Ruby master - Feature #14007

open mode 'x' to raise error if file exists

10/12/2017 03:14 AM - kernigh (George Koehler)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

I propose (and attach a patch) to add a mode 'x' for Kernel#open, File.open, and similar methods. Mode 'wx' or 'ax' would create a new file, or raise an error if the file exists. Mode 'x' would be a shortcut for IO::EXCL. It would work like mode 'x' of fopen(3) in C.

```
# Create file.txt, or raise an error if it exists.
open('file.txt', 'wx') {|f| f.puts("Some text") }
```

Background

Mode 'x' appears in the fopen(3) manuals of Linux, Illumos, and multiple BSDs. In all these systems, mode 'x' to fopen(3) acts like flag O_EXCL to open(2). Linux fopen(3) describes 'x' as an extension in glibc, but 'x' now appears in other systems. FreeBSD fopen(3) and cppreference.com describe 'wx' as a C11 standard feature.

- Linux: http://man7.org/linux/man-pages/man3/fopen.3.html
- FreeBSD: https://www.freebsd.org/cgi/man.cgi?query=fopen&apropos=0&sektion=0&manpath=FreeBSD+11-current&arch=default&format=html
- http://en.cppreference.com/w/c/io/fopen

Mode 'x' is a shortcut in C programs. Without 'x', program would need to call open(2) then fdopen(3):

```
/* short way */
fp = fopen("file.txt", "wx");
if (!fp) ...

/* long way */
fd = open("file.txt", O_WRONLY|O_CREAT|O_EXCL, 0666);
if (fd == -1) ...
fp = fdopen(fd, "w");
if (!fp) { close(fd); ... }
```

Some C libraries also have a mode 'e' to set close-on-exec when opening the file. I don't propose to add mode 'e' to Ruby, because Ruby sets close-on-exec by default on most files, so I would never need to use 'e' in Ruby. NetBSD also has a mode 'f' to open only regular files, but my patch doesn't add 'f' to Ruby. (NetBSD fopen(3): http://netbsd.gw.com/cgi-bin/man-cgi?fopen)

Proposal

I propose to add mode 'x' to Ruby. If the mode is 'wx' or 'ax', then Ruby would pass O_EXCL to open(2). It would create the file, or raise an error if the file exists.

Mode 'x' would be a shortcut in Ruby, but the benefit is less than in C, because Ruby provides other ways to pass O_EXCL.

```
# with 'x'
open('file.txt', 'wx') { ... }
# short way without 'x'
open('file.txt', 'w', flags: IO::EXCL) { ... }
# long way without 'x'
open('file.txt', IO::WRONLY|IO::CREAT|IO::EXCL) { ... }
```

I wouldn't need 'x' to create a temporary file (because Ruby's Tempfile handles that), but I might use 'x' in other places, like IRB, if I didn't want to modify an existing file by mistake. One can also use 'x' when spawning external commands.

```
# Don't clobber /tmp/example if it exists.
```

```
system 'dmesg', out: ['/tmp/example', 'wx']
```

I also propose that 'rx' would raise ArgumentError in Ruby, because 'rx' would probably be a mistake of the Ruby programmer. Without the error, 'rx' would pass O_EXCL without O_CREAT and cause undefined behavior in open(2). I don't want easy undefined behavior, so my patch doesn't allow 'rx' in Ruby. It allows 'wx' and 'ax' because they pass both O_CREAT and O_EXCL.

```
irb(main):011:0> File.read('/tmp/example', mode: 'rx')
ArgumentError: can't use mode "x" with "r"
        from (irb):11:in `read'
        from (irb):11:in `<top (required)>'
        from /home/kernigh/prefix/bin/irb:11:in `<main>'
```

One can bypass this 'rx' check by not using 'x' in the mode string. For example, open('file.txt', 'r', flags: IO::EXCL) passes O_EXCL without O_CREAT, both before and after my patch.

Implementation

My patch

1. defines FMODE_EXCL in the public header ruby/io.h. (I'm not sure how to pick a value; I picked 0x00002000.)
2. edits a few functions in io.c, so
    - rb_io_modestr_fmode() accepts 'x' and rejects 'rx'. It translates 'x' to FMODE_EXCL.
    - rb_io_oflags_fmode() translates O_EXCL to FMODE_EXCL. (I'm not sure if this part is needed.)
    - rb_io_fmode_oflags() translates FMODE_EXCL to O_EXCL.
3. adds 'x' to the document for IO.new.
4. specifies 'x' in Ruby spec suite. The only tested method is File.open.

