# Ruby - Bug #21391

# Inconsistent trailing slash behavior of File.join and Pathname#join with empty strings

06/01/2025 09:42 AM - Iovro-bikic (Lovro Bikić)

Status:       Open         Priority:       Normal         Assignce:       Target version:         Target version:       na3831630 + PRISM [v86 64-darwin23]         Backport:       3.2: UNKNOWN, 3.3: UNKNOWN, 3.4:         Description       3.2: UNKNOWN, 3.3: UNKNOWN, 3.4:         Priority:					
<pre>Priority: Normal Assignee: Target version: Target version</pre>	Status:	Open			
Assignee: Target version: ruby v: uby 34.4 (2025-05-14 revision as89311030] +PRISM [x86_04-darwin23] Backport: 3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: UNKNOWN Description File.poin (//usr*, '') f - ~ ruar' Fal-point (//usr*, '') f - ~ ruar' Fal-point (//usr*, '') f - ~ ruar' Fal-point (//usr*, '') f - ~ ruar' File.join (//usr*, '') file.join (//usr*, '') f - Zahhannet/join adds a trailing slash. Pathnamet/join doesn't. File.join (//usr*, '') file.join (//usr*, '') f - Zahhannet/join adds a trailing slash. Pathnamet/join doesn't. file.join (//usr*, '') f - Zahhannet/join adds a trailing slash. string interpolation (e.g. in Rais, "#[Rais.root]") or File.join (e.g. File.join(Rais.root, ''). In other popular languages, both approaches have been taken: • as path.join in Python adds a trailing slash: string interpolation (e.g. in Rais, "#[Rais.root]") or File.join (e.g. File.join(Rais.root, ''). In other popular languages, both approaches have been taken: • as path.join in Python adds a trailing slash: string f os so path.join('/usr*, '') f //dar/( • Path.join(//usr*, ''); f //dar/( • path.join(//usr*, ''); path.join(//usr*, ''); path.join('/usr*, ''); path.join('/usr*, ''); path.join('/usr*, ''); // //usr' • gliasnt.Join f Cig doesnt add a trailing slash: socotage main import ('fmt*; "path/filepath*) func main() ( fm. ferinin(filepath.Join(*/usr*, **)) // / pints */usr*	Priority:	Normal			
<pre>Target version: ruby v: ruby 3.4.4 (2025-05-14 revision</pre>	Assignee:				
<pre>ruby -x: uby 3.4.4 (2025-05-14 rowing</pre>	Target version:				
<pre>Description File.join('/usr', '') # -&gt; "/usr' # no trailing slash File.join('/usr', ') # -&gt; "/usr' # no trailing slash File.join('/usr', ') # -&gt; "/usr' # Pathname.new('/usr').join('').to_s # -&gt; "/usr/ " File.join with an empty string adds a trailing slash. Pathname#join doesn't. When Pathname#join argument is a string with empty whitespace, a trailing slash is added (plus whitespace). It think it's a common use-case to append a trailing slash to Pathname, and currently you have to resort to other methods such as string interpolation (e.g. in Raik, "#(Rais.root)") or File.join (e.g. File.join(Rais.root, ")). In other popular languages, both approaches have been taken: • aspath.join in Python adds a trailing slash: import os ospath.join('/usr', '') # /'usr/' • Path.join.in Rust adds a trailing slash: use stdi:pathr:(Path); fn main() { Path.join.in Rust adds a trailing slash: use stdi:pathr:(Path); fn main() {</pre>	ruby -v:	ruby 3.4.4 (2025-05-14 revision a38531fd3f) +PRISM [x86_64-darwin23]	Backport:	3.2: UNKNOWN, 3.3: UNKNOWN, 3.4: UNKNOWN	
<pre>File.join('/usr', '') # -&gt; '/usr' File.join('/usr', '') # -&gt; '/usr' File.join('/usr', '') # -&gt; '/usr' File.join('/usr', '') # -&gt; '/usr' File.join with an empty string adds a trailing slash. Pathname#join doesn't. When Pathname.new('/usr').join('').to_s # +&gt; '/usr' File.join with an empty string adds a trailing slash. Pathname#join doesn't. When Pathname.new('/usr').join('').to_s # +&gt; '/usr' File.join with an empty string adds a trailing slash. Pathname#join doesn't. When Pathname#join argument is a string with empty whitespace, a trailing slash is added (plus whitespace). I think its a common use-case to append a trailing slash to Pathname, and currently you have to resort to other methods such as string interpolation (e.g. in Rails, '#(Rails.root)') or File.join (e.g. File.join(Rails.root, ''). In other popular languages, both approaches have been taken:</pre>	Description				
<pre>Pathname.new('/usr').join('').to_s # -&gt; "/usr" # no trailing slash File.join('/usr', ') # -&gt; "/usr' # no trailing slash File.join('/usr', ') Fathname.new('/usr').join('').to_s # -&gt; "/usr' # Fathname.new('/usr').join(e.g. File.join(Rais.root,''). In other popular languages, both approaches have been taken:     • os.path.join('/usr', '') # '/usr'!     • Path.join('/usr', '') # '/usr'!     • Path.join('/usr', '') # '/usr'!     • Path.join faust adds a trailing slash:     use std::pathr:(Path); fin main() {     println!("{'}", Path::new("/usr").join("").display());     // prints "/usr'#     • path.join in Node doesn't add a trailing slash:     const path - require('path');     path.join('/usr', '');     // '/usr'     • filepath.join in Go doesn't add a trailing slash:     upockage main     import ("fnt"; "path/filepath")     func main() {         filepath.join in Go doesn't add a trailing slash:         path.join('/usr', '');         // '/usr'         • filepath.join in Go doesn't add a trailing slash:         path.join('/usr', '');         // '/usr' </pre>	File.join('/usr', # => "/usr/"	•••)			
