries to represent "filesystem path object" in a meaningful way, consistent with filesystem's intuitions.

That is me who thought "Pathname#join as a fancy way to write Array#join(SYSTEM_PATH_DELIMITER)". However, Rails developers like me often see Rails.root.join('foo/bar.jpg') as main use case of Pathname#join. So I guess many of them might misunderstand it just joins two strings in a readable way.

In addition, it is very confusing File#join has the same name and different behavior, and File and Pathname are also close idea. Many people might misunderstand they act in the same way. So probably Pathname#join should have been given an another good name. (Pathname#merge might not be a bad idea.)

#9 - 12/07/2021 05:26 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed