Ruby - Bug #18909

ARGF.readlines reads more than current file

07/13/2022 01:00 PM - JohanJosefsson (Johan Josefsson)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.3.1p112 (2016-04-26) [x86_64-linux-gnu]	**Backport:**	2.7: REQUIRED, 3.0: REQUIRED, 3.1: DONE

Description

The docuentation says that ARGF.readlines: *Reads ARGF's current file in its entirety* , but this is what happens:

$ cat fileA A $ cat fileB B $ ruby -e 'puts ARGF.readlines' fileA fileB A B
i.e. it reads both the current file and the next one (all files?).

Associated revisions

Revision 280b805d040fa537d5a459b40d4bfa6d49700905 - 07/15/2022 02:30 PM - peterzhu2118 (Peter Zhu)

[DOC] Fix documentation for ARGF#readlines

[Bug #18909]

Revision 280b805d040fa537d5a459b40d4bfa6d49700905 - 07/15/2022 02:30 PM - peterzhu2118 (Peter Zhu)

[DOC] Fix documentation for ARGF#readlines

[Bug #18909]

Revision 280b805d - 07/15/2022 02:30 PM - peterzhu2118 (Peter Zhu)

[DOC] Fix documentation for ARGF#readlines

[Bug #18909]

Revision 5e25ba5d07d42f02485235e3962a4a28373c50e1 - 10/23/2022 10:09 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 280b805d040fa537d5a459b40d4bfa6d49700905: [Backport #18909]

```
    [DOC] Fix documentation for ARGF#readlines

    [Bug #18909]
    ---
     io.c | 8 ++++----
     1 file changed, 4 insertions(+), 4 deletions(-)
```

Revision 5e25ba5d07d42f02485235e3962a4a28373c50e1 - 10/23/2022 10:09 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 280b805d040fa537d5a459b40d4bfa6d49700905: [Backport #18909]

```
    [DOC] Fix documentation for ARGF#readlines

    [Bug #18909]
    ---
     io.c | 8 ++++----
     1 file changed, 4 insertions(+), 4 deletions(-)
```

Revision 5e25ba5d - 10/23/2022 10:09 AM - nagachika (Tomoyuki Chikanaga)

merge revision(s) 280b805d040fa537d5a459b40d4bfa6d49700905: [Backport #18909]

```
    [DOC] Fix documentation for ARGF#readlines

    [Bug #18909]
    ---
     io.c | 8 ++++----
     1 file changed, 4 insertions(+), 4 deletions(-)
```

History

#1 - 07/14/2022 08:47 PM - peterzhu2118 (Peter Zhu)

Thank you for the report. I opened a pull request that fixes this issue.

#2 - 07/15/2022 06:33 AM - JohanJosefsson (Johan Josefsson)

Hm…if I understand it correctly, this just changed the documentation to conform to a strange behavior. Strange in the sense that the -i (inplace) option becomes meaningless in the case of multiple files an ARGV.readlines. Would it not be better to keep the documentation and change the implementation so it conforms?

#3 - 07/15/2022 02:28 PM - peterzhu2118 (Peter Zhu)

This is how it's designed to work, as specified in the Ruby spec. Unfortunately, this makes it difficult to use the -i flag since ARGF.readlines will read to the very last file, meaning that all the output will be written to the very last file.

#4 - 07/15/2022 03:53 PM - ufuk (Ufuk Kayserilioglu)

Moreover, it is documented that ARGF works this way in general: https://ruby-doc.org/core-3.1.0/ARGF.html

> You can now use ARGF to work with a concatenation of each of these named files. For instance, ARGF.read will return the contents of file1 followed by the contents of file2.

#5 - 07/15/2022 05:30 PM - peterzhu2118 (Peter Zhu)

- Status changed from Open to Closed

Applied in changeset git|280b805d040fa537d5a459b40d4bfa6d49700905.

[DOC] Fix documentation for ARGF#readlines

[Bug #18909]

#6 - 07/15/2022 05:30 PM - peterzhu2118 (Peter Zhu)

- Backport changed from 2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN to 2.7: REQUIRED, 3.0: REQUIRED, 3.1: REQUIRED

#7 - 07/15/2022 05:36 PM - JohanJosefsson (Johan Josefsson)

Yes, ARGF can work with a concatenated pseudo file in several ways but this idea collides squarely with the idea of inplace editing (-i option). Kernel.readlines already makes an array from the concatenated file. Therefore I think a method that only reads the current file an thereby can keep the inplace sematics would be useful.

