Ruby master - Bug #14591

Files with invalid multi-byte characters will cause Find::find() to raise EINVAL exception

03/08/2018 02:36 PM - jeffgrover (Jeff Grover)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.5.0p0 (2017-12-25 revision 61468) [x64-mingw32]	**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN, 2.5: UNKNOWN

Description

This can be easily duplicated by the following simple program. I believe this is mostly going to be a problem for users on Windows, where Unicode filenames are common. The example below was the name of a real file in my Recycle Bin:

First, create the problematic file:

```
c:\Users\me>copy con .◆◆◆◆000100000003f582f1e810a56094d18e
File contents
^Z
        1 file(s) copied.

c:\Users\me>dir
 Volume in drive C is Windows
 Volume Serial Number is 64A1-A9E3

 Directory of c:\Users\grove\Documents\Ruby

03/08/2018  07:18 AM    <DIR>          .
03/08/2018  07:18 AM    <DIR>          ..
03/08/2018  07:18 AM                15 .◆◆◆◆000100000003f582f1e810a56094d18e
```

Then, run the following simple Ruby program:

require 'find'

```
Find.find('.') { |path_name|
    puts path_name
}
```

You will see the exception raised:

```
Traceback (most recent call last):
        6: from dirhog.rb:22:in `<main>'
        5: from C:/Ruby25-x64/lib/ruby/2.5.0/find.rb:43:in `find'
        4: from C:/Ruby25-x64/lib/ruby/2.5.0/find.rb:43:in `each'
        3: from C:/Ruby25-x64/lib/ruby/2.5.0/find.rb:48:in `block in find'
        2: from C:/Ruby25-x64/lib/ruby/2.5.0/find.rb:48:in `catch'
        1: from C:/Ruby25-x64/lib/ruby/2.5.0/find.rb:51:in `block (2 levels) in find'
C:/Ruby25-x64/lib/ruby/2.5.0/find.rb:51:in `lstat': Invalid argument @ rb_file_s_lstat - c:\$Recyc
le.Bin/S-1-5-21-2582874610-2078213686-3622711573-1001/.????000100000003f582f1e810a56094d18e (Errno
::EINVAL)
```

The (work-around) solution I came up with was to add to the list of exceptions already handled by "ignore_errors" in lib/find.rb line 52:

```
          begin
            s = File.lstat(file)
          rescue Errno::ENOENT, Errno::EACCES, Errno::ENOTDIR, Errno::ELOOP, Errno::ENAMETOOLONG,
Errno::EINVAL
            raise unless ignore_error
            next
          end
```

This seems a reasonable compromise for now, although there is probably a better solution which involves dealing with the invalid characters and translating them to something that can be handled. The reason I am submitting this bug instead of monkey-patching a solution or something is that it is almost impossible to do this externally, you can't catch the exception while in the block of find().

If you want, I can do a GitHub pull request with the change.

As a side note, I'd also like to consider making "ignore_errors=false" the default, instead of true... as it hides problems the programmer might want to know about. If the programmer doesn't care about errors (as in my case) the documentation should clearly emphasize this option.

History

#1 - 03/08/2018 10:23 PM - shevegen (Robert A. Heiler)

> rescue Errno::ENOENT, Errno::EACCES, Errno::ENOTDIR, Errno::ELOOP, Errno::ENAMETOOLONG, Errno::EINVAL
>
> This seems a reasonable compromise for now

I am not sure.

It requires of ruby hackers to know all these various error types that they want to rescue. It's ok to be able to be very specific but perhaps it should not be so ... verbose. That's 6 entries!

I suppose you don't want to "rescue Exception" in general but perhaps there could be some sub-range such as "rescue Errno" or something. Or perhaps an even more elegant way.

IMO the above is not very elegant.

I myself also have a very few situations where I rescue multiple exceptions, in particular related to internet-connection (querying a remote phpbb webforum ... so many things can go wrong and whenever I ran into a problem, I added that particular error... but at a later time, I wondered whether that was a good approach since it takes a LOT of specific error-catching...)

#2 - 03/08/2018 10:25 PM - shevegen (Robert A. Heiler)

> If the programmer doesn't care about errors (as in my case) the documentation should clearly emphasize this option.

Agreed.

Reminds me of my own suggestion to have a variant of require where we don't have to use begin/rescue (e. g. because of a similar reason that you mentioned ... "if the programmer does not care about this or that"; in your case the error, in my case also if some external smaller add-on is unavailable).

I think matz and the core team are listening; the general error/warning means of ruby have been suggested to be enhanced by other people in the past too. I guess there may be some changes eventually sooner or later towards the path to 3.x, it just takes a little while sometimes.

mjit and optcarrot-tests for ducks, hares and cats probably have higher priority right now. :)

#3 - 08/26/2019 06:45 PM - jeremyevans0 (Jeremy Evans)

This issue started in Ruby 2.5 and is still present in 2.6. It appears to be related to the code page. If I have a directory with Japanese characters (□□□), the error occurs with my default code page (437), but does not occur if I switch to a UTF-8 code page (65001), or if I use the -Ku flag when starting Ruby. I'm not sure whether this behavior is expected with the non-UTF8 code page.

#4 - 08/27/2019 10:20 AM - duerst (Martin Dürst)

It's a result of treating file names as sequences of characters rather than as binary garbage. For some operations, one doesn't really care about the filename itself, so treating it as binary garbage would be okay. But for other operations, it isn't.

It's very difficult for Ruby to guess where the filename details matter and where not. A more complicated solution (treat filenames as characters in the current encoding if possible, as binary garbage otherwise) might work in some cases, but it may as easily just kick the can down the road, making the error appear in another location. As Jeremy wrote, running the script in UTF-8 (which we all should be doing anyway these days) is the best solution.