Ruby - Bug #18882

File.read cuts off a text file with special characters when reading it on MS Windows

06/27/2022 11:12 AM - magynhard (Matthäus Johannes Beyrle)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.1.2p20 (2022-04-12 revision 4491bb740a) [x64-mingw-ucrt]	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN

Description

When using File.read to read a text file (in this case a javascript file) with special characters, the content is cut off at special characters.

It occurs only when running ruby on Windows, tried several versions, including the latest.

Does not occur on Linux or WSL (Windows Subsystem for Linux).

I created a github repo including a test script and the source file as the result inside a file as well:
https://github.com/grob-net4industry/ruby_win_file_bug

Related issues:

Related to Ruby - Feature #19193: drop DOS TEXT mode support	**Assigned**

Associated revisions

Revision f1057393da7a98e447ee7679db69aeec8f4d1650 - 08/05/2022 06:34 PM - alanwu (Alan Wu)

[DOC] Clarify that IO.read uses text mode

See: https://bugs.ruby-lang.org/issues/18882#note-13

[Bug #18882]

Revision f1057393da7a98e447ee7679db69aeec8f4d1650 - 08/05/2022 06:34 PM - alanwu (Alan Wu)

[DOC] Clarify that IO.read uses text mode

See: https://bugs.ruby-lang.org/issues/18882#note-13

[Bug #18882]

Revision f1057393 - 08/05/2022 06:34 PM - alanwu (Alan Wu)

[DOC] Clarify that IO.read uses text mode

See: https://bugs.ruby-lang.org/issues/18882#note-13

[Bug #18882]

History

#1 - 06/27/2022 12:47 PM - chrisseaton (Chris Seaton)

Don't you need to use binary mode to read these characters?

#2 - 06/27/2022 01:44 PM - magynhard (Matthäus Johannes Beyrle)

chrisseaton (Chris Seaton) wrote in #note-1:

> Don't you need to use binary mode to read these characters?

As it is a text file and reading from it behaves correctly on Linux, i expect not to need to use binary mode.

Otherwise, should it be windows specific for a reason, that cannot be changed, at least an error message would be appropriate? Because it fails silently.

#3 - 06/27/2022 02:02 PM - austin (Austin Ziegler)

This is Windows-specific, and there is not an error condition to check for reporting on this.

The distinction between "binary" and "text" mode is simply the way that DOS works (yes, that's how old this behaviour is), and the definition of text is *essentially* just characters 0x20 to 0x1e. If I remember correctly.

Using open(filename, 'rb') is the only way to do this on Windows, and there is no such thing as "text mode" and "binary mode" on any system other than an old DOS-based mechanism.

This is not a bug.

#4 - 06/27/2022 02:14 PM - chrisseaton (Chris Seaton)

> it behaves correctly on Linux, i expect not to need to use binary mode

I think binary mode is a no-op on Linux though? Different systems behave differently - Ruby can't completely remove those differences when you're doing IO.

#5 - 06/27/2022 02:19 PM - alanwu (Alan Wu)

File.read (actually IO.read) interprets the file based on the
external encoding, which is a function of the platform and the environment.
If you put a p [content.encoding, ::Encoding.default_external] in your script
I think you should see a difference between the Windows Ruby and WSL Ruby.

Assuming that's the problem, I don't know how one could issue a warning for it
because Ruby can't know if the external encoding matches user expectation.
We could potentially improve the docs, though.

#6 - 06/27/2022 02:30 PM - austin (Austin Ziegler)

alanwu (Alan Wu) wrote in #note-5:

> File.read (actually IO.read) interprets the file based on the
> external encoding, which is a function of the platform and the environment.
> If you put a p [content.encoding, ::Encoding.default_external] in your script
> I think you should see a difference between the Windows Ruby and WSL Ruby.

This is true, but not the source of the problem here. See
https://docs.microsoft.com/en-us/cpp/c-runtime-library/text-and-binary-mode-file-i-o?view=msvc-170 and
https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/fopen-wfopen?view=msvc-170.

The only real improvement to the documentation would be to suggest opening files for read as 'rb' *all* the time if your software might be used on Windows or might be anything other than ASCII text with only a few control characters supported (0x1A is particularly dangerous, but others like 0x00 might also cause problems). This is well-known Windows behaviour.

#7 - 06/27/2022 02:46 PM - magynhard (Matthäus Johannes Beyrle)

alanwu (Alan Wu) wrote in #note-5:

> File.read (actually IO.read) interprets the file based on the
> external encoding, which is a function of the platform and the environment.
> If you put a p [content.encoding, ::Encoding.default_external] in your script
> I think you should see a difference between the Windows Ruby and WSL Ruby.
>
> Assuming that's the problem, I don't know how one could issue a warning for it
> because Ruby can't know if the external encoding matches user expectation.
> We could potentially improve the docs, though.