<pre>File.join('/usr', '') # - "/usz/ " Pathname.new('/usr').join(' ').to_s # - "/usz/ " Pathname.new('/usr').join(' ').to_s # - ''/usz/ " Filejoin with an empty string adds a trailing slash. Pathname#join doesn't. When Pathname#join argument is a string with empty whitespace, a trailing slash is added (plus whitespace). Ithik it's a common use-case to append a trailing slash to Pathname, and currently you have to resort to other methods such as string interpolation (e.g. in Rails, "#[Rails.root!/) or File.join (e.g. File.join(Rails.root, ')). In other popular languages, both approaches have been taken:     os.path.join in Python adds a trailing slash:     import os     os.path.join('/usr', '')     */'usr/"     Path.join in Bust adds a trailing slash:     use stdi:path::(Path); fn main() {     println!(")", Path::new("/usr").join("").display());     // prints "/usr/"     elath.join in Node doesn't add a trailing slash:     const path = require('path');     path.join in Go doesn't add a trailing slash:     const path = require('path');     path.join Go doesn't add a trailing slash:     const path = require('path');     path.join in Go doesn't add a trailing slash:     const path = require('path');     path.join ('/usr', '');     // '/uar'     filepath.join in Go doesn't add a trailing slash:     const path = require('path');     path.join ('/usr', '');     // '/uar'     filepath.join in Go doesn't add a trailing slash:     const path = require('path');     path.join ('/usr', '');     // '/uar'     filepath.join in Go doesn't add a trailing slash:     const path = require('path');     path.join ('/usr', '');     // '/uar'     filepath.join in Go doesn't add a trailing slash:     const path = require('path');     path.join in Go doesn't add a trailing slash:     const path = require('path');     filepath.join in Go doesn't add a trailing slash:     const path = require('path');     filepath.join in Go doesn't add a trailing slash:     const path = require('path');     filepath.join ('/usr', '');     // //uar'</pre>	Pathname.new('/usr').join('').to_s # => "/usr" # no trailing slash				
<pre>Pathname.new('/usr').join('').to_s # =&gt; "/usr/" File join with an empty string adds a trailing slash, Pathname#join doesn't. When Pathname#join argument is a string with empty whitespace, a trailing slash is added (plus whitespace). It hink it's a common use-case to append a trailing slash to Pathname, and currently you have to resort to other methods such as string interpolation (e.g. in Rails, "#(Rails.root)") or File.join (e.g. File.join(Rails.root, ")). In other popular languages, both approaches have been taken:     • os.path.join in Python adds a trailing slash:     import os     os.path.join('/usr', '')     * '/uar/'     • Path.join('/usr', '')     * '/uar/'     • Path.join f. Rust adds a trailing slash:     use std::path::(Path); fn main() {     println!("{}", Path::new("/usr").join("").display());     // prints "/usr/"     • path.join in Node doesn't add a trailing slash:     const path = require('path');     path.join('/usr', '');     // '/usr'     • filepath.Join in Go doesn't add a trailing slash:     package main     import ("fmt"; "path/filepath") fnuc main() {     filepath.Join('/usr', '');     // //usr'     • filepath.Join in Go doesn't add a trailing slash:     const path = require('path');     path.join('/usr', '');     // //usr'     • filepath.Join in Go doesn't add a trailing slash:     package main     import ("fmt"; "path/filepath") fnuc main() {     filepath.Join("/usr", "***********************************</pre>	File.join('/usr', ' ') # => "/usr/ "				
