Ruby master - Feature #11428

system/exec/etc. should to_s their argument to restore Pathname functionality as it was in 1.8

08/10/2015 12:34 PM - taw (Tomasz Wegrzanowski)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

The safest way to interact with Unix shell with ruby is using Pathname and multi-argument system/exec/etc. commands.

In 1.8 days (and early 1.9) it was possible to do nice things like:

```
path = Pathname("/dev/null"); system "ls", "-l", path
```

In 1.9 branch Pathname#to_str got removed, so now this kind of code returns
system: no implicit conversion of Pathname into String (TypeError)

Im not totally convinced removing Pathname#to_str was ever a good idea, but at least commands for interacting with Unix environment like system/exec should support it - the easiest way being just to_sing their arguments if passed multiple argument form.

Here are some current alternatives and why they're worse:

```
# system call looks ok, but any kind of pathname manipulation
# is going to be nasty and error-prone using raw strings
path = "/dev/null"; system "ls", "-l", path

# this gets really ugly with complex commands, at no gain
path = Pathname("/dev/null"); system "ls", "-l", path.to_s

# this is what people are going to do, very insecure and unreliable
# will crash even in case as simple as spaces in path
path = Pathname("/dev/null"); system "ls -l #{path}"

# people will try this as well, still very insecure and unreliable
# will crash in case of ' in path or a few other weird things
path = Pathname("/dev/null"); system "ls -l '#{path}'"

# this is probably most sensible way now, but it's a bit overkill and *%W part is not pretty
path = Pathname("/dev/null"); system *%W[ls -l #{path}]
```

Currently code like:

```
 system "command", arg, arg, arg
 exec "command", arg, arg, arg
```

has no meaning when arg is not a String or something that can be #to_str-ed, so this shouldn't cause any backwards compatibility issues
(and will be a fairly late fix for unnecessary compatibility breakage with 1.8/early 1.9 era scripts).

Minor issues:

It's most important with system, I'm listing exec here mostly to avoid surprises if it behaves differently from system; there's also IO.popen etc. which might be considered for similar fix

Sometimes you want to pass other objects, like URIs and Integers, as arguments to Unix commands. This is less common than Pathname, but still nice to have, and fix I'm proposing would deal with this too

Example above is silly, but I keep running into this problem fairly often.
Some more realistic examples from my git grep and .pry_history:

```
system "open", google_url.to_s
system "du", "-hcs", *media_paths.map(&:to_s)
```

```
system "trash", *paths.map(&:to_s)
system "wget", "-nc", "-nv", url.to_s, "-O", path.to_s

Pathname(".").find.select(&:file?).each{|x| system "mv", "-nv", x.to_s, x.to_s.gsub("/", "-") }
system *%W[sox --show-progress #{source} -t mp3 #{target}.part tempo -s #{@factor}]
system *%W[convert #{apath} -modulate 100,200,100 #{bpath}]
system *%W[faad #{fn} -o #{fn_wav}]
system *%W[lame -h -b 192 #{fn_wav} #{fn_out}]
```

History

#1 - 11/06/2016 01:05 AM - akr (Akira Tanaka)

I think to_s is too strong. However calling to_path is possible option.

#2 - 11/06/2016 02:17 AM - nobu (Nobuyoshi Nakada)

Akira Tanaka wrote:

> I think to_s is too strong. However calling to_path is possible option.

After to_s failed, try to_path?