Ruby master - Feature #16688

Allow #to_path object as argument to system()

03/11/2020 03:23 PM - Dan0042 (Daniel DeLorme)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

I often work with Pathname objects, but when passing them to a system command I find it a bit tedious that they have to be explicitly converted to a String.

```
file = BASE + "config.json"
system(@cmd, file)      #=> TypeError (no implicit conversion of Pathname into String)
system(@cmd, file.to_s) #=> works
```

I propose that the system/exec/spawn family of methods should try to convert their arguments using to_path, if to_str fails. I believe it makes perfect sense, since commandline arguments are so often pathnames.

This includes in/out redirection. system("ls", out: Pathname.new("file")) should be valid.

History

#1 - 03/11/2020 05:01 PM - zverok (Victor Shepelev)

I always wondered, why Pathname doesn't define to_str?.. It seems to be a perfectly suitable option: implicit conversion for "specialized" variety of the String, and it will make all non-Pathname-aware code work as expected... What are the downsides?

#2 - 03/11/2020 05:38 PM - Dan0042 (Daniel DeLorme)

I don't know the downsides, just that Pathname#to_str was specifically removed 10 years ago: #1970
So I imagine there were some design considerations there...

#3 - 03/11/2020 08:07 PM - Dan0042 (Daniel DeLorme)

- Description updated

#4 - 03/11/2020 08:29 PM - shevegen (Robert A. Heiler)

I do not have a strong opinion either way; I myself sort of just use Dir[]
rather than Pathname (I just love Dir['*.rb'] and things like that), so I would
not need pathname anyway. But I can understand the argument given - 10 years is
also quite a long time ago, and the linked issue actually then refers to a commit
from matz in 2004, so 16 years ago. That's a long time.

I would recommend to focus primarily on #to_path rather than a #to_str, if only
for the purpose of the suggestion here; and perhaps suggest it for an upcoming
developer meeting, if this is wanted by Daniel, such as for
https://bugs.ruby-lang.org/issues/16661.

#5 - 03/11/2020 11:47 PM - Eregon (Benoit Daloze)

+1, this would be useful and the lack of it caused a few bugs when moving some code to using Pathname.

#6 - 04/08/2020 04:41 PM - dsisnero (Dominic Sisneros)

I think what would be useful is when interpolating Path like objects (to_path) that the path objects are shell escaped or cmd escaped when used with system() or other cmd running objects. There are always errors on windows with directories or files with spaces. I like how Julia language deals with these in shell commands- they automatically escape the strings and expand the arrays but if we can't do that, I do agree with this proposal.

#7 - 04/10/2020 06:36 AM - akr (Akira Tanaka)

I'm positive.

It would be useful for most options for spawn.
(except open mode, "w" of system("ls", out: ["/tmp/foo", "w"]))

#8 - 04/10/2020 06:38 AM - matz (Yukihiro Matsumoto)

Accepted.

Matz.

#9 - 04/10/2020 11:44 AM - mame (Yusuke Endoh)

I'd like to confirm: system(Pathname("cat /etc/passwd")) should attempt to execute a file whose path is ./cat\ /etc/passwd, not attempt to show the content of /etc/passwd, right?

#10 - 04/10/2020 11:56 AM - Eregon (Benoit Daloze)

mame (Yusuke Endoh) wrote in #note-9:

> I'd like to confirm: system(Pathname("cat /etc/passwd")) should attempt to execute a file whose path is ./cat\ /etc/passwd, not attempt to show the content of /etc/passwd, right?

Definitely the first, we should take advantage that we know it's supposed to be a path and a not a command line with spaces.
The second would be waiting to become a security vulnerability.

#11 - 04/13/2020 05:37 PM - Dan0042 (Daniel DeLorme)

There's maybe just one small gotcha to consider:

```
o = File.open("o","w")
system("ls", out: o)          #should be equivalent
system("ls", out: o.fileno)   #to this
system("ls", out: o.to_path) #not to this
```