To run the tests,
$ make test-spec SPECOPTS=core/file/open_spec.rb

My patch doesn't change make test-all. (Because I run OpenBSD, I have some difficulty running Ruby's tests. The FIFO tests can get stuck because of OpenBSD's bug. I have hacked those tests to fail instead of getting stuck, using the diff at https://gist.github.com/kernigh/5770f8b90427ce6ede535dae729cb960)

My patch assumes that O_EXCL works on every system. Ruby made this assumption before me. Ruby always defines File::Constants::EXCL in file.c. Also, the Tempfile library doesn't work unless the system knows O_EXCL.

Odd behavior of 'x'

The document for 'x' in my patch says only,

```
  "x"  Exclusive open
       Creates a new file, or raises an error if the file
       exists.  Mode "x" is available since Ruby 2.5.
```

This document might be too brief. It doesn't mention that "x" acts like File::EXCL. It also doesn't describe some oddities of 'x', like how 'rx' raises an ArgumentError, or when 'x' is ignored.

With my patch, Ruby ignores 'x' if it isn't calling open(2). For example, opening a process always ignores 'x'. That's not too bad, because Ruby creates a new process.

```
# ignores 'x'
open('|cat', 'wx') {|f| f.puts "Hi" }
```

Also, Ruby ignores 'x' when opening a file descriptor. That's odd, because 'x' should never open an existing file, but it can do so if we use fd.

```
fd = IO.sysopen('/tmp/example', 'a')
# ignores 'x', also doesn't truncate file
IO.open(f, 'wx') {|f| f.puts('the last line') }
```

It may seem strange that my patch raises ArgumentError for 'rx', but ignores other strange uses of 'x'.

Related issues:

Is duplicate of Ruby master - Feature #11258: add 'x' mode character for O_EXCL	**Closed**

History

#1 - 10/12/2017 04:54 AM - shevegen (Robert A. Heiler)

Might be interesting. I have nothing against it but I guess matz and the ruby core team have to decide on whether it is sufficiently useful. In the case that it is approved, I hope that the documentation can be clear since ruby hackers need some explanation as to when/why to use the new parameter.

As for ArgumentError via 'rx' such as "mode: 'rx'" - if we ignore the reasoning given above, I think there is one reason that speaks against it at the least, which is backwards compatibility. Matz said a few times that he is reluctant to break backwards compatibility unless there is a very good reason, in the ruby 2.x branch. So the changed behaviour may be more fitting towards ruby 3.x, but to be honest, I would suggest to file a separate issue for the ArgumentError case and decouple it from your suggestion above about adding a new parameter. People may be more likely to adpopt new behaviour rather than be forced to change any existing code base, without it bringing them a lot of advantages (e. g. the ArgumentError case). I myself am neutral on it largely because it does not seem to be a big issue for me either way, I am fine with things as they are and I am also fine with things if they change (unless I may have missed something fundamental).

#2 - 10/12/2017 05:21 AM - normalperson (Eric Wong)

xkernigh@netscape.net wrote:

> Feature #14007: open mode 'x' to raise error if file exists
> https://bugs.ruby-lang.org/issues/14007

I like this, it's consistent with fopen in glibc and FreeBSD, at least
and much easier to type than IO::EXCL.

#3 - 10/13/2017 12:50 AM - shyouhei (Shyouhei Urabe)

- Is duplicate of Feature #11258: add 'x' mode character for O_EXCL added

#4 - 10/16/2017 02:43 AM - kernigh (George Koehler)

Sorry, I didn't know that feature #11258 existed. I might have searched bugs.ruby-lang.org for "open mode" but not looked through the long list of results.

#5 - 08/09/2018 08:49 AM - znz (Kazuhiro NISHIYAMA)

- Status changed from Open to Closed

Applied in changeset trunk|r64245.

---

add 'x' mode character for O_EXCL

[Feature #11258]
Patch by cremno (cremno phobia)

Files

ruby-mode-x.diff	3.52 KB	10/12/2017	kernigh (George Koehler)