<pre>File.join with an empty string adds a trailing slash, Pathname#join doesn't. When Pathname#join argument is a string with empty whilespace, a trailing slash is added (plus whitespace). I think if's a common use-case to append a trailing slash to Pathname, and currently you have to resort to other methods such as string interpolation (e.g. in Rails, "#(Rails.root)/") or File.join (e.g. File.join(Rails.root, ")). In other popular languages, both approaches have been taken: • os.path.join in Python adds a trailing slash: import os os.path.join ('/usr', '')) # '/usr'! • Path.join in Rust adds a trailing slash: use std::path::(Path); fin main() { printIn!("{}", Path::new("/usr").join("").display()); // prints "/usr/" } • path.join in Node doesn't add a trailing slash: const path = require('path'); path.join ('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { filepath.Join in Go doesn't add a trailing slash: const path = require('path'); path.join ('/usr', ''); // '/usr' filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fit.Println((filepath.Join("/usr", "")) // prints "/usr"</pre>	Pathname.new('/usr').join(' ').to_s # => "/usr/ "				
<pre>http://time.com/common/co</pre>	File.join with an empty string adds a trailing slash, Pathname#join doesn't. When Pathname#join argument is a string with empty whitespace, a trailing slash is added (plus whitespace).				
<pre>In other popular languages, both approaches have been taken: • os path.join in Python adds a trailing slash: import os os.path.join('/usr', '') # '/usr/' • Path.join in Bust adds a trailing slash: use std::path::(Path); fn main() { println!("()", Path::new("/usr").join("").display()); // prints "/usr/" path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	I think it's a common use-case to append a trailing slash to Pathname, and currently you have to resort to other methods such as string interpolation (e.g. in Rails, "#{Rails.root}/") or File.join (e.g. File.join(Rails.root, ")).				
<pre>• os.path.join in Python adds a trailing slash: import os os.path.join('/usr', '') # '/usr/' • Path.join in Rust adds a trailing slash: use std::path::{Path}; fn main() { println!("{)", Path::new("/usr").join("").display()); // prints "/usr/" } • path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	In other popular languages, both approaches have been taken:				
<pre>import os os.path.join('/usr', '') # '/usr/' • Path.join in Rust adds a trailing slash: use std::path::(Path); fn main() { println!("{}", Path::new("/usr").join("").display()); // prints "/usr/" } • path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	• <u>os.path.join in Python</u> adds a trailing slash:				
<pre>• Path.join in Rust adds a trailing slash: use std::path::{Path}; fn main() { println!("{}", Path::new("/usr").join("").display()); // prints "/usr/" } • path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	<pre>import os os.path.join('/usr', '') # '/usr/'</pre>				
<pre>use std::path::{Path); fn main() {     println!("{}", Path::new("/usr").join("").display());     // prints "/usr/" }     • path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr'     • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() {     fmt.Println(filepath.Join("/usr", ""))     // prints "/usr"</pre>	Path.join in Rust adds a trailing slash:				
<pre>fn main() {     println!("{}", Path::new("/usr").join("").display());     // prints "/usr/" }     • path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr'     • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() {     fmt.Println(filepath.Join("/usr", ""))     // prints "/usr"</pre>	<pre>use std::path::{Path};</pre>				
<ul> <li>path.join in Node doesn't add a trailing slash:</li> <li>const path = require('path');</li> <li>path.join('/usr', '');</li> <li>// '/usr'</li> <li>filepath.Join in Go doesn't add a trailing slash:</li> <li>package main</li> <li>import ("fmt"; "path/filepath")</li> <li>func main() {     fmt.Println(filepath.Join("/usr", ""))     // prints "/usr"</li> </ul>	<pre>fn main() {     println!("{}", Path::new("/usr").join("").display());     // prints "/usr/" }</pre>				
<pre>• path.join in Node doesn't add a trailing slash: const path = require('path'); path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	·				
<pre>const path = require('path'); path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	• <u>path.join in Node</u> doesn't add a trailing slasn:				
<pre>path.join('/usr', ''); // '/usr' • filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	const path = requi	re('path');			
<pre>• filepath.Join in Go doesn't add a trailing slash: package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	path.join('/usr', ''); // '/usr'				
<pre>package main import ("fmt"; "path/filepath") func main() { fmt.Println(filepath.Join("/usr", "")) // prints "/usr"</pre>	• <u>filepath.Join in Go</u> doesn't add a trailing slash:				
<pre>import ("fmt"; "path/filepath") func main() {   fmt.Println(filepath.Join("/usr", ""))   // prints "/usr"</pre>	package main				
<pre>func main() {   fmt.Println(filepath.Join("/usr", ""))   // prints "/usr"</pre>	<pre>import ("fmt"; "path/filepath")</pre>				

