Ruby - Bug #12307

File.new and File.open change permissions even if the file exists on Windows

04/21/2016 09:14 PM - Eregon (Benoit Daloze)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.2.4p230 (2015-12-16 revision 53155) [i386-mingw32]	**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

For instance:

```
# New file
File.open("abc", "w", 0666) { |f|
  puts f.stat.mode.to_s(8) # => 100666, OK
}

# File exists
File.open("abc", "w", 0466) { |f|
  puts f.stat.mode.to_s(8) # => 100444, BUG
}
```

So the mode of the file was changed even though the file already exists.
This is inconsistent with the behavior on other platforms such as UNIX which only consider mode for new files.
open(2) is fairly clear about this on Linux and OS X: "if neither O_CREAT nor O_TMPFILE is specified, then mode is ignored".

History

#1 - 04/21/2016 09:51 PM - usa (Usaku NAKAMURA)

To my understanding, POSIX doesn't say so.
It only says that the mode is not changed when specified O_TRUNC for an existing file.
(http://pubs.opengroup.org/onlinepubs/9699919799/)

This behavior is derived from the implementation of MSVCRT.
Changing this is a little difficult because of the limitation of the Windows API.
So, to persuade me, please show more strong example, such as a standard like POSIX, or more samples from UNIX-like platforms (BSDs, Solaris, AIX, HP-UX and so on).

#2 - 04/21/2016 10:25 PM - Eregon (Benoit Daloze)

Usaku NAKAMURA wrote:

> To my understanding, POSIX doesn't say so.
> It only says that the mode is not changed when specified O_TRUNC for an existing file.
> (http://pubs.opengroup.org/onlinepubs/9699919799/)
>
> This behavior is derived from the implementation of MSVCRT.
> Changing this is a little difficult because of the limitation of the Windows API.
> So, to persuade me, please show more strong example, such as a standard like POSIX, or more samples from UNIX-like platforms (BSDs, Solaris, AIX, HP-UX and so on).

My understanding is that the 3rd argument to open(2) only makes sense with O_CREAT.
In fact, most man pages don't even show the 3rd argument, and only explain it in the context of O_CREAT, so it is clearly an unexpected behavioral extension to interpret it in other cases.

OS X and Open BSD man pages seem to explain that clearly:
http://man.openbsd.org/?query=open&apropos=0&sec=0&arch=default&manpath=OpenBSD-current:
"The flags argument may indicate the file is to be created if it does not exist (by specifying the O_CREAT flag), in which case the file is created with a mode specified by an additional argument of type mode_t as described in chmod(2) and modified by the process' umask value (see umask(2))."

And I believe the POSIX page you linked as well, in slightly less clear terms:
"If the file exists, this flag has no effect except as noted under O_EXCL below.
Otherwise, the file shall be created; the user ID of the file shall be set ...; the group ID of the file shall be set ...; and the access permission bits (see sys/stat.h) of the file mode shall be set to *the value of the argument following the oflag argument* taken as type mode_t modified as follows: a bitwise

AND is performed on the file-mode bits and the corresponding bits in the complement of the process' file mode creation mask."

This is the only time the 3rd argument is mentioned and says that O_CREAT does set the permission bits.
It is also said that if O_CREAT is passed and the file exists it has no effect (and so the 3rd argument is not interpreted).

The document does not explicitly forbids to change file permissions (except for O_TRUNC but the motivation seems to be to have well-defined semantics with concurrent access and change nothing but the file size), but I believe that to be an unexpected and unreasonable side-effect. If such a thing happens, then one could argue any random function taking extra arguments is allowed to change file permissions.

#3 - 04/21/2016 10:28 PM - Eregon (Benoit Daloze)

FWIW, Ruby documentation of File.open and File.new is also phrased the same way:
"If a file is being created, its initial permissions may be set using the perm parameter."
"If a file is being created, permission bits may be given in perm."

It is very clear the permissions are only for newly-created files.

#4 - 04/21/2016 10:31 PM - Eregon (Benoit Daloze)

Usaku NAKAMURA wrote:

> This behavior is derived from the implementation of MSVCRT.
> Changing this is a little difficult because of the limitation of the Windows API.

Does open(2) in MSVCRT causes this bug?
Then it seems a bug in MSVCRT since
https://msdn.microsoft.com/en-us/library/aa298516%28v=vs.60%29.aspx
mentions "The pmode argument is required only when _O_CREAT is specified. If the file already exists, pmode is ignored."

#5 - 04/21/2016 11:00 PM - usa (Usaku NAKAMURA)

> Does open(2) in MSVCRT causes this bug?

No.
Ruby always call open(2) with O_CREAT when opening files with "w".
On a simple level, ruby's File.open("abc", "w", 0666) is translated into C's open("abc", O_CREAT | O_TRUNC | O_RDONLY, 0666).

Ok, let me show what is happening with MSVCRT.

file open.c:

```
#include <fcntl.h>
#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

main(void)
{
    int flags = _O_CREAT | _O_TRUNC | _O_WRONLY;
    int fd = _open("abc", flags, 0666);
    struct _stat stat;

    if (fd == -1) {
 printf("error(open 1) %d\n", errno);
 exit(1);
    }
    if (_fstat(fd, &stat)) {
 printf("error(fstat 1) %d\n", errno);
 exit(1);
    }
    printf("%o\n", stat.st_mode);
    close(fd);

    fd = _open("abc", flags, 0466);
    if (fd == -1) {
 printf("error(open 2) %d\n", errno);
 exit(1);
    }
    if (_fstat(fd, &stat)) {
 printf("error(fstat 2) %d\n", errno);
```

```
 exit(1);
    }
    printf("%o\n", stat.st_mode);
    close(fd);
}
```

Result:

```
D:\test> cl -MD open.c
Microsoft (R) C/C++ Optimizing Compiler Version 16.00.40219.01 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

open.c
Microsoft (R) Incremental Linker Version 10.00.40219.01
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:open.exe
open.obj

D:\test> open
100666
100444
```

#6 - 04/21/2016 11:17 PM - usa (Usaku NAKAMURA)

> The document does not explicitly forbids to change file permissions (except for O_TRUNC but the motivation seems to be to have well-defined semantics with concurrent access and change nothing but the file size), but I believe that to be an unexpected and unreasonable side-effect. If such a thing happens, then one could argue any random function taking extra arguments is allowed to change file permissions.

Of course, I agree with you.
I've failed to cloud the issue :-P

BTW, this problem occurs only when opening an existing file with "w" and mode 04xx.
I think this is not a real use case, and this problem may not causes any real problem.
As shown above, this is the limitation of MSVCRT (and Windows API).
I want to leave this as a specification of Ruby on Windows (or, if you want to call so, a known bug).

#7 - 04/22/2016 11:42 AM - Eregon (Benoit Daloze)

- Status changed from Open to Rejected

Thank you for the C example and explanation!
So indeed the behavior of MSVCRT contradicts its description:
"[...] If the file already exists, pmode is ignored." but it's not in practice.

So it's indeed a bug either in MSVCRT documentation or implementation, right?

> BTW, this problem occurs only when opening an existing file with "w" and mode 04xx.
> I think this is not a real use case, and this problem may not causes any real problem.

I am not sure, maybe it can be used to create a read-only file but still allow to write the initial contents ("w" counts for permissions to write for the fd, 04xx is only for further open calls):
File.open("abc", "w", 0444) { |f| f.write "abcd" }

It's probably not very frequent in Ruby code, indeed.
You are the maintainer, so this is your decision :)

If somebody has this problem in a real scenario, please reopen this bug or comment.