#8 - 07/15/2022 05:51 PM - ufuk (Ufuk Kayserilioglu)

JohanJosefsson (Johan Josefsson) wrote in #note-7:

> Yes, ARGF can work with a concatenated pseudo file in several ways but this idea collides squarely with the idea of inplace editing (-i option). Kernel.readlines already makes an array from the concatenated file. Therefore I think a method that only reads the current file an thereby can keep the inplace sematics would be useful.

It is not the case that ARGF *can* work with a concatenated stream, it **does** work with a concatenated stream of all the files in ARGV, which you can control by modifying ARGV, btw. It is true that this is problematic with the -i option, but you are asking to change the existing behaviour of a documented API. That should be a feature request at best, but is certainly not a bug.

#9 - 07/15/2022 06:48 PM - JohanJosefsson (Johan Josefsson)

But this "documented API" was not implemented in the way it was described. From my limited viewpoint I would have preferred a change in the implementation and not in the documentation.
There is nothing in the introductory paragraphs on ARGF that says that the input files must be treated as a concatenated file. On the contrary, there is this 'current file' concept that is mentioned in several places.
I will maybe try to modify ARGV but that seems like a clumsy way to do what ARGF could have done if 'current file' was exposed to the user in a more suitable way.
But never mind, now I know how it works and the documentation and implementation will be in synch!
Thank you

#10 - 07/15/2022 07:17 PM - austin (Austin Ziegler)

JohanJosefsson (Johan Josefsson) wrote in #note-9:

> There is nothing in the introductory paragraphs on ARGF that says that the input files must be treated as a concatenated file. On the contrary,

> there is this 'current file' concept that is mentioned in several places.

I don’t agree with your assessment here. I just copied this from the Ruby 3.1 documentation snapshot that I use from Dash (mirroring https://ruby-doc.org/core-3.1.2/ARGF.html):

> ARGF is a stream designed for use in scripts that process files given as command-line arguments or passed in via STDIN.
>
> The arguments passed to your script are stored in the ARGV Array, one argument per element. ARGF assumes that any arguments that aren't filenames have been removed from ARGV. For example:
>
> ```
> $ ruby argf.rb --verbose file1 file2
>
> ARGV  #=> ["--verbose", "file1", "file2"]
> option = ARGV.shift #=> "--verbose"
> ARGV  #=> ["file1", "file2"]
> ```
>
> You can now use ARGF to work with a concatenation of each of these named files. For instance, ARGF.read will return the contents of file1 followed by the contents of file2.

Yes, it’s the third paragraph, but it is fairly clear.

On the other hand, #each_line and #inplace_mode= make it clearer that if you don’t force a full *read* (#readlines or #read), you can have it work properly with the in-place mode.

I would suggest that what you want is perhaps an added function, #each_file, that allows you to loop over each file separately—that would make a fairly good feature request, IMO.

#11 - 07/16/2022 10:06 AM - Eregon (Benoit Daloze)

My two cents, options like -i/-n/-p shouldn't be used, I think they are vestigial legacy flags inherited from Perl or so.
Ruby is expressive enough, it is short enough to have an explicit File.write, loop, or whatever you need without needing to rely on a lot of magic with these old flags (which make the logic unreadable).

ARGF is well established as being the concatenation of all such files, there is no chance to change that.

#12 - 07/16/2022 10:25 AM - JohanJosefsson (Johan Josefsson)

austin (Austin Ziegler) wrote in #note-10:
...
In your link it is clear that e.g. in the case of read, ARGF concatenates the input files, not that it always treats the input as a concatenated file. (That third paragraph.) It is also very clear that readlines operates on the current file, not on a concatenated input file.
I see a use case for this specified behavior and I cannot see why this could not be kept and the implementation be updated to conform. (I.e. apart from practical problems.) Kernel.readlines still operates on the concatenated file if that behavior is needed.

