IO#advise should raise NotImplementedError on platforms that do not support that call


Right now IO#advise returns nil when passed legal arguments even on platforms that do not support that POSIX function (like OS X). To be consistent with other calls in core, I think that IO#advise should raise NotImplementedError for unsupported platforms.

A similar topic on IO#advise was recently discussed in https://bugs.ruby-lang.org/issues/11806#note-6.

To support this idea, I’d like to point to how File#link behaves in a similar situation on platforms that do not support hard links in the filesystem. It raises NotImplementedError.

https://github.com/ruby/ruby/blob/trunk/file.c#L2723

I am bringing up this issue so we can make Ruby behave more consistently and behave in a consistent manner when trying to access functions on unsupported platforms.

# Backport: 2.0.0: UNKNOWN, 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Right now IO#advise returns nil when passed legal arguments even on platforms that do not support that POSIX function (like OS X). To be consistent with other calls in core, I think that IO#advise should raise NotImplementedError for unsupported platforms.

Simply impossible.
For example, Linux define POSIX_FADV_NOREUSE, but not implement AND no error return. Then, IO#advise return nil and just no-op in this case.

We can implement raise NotImplementedError SOME CASES and not raise another some cases. But nobody want such inconsistency mess.

Unsubscribe: ruby-core-request@ruby-lang.org?subject=unsubscribe
http://lists.ruby-lang.org/cgi-bin/mailman/options/ruby-core

#2 - 01/12/2016 01:45 PM - cremes (Chuck Remes)

I may have miscommunicated, so let me try to clarify. I think these should be the rules

1. If the platform supports posix_fadvise, then it should NOT raise NotImplemented under any circumstances.

2. If the platform supports posix_fadvise but it does NOT support all arguments like POSIX_FADV_NOREUSE, then it should return nil like it does now. It is not reasonable to test for and figure out all different legal arguments across multiple platforms, so the current behavior is the best that we can do.
3. If the platform DOES NOT SUPPORT posix_fadvise, then it should raise NotImplementedError in all cases.

I hope that is more clear. These rules should make IO#advise MORE CONSISTENT.

#3 - 01/13/2016 12:17 AM - normalperson (Eric Wong)

git@chuckremes.com wrote:

1. If the platform DOES NOT SUPPORT posix_fadvise, then it should raise NotImplementedError in all cases.

I hope that is more clear. These rules should make IO#advise MORE CONSISTENT.

I disagree. File#link is not analogous, nor are any other methods in Ruby. Unlike existing methods, IO#advise is a hint, not an imperative command.

Things are supposed to go horribly wrong if errors to imperative commands are not reported properly. However, advice should be safely ignorable without catastrophe.

Furthermore, if (and a big "if") we were to start raising NotImplementedError, it would break existing users who (perhaps unknowingly) rely on that behavior.

Unsubscribe: ruby-core-request@ruby-lang.org?subject=unsubscribe
http://lists.ruby-lang.org/cgi-bin/mailman/options/ruby-core

#4 - 01/13/2016 08:28 AM - kosaki (Motohiro KOSAKI)

On Tue, Jan 12, 2016 at 7:06 PM, Eric Wong normalperson@yhf.ts.net wrote:

git@chuckremes.com wrote:

1. If the platform DOES NOT SUPPORT posix_fadvise, then it should raise NotImplementedError in all cases.

I hope that is more clear. These rules should make IO#advise MORE CONSISTENT.

I disagree. File#link is not analogous, nor are any other methods in Ruby. Unlike existing methods, IO#advise is a hint, not an imperative command.

Things are supposed to go horribly wrong if errors to imperative commands are not reported properly. However, advice should be safely ignorable without catastrophe.

Furthermore, if (and a big "if") we were to start raising NotImplementedError, it would break existing users who (perhaps unknowingly) rely on that behavior.

Currently, I stand aside Eric.
Chuck, do you have any specific and useful use cases of your proposal?
That is extremely important for introducing an incompatibility.

Unsubscribe: ruby-core-request@ruby-lang.org?subject=unsubscribe
http://lists.ruby-lang.org/cgi-bin/mailman/options/ruby-core

#5 - 01/13/2016 06:35 PM - cremes (Chuck Remes)

I do not have any use-cases, unfortunately. I am just arguing from a standpoint of what I believe to be correct. If a platform DOES NOT have a function, when I call that function it should fail loudly. I don't care if the function is a hint or otherwise.

Thanks for your time.

#6 - 01/13/2016 07:16 PM - Eregon (Benoit Daloze)

Ruby APIs try to unify across platforms, and IO is not just POSIX.

So I think "noop returning nil" is an appropriate implementation on systems which do not support posix_fadvise. For instance, Process.clock_gettime also works for generic clocks, even on system without clock_gettime(2). There is no appropriate replacement for "fork" on Windows, so there it makes sense to fail hard.
Closing this issue, since people seem to agree the current behavior is intended. (Also see https://github.com/ruby/spec/issues/183#issuecomment-171467414)