### History

#### #1 - 06/01/2025 11:09 AM - lovro-bikic (Lovro Bikić)

- Subject changed from Inconsistent trailing slash behavior of File#join and Pathname#join with empty strings to Inconsistent trailing slash behavior of File.join and Pathname#join with empty strings

#### #2 - 06/01/2025 01:02 PM - Dan0042 (Daniel DeLorme)

It's not the only inconsistent behavior:

```
File.join("/usr","/var")  #=> "/usr/var"
Pathname.new("/usr").join("/var").to_s #=> "/var"
File.join("/usr","../var")  #=> "/usr/../var"
Pathname.new("/usr","/../var")  #=> "/usr/../var"
File.join("/usr","/../var")  #=> "/usr/../var"
```

File.join simply joins two strings together with a separator, whereas Pathname#join is a logical operation on two paths. It's normal for there to be differences.

That being said, I feel that Pathname.new('/usr').join(") is a nonsensical operation. It seems to result in a no-op, but it might be better to warn or raise an error.

#### #3 - 06/01/2025 02:03 PM - lovro-bikic (Lovro Bikić)

Dan0042 (Daniel DeLorme) wrote in <u>#note-2</u>:

It's not the only inconsistent behavior: (absolute and relative paths example)

To clarify, I don't expect the result of the two to be equivalent in all cases, it's clearly documented what each method does.

What I am reporting is that there's undocumented and possibly inconsistent behavior when it comes to empty strings. Furthermore, the example with a whitespace string shows that Pathname#join is capable of adding trailing slashes under certain conditions.

File.join behavior has been tested for empty strings, but Pathname#join hasn't, so it's unclear if this is a bug or an expected difference in behavior.

Dan0042 (Daniel DeLorme) wrote in #note-2:

That being said, I feel that Pathname.new('/usr').join(") is a nonsensical operation. It seems to result in a no-op, but it might be better to warn or raise an error.

Perhaps, but it's already a common pattern with File.join. Whether it's a nonsensical operation is up for debate, but I think there should be a clear way for Pathname#join to allow appending trailing slashes to pathnames.

# #4 - 06/02/2025 03:26 PM - Dan0042 (Daniel DeLorme)

Dan0042 (Daniel DeLorme) wrote in <u>#note-2</u>:

That being said, I feel that Pathname.new('/usr').join(") is a nonsensical operation. It seems to result in a no-op, but it might be better to warn or raise an error.

Ah, looks like I might have to retract that statement. In bash, cd "" is a no-op

cd /usr cd "" pwd #=> /usr

So Pathname#join, which is equivalent to cd, has the same behavior. Not sure it makes sense, but at least it's consistent. Although I should note that Dir.chdir("") raises an error.

lovro-bikic (Lovro Bikić) wrote in #note-3:

Furthermore, the example with a whitespace string shows that Pathname#join is capable of adding trailing slashes under certain conditions.

That's not a trailing slash, that the file/directory " inside "usr". But it's true that if you do .join("a/") the resulting Pathname has a trailing slash, so indeed Pathname#join is capable of adding trailing slashes under certain conditions.

I think there should be a clear way for Pathname#join to allow appending trailing slashes to pathnames.

Perhaps, but personally I don't think that joining with an empty string should be it. Maybe more like .join("./") ?

# #5 - 06/05/2025 10:13 AM - akr (Akira Tanaka)

I don't recommend trailing slash on a pathname because it is not portable between operating systems.

The behavior of Pathname (it doesn't add a trailing slash) reflects this my opinion.

https://pubs.opengroup.org/onlinepubs/9699919799/xrat/V4\_xbd\_chap04.html

| Two types of implementation have been prevalent; those that ignored trailing characters on all pathnames regardless, and those that permitted them only on existing directories.

It seems the standard tries to fix this situation, though.