My reference toy example:
I want to prepend every specified file with a linecount. Here I have implemented ARGF.readlines according to the original spec and the task becomes very simple:

```
$ cat a b
primo
secundo
PRIMO
$ ruby -r ./argf_readlines.rb -i.bak -e 'while x=argf_readlines do puts "#{x.size} lines:";puts x end' a b
$ cat a b
2 lines:
primo
secundo
1 lines:
PRIMO
```

Maybe a feature request for an added function #readlines_current would make us all happy?
(I cannot intuitively see what a function #each_file would do so I cannot comment on that.)

PS: the documentation of the inplace_mode methods is not correct. I will make a report on that.

#13 - 07/16/2022 10:47 AM - JohanJosefsson (Johan Josefsson)

Eregon (Benoit Daloze) wrote in #note-11:

> My two cents, options like -i/-n/-p shouldn't be used, I think they are vestigial legacy flags inherited from Perl or so.
> Ruby is expressive enough, it is short enough to have an explicit File.write, loop, or whatever you need without needing to rely on a lot of magic with these old flags (which make the logic unreadable).

> ARGF is well established as being the concatenation of all such files, there is no chance to change that.

Sad to hear that. Those perl-like features have been my favorite part of ruby. It lets me be a command line ninja and make exceedingly powerful one liners while employing less magic than what I had done using perl/sed/awk/random command line utility. While all the time knowing that my tiny command line can grow, seamlessly without change of language, to a one file program, to a larger program and even to a production quality program. Maybe I am fishing in the wrong pond/barking up the wrong tree/fighting windmills but I really like this as a relative newcomer. In my view, it is here that ruby outperforms the competition and it has been the reason for me not to go with more mainstream alternatives.

#14 - 07/19/2022 09:28 AM - mame (Yusuke Endoh)

Is this what you want?

```
$ echo primo > a
$ echo secundo >> a
$ echo PRIMO > b

$ ruby -i.bak -e 'until ARGF.closed?; x = ARGF.file.readlines; puts "#{ x.size } lines"; puts x; ARGF.skip; en
d' a b

$ cat a
2 lines
primo
secundo

$ cat b
1 lines
PRIMO
```

TBH I don't recommend to use -i either

#15 - 07/19/2022 10:47 AM - Eregon (Benoit Daloze)

JohanJosefsson (Johan Josefsson) wrote in #note-13:

> Sad to hear that. Those perl-like features have been my favorite part of ruby. It lets me be a command line ninja and make exceedingly powerful one liners while employing less magic than what I had done using perl/sed/awk/random command line utility. While all the time knowing that my tiny command line can grow, seamlessly without change of language, to a one file program, to a larger program and even to a production quality program.

To be clear, I think ruby -e '...' is great and useful, and I agree with your comment about growing from tiny command line to one file to larger program.
I just think -i/-n/-p are unnecessary because they can be replaced with just a few more characters and then you have full control and it is explicit.
It is the problem of those 3 flags, they only work in very few situations and then are useless for everything else.

#16 - 08/05/2022 08:02 PM - JohanJosefsson (Johan Josefsson)

mame (Yusuke Endoh) wrote in #note-14:

> Is this what you want?
>
> ```
> $ echo primo > a
> $ echo secundo >> a
> $ echo PRIMO > b
>
> $ ruby -i.bak -e 'until ARGF.closed?; x = ARGF.file.readlines; puts "#{ x.size } lines"; puts x; ARGF.skip
> ; end' a b
>
> $ cat a
> 2 lines
> primo
> secundo
>
> $ cat b
> 1 lines
> PRIMO
> ```
>
> TBH I don't recommend to use -i either

Not really. I cannot understand that skip thing. I would expect the ARGF.file.readlines statement to remove the current file from ARGV since it is read. It is confusing.

This is a thing even without -i option. It is a thing whenever we need to know what file we work on.

Re #note-14: I guess it is a matter of taste but I really think -e hits a sweet spot. In the realm of sed/awk/grep -e is really a killer feature.

#17 - 10/23/2022 10:41 AM - nagachika (Tomoyuki Chikanaga)

- Backport changed from 2.7: REQUIRED, 3.0: REQUIRED, 3.1: REQUIRED to 2.7: REQUIRED, 3.0: REQUIRED, 3.1: DONE

ruby_3_1 5e25ba5d07d42f02485235e3962a4a28373c50e1 merged revision(s) 280b805d040fa537d5a459b40d4bfa6d49700905.