Ruby - Bug #6608

File.join behavior

06/19/2012 08:02 PM - Eregon (Benoit Daloze)

Status:	Rejected		
Priority:	Normal		
Assignee:	usa (Usaku NAKAMURA)		
Target version:			
ruby -v:	ruby 1.9.2p290 (2011-07-09) [i386-mingw32]	**Backport:**	

Description

Hello,

I would like to know if the current behavior of File.join is intended, and what should be its specification, especially on systems with File::ALT_SEPARATOR.

File.join adds '/' for joining if no separator is present:
File.join('a', 'b') # => "a/b"
And if a forward slash is present or two, it just acts the same:
File.join('a', '/b') or File.join('a/', 'b') # => "a/b" # => "a/b"

(These are expected)

But when a backward slash is present (= File::ALT_SEPARATOR), it seems it is kept only if it is on the right side or no other separator is present:
File.join('a', '\b') or File.join('a\', 'b') # => "a\b"
File.join('a/', '\b') # => "a\b"
but
File.join('a\', '/b') # => "a/b"

And it seems the right part is never touched, while the left part loses all its separators, unless there are none on the right side.
File.join('a//', '\b') # => "a\b"
File.join('a//', '//b') # => "a\//b"
File.join('a//', 'b') # => "a//b"

Ruby is usually giving only forward-slash paths (Dir.getwd) and accepts them only in some cases (Dir.glob).
I think conceptually File.join should remove all "" and replace them by "/", like File.expand_path does for example.
I guess this is not done for efficiency reasons.
But it also means it might produce not natural behavior:
Dir.glob(File.join('C:', '\WINDOWS', '*')) == Dir.glob('C:\WINDOWS/*') # => ["C:WINDOWS/system"] # sounds like a bug, isn't it?
but
Dir.glob(File.join('C:', '/WINDOWS', '*')) # => all files/dirs under C:/WINDOWS

The current RubySpec is a bit misleading in this regard:
https://github.com/rubyspec/rubyspec/blob/master/core/file/join_spec.rb#L21-26

History

#1 - 06/30/2012 11:33 PM - Eregon (Benoit Daloze)

Could I get some feedback on this?
Or should I go ahead and propose what I think is the right behavior as a feature?

#2 - 07/14/2012 05:52 PM - ko1 (Koichi Sasada)

- Assignee set to usa (Usaku NAKAMURA)

#3 - 07/14/2012 06:37 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

#4 - 07/27/2012 05:13 PM - usa (Usaku NAKAMURA)

- Status changed from Assigned to Rejected

=begin
These behavor is the spec of File.join.

(0) File.join never touchs any charcters in the parameters except
boundary ones.
(1) When a separator is passed, File.join respects it.
cf. File.join('a\', 'b') #=> 'a\b'
File.join('a', '\b') #=> 'a\b'
(2) If not, File.join uses File::SEPARATOR
cf. File.join('a', 'b') #=> 'a/b'
(3) If both of left and right parameters have separators,
File.join respects the right one.
The reason is that we guess that the right one is reflecting
the intention of the programmer more strongly.
cf. dir = read_from_user_edited_config_file
File.join(dir, '/file') #=> '..some_dir../file'
File.join(dir, '\file') #=> '..some_dir..\file'
Of course, I strongly recommend that we should use File.expand_path
in such case.

Your example of Dir.glob is a sad case.
Please understand this as a defect of the specification of Dir.glob, not
of File.join.
=end

#5 - 07/27/2012 06:44 PM - Eregon (Benoit Daloze)

=begin
Thank you for the detailed explanation.

I see your point and the spec seems clear, I'll update RubySpec with this.

Indeed, the documentation of Dir.glob mentions:

```
\ :
  Escapes the next metacharacter. Note that this means you cannot use
  backslash in windows as part of a glob, i.e. Dir["c:\foo*"] will not work
  use Dir["c:/foo*"] instead
```

=end