Added it to the script, both have the same output in this case:

```
[#<Encoding:UTF-8>, #<Encoding:UTF-8>]
```

But as @alanwu (Alan Wu) stated, there is a 0x1A charachter (26, = CTRL+Z) inside the javascript file, which is interpreted as end of file on text mode on windows:
https://stackoverflow.com/a/230878/11205329

Might be nice if File.read would use binary mode by default, but that would be a breaking change. So extending the documentation might be the better choice. Or do you think that there is a smart way to throw an error in cases, where a 0x1A character is included and therefore the read file is incomplete?

#8 - 06/27/2022 03:10 PM - austin (Austin Ziegler)

magynhard (Matthäus Johannes Beyrle) wrote in #note-7:

> alanwu (Alan Wu) wrote in #note-5:
> Might be nice if File.read would use binary mode by default, but that would be a breaking change. So extending the documentation might be the better choice. Or do you think that there is a smart way to throw an error in cases, where a 0x1A character is included and therefore the read file is incomplete?

There is no way to throw an error in this case, because there's no error condition. 0x1a is read just fine on non-Windows systems in "text" mode, but ignored on Windows, because the underlying operating system APIs respect file mode, which is not really accessible to Ruby. The OS sees an EOF and reports that back to Ruby. Ruby does not know this.

#9 - 06/27/2022 04:40 PM - alanwu (Alan Wu)

Huh, Ruby might be exposing a C runtime behavior here.
Assuming I'm reading the call graph right, and there
isn't some Ruby level defaulting going on, a File.read()
with just a path and no mode maps to a call to _wopen()
with neither _O_TEXT nor _O_BINARY set. In this
case the runtime uses a global _fmode to decide the
actual mode, according to the docs here: https://docs.microsoft.com/en-us/cpp/c-runtime-library/text-and-binary-mode-file-i-o?view=msvc-170

So potentially, someone could use Fiddle or some C extension
to call _set_fmode(_O_BINARY) to change the behavior of plain
File.read() calls.

Anyways clearly I'm out of my depth here as you can tell how
I got the issue wrong initially. I agree that it's surprising
that File.read doesn't read every byte, though. I will
add this to the next dev meeting because I'm interested in what
other people think.

The call graph from File.read() to _wopen() (again, it might be wrong):

```
rb_io_s_read
  open_key_args
    rb_io_open
      rb_io_open_generic
        rb_sysopen
          rb_sysopen_internal
            sysopen_func
              rb_cloexec_open
                open [(rb_w32_uopen when _WIN32)](https://github.com/ruby/ruby/blob/aba804ef91a5b2aa88efdd74205026aca3f943b2/io.c#L178-L181)
                  _wopen
```

#10 - 06/28/2022 02:09 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected

You can use File.binread.

#11 - 07/01/2022 11:47 AM - nobu (Nobuyoshi Nakada)

alanwu (Alan Wu) wrote in #note-9:

> So potentially, someone could use Fiddle or some C extension
> to call _set_fmode(_O_BINARY) to change the behavior of plain
> File.read() calls.

You mean we should use O_TEXT explicitly when calling _wopen?

#12 - 07/01/2022 11:52 AM - nobu (Nobuyoshi Nakada)

Actually, NEED_NEWLINE_DECORATOR_ON_READ_CHECK() and do_writeconv() set O_TEXT or O_BINARY always on each I/O.

#13 - 07/21/2022 12:20 PM - mame (Yusuke Endoh)

We discussed this ticket at the dev meeting.

@usa (Usaku NAKAMURA) and @nobu (Nobuyoshi Nakada) said that File.read reads a file in text mode. And the VC runtime (msvcrt) does EOF

character handling, CRLF conversion, etc under text mode. There is no fine-grained control on the Ruby side.

It is possible for File.read to read a file in binary mode, but it would require careful consideration about implementation, side effect, compatibility, etc. For now, it is good to add a sentence like "This method reads a file in text mode." to the rdoc of File.read (and File.write).

#14 - 12/09/2022 04:56 PM - mame (Yusuke Endoh)

- Related to Feature #19193: drop DOS TEXT mode support added

Files

copy_pdfmake.min.js	582 KB	06/27/2022	magynhard (Matthäus Johannes Beyrle)
pdfmake.min.js	1.29 MB	06/27/2022	magynhard (Matthäus Johannes Beyrle)
diff.png	55.9 KB	06/27/2022	magynhard (Matthäus Johannes Beyrle)