Ruby - Bug #10013

[CSV] Yielding all elements from a row

07/07/2014 01:03 PM - Gat (Dawid Janczak)

Status:	Closed		
Priority:	Normal		
Assignee:	kou (Kouhei Sutou)		
Target version:			
ruby -v:	ruby 2.2.0dev (2014-07-06 trunk 46722) [x86_64-linux]	**Backport:**	2.0.0: UNKNOWN, 2.1: UNKNOWN

Description

Let's say I have the following CSV file:
col1,col2,col3
1,2,3
4,5,6
(...)

I want to iterate over values yielding them to a block. I can do that like this:
CSV.foreach('file.csv') { |col1, col2, col3| print col2 + " " } # => "col2 2 5"
This works fine, but I would like to skip the headers:
CSV.foreach('file.csv', headers: true) { |col1, col2, col3| print col2 + " " } # => NoMethodError

CSV yields rows as arrays if headers option is not specified and destructuring works fine.
When headers option is specified however, CSV::Row objects are yielded instead and destructuring fails.

It would be nice to have both scenarios working in the same manner, but I don't know how to approach this. Calling to_a on yielded row (https://github.com/ruby/ruby/blob/trunk/lib/csv.rb#L1731) worked, but obviously this would break when people actually expect CSV::Row instance. Any ideas?

History

#1 - 07/07/2014 01:05 PM - Gat (Dawid Janczak)

Sorry, this should be in lib category, but I'm not able to change it now.

#2 - 07/07/2014 05:09 PM - avit (Andrew Vit)

Do you mean that it should consider the block arity to decide whether to yield a Row or destructure it into column parts? i.e.

```
CSV.foreach('file.csv', headers: true) do |col1, col2, col3|
  col1 == "1"
  col2 == "2"
  col3 == "3"
end
```

and also

```
CSV.foreach('file.csv', headers: true) do |row|
  row["col1"] == "1"
  row["col2"] == "2"
  row["col3"] == "3"
end
```

I think this would be too confusing and magical. What to do when the block arity doesn't match the count of row items? What about the case of a CSV with one column?

I haven't tried, but this might do what you expect (if you must use headers in the input):

```
CSV.foreach('file.csv', headers: true).lazy.map(&:to_a).each do |col1, col2, col3|
  col1 == "1"
  col2 == "2"
  col3 == "3"
end
```

I'll leave it for someone else to chime in whether there's a case for a special method here (e.g. "foreach_array").

#3 - 07/20/2014 07:28 PM - hsbt (Hiroshi SHIBATA)

- Category set to lib

- Status changed from Open to Assigned

- Assignee set to JEG2 (James Gray)

- Target version set to 2.2.0

#4 - 07/28/2014 02:32 PM - Gat (Dawid Janczak)

First of all sorry for the late answer Andrew.

Checking arity was one thing I was considering. You're right that the arity of the block might not match the number of items, but that works fine with arrays.

```
foo = [[1, 2], [3, 4]]
foo.each { |el| p el } # => prints [1, 2] then [3, 4]
foo.each { |el1, el2| p el2 } # => prints 2 then 4
foo.each { |el1, el2, el3| p el3 } # => prints nil then nil
```

This is basically the same behaviour as parallel assignment with more LVals than RVals.

Another thing I was considering was to always yield CSV::Row objects.

#5 - 01/05/2018 09:01 PM - naruse (Yui NARUSE)

- Target version deleted (2.2.0)

#6 - 03/08/2018 07:33 AM - mame (Yusuke Endoh)

- Assignee changed from JEG2 (James Gray) to kou (Kouhei Sutou)

@kou (Kouhei Sutou), could you check this ticket?

#7 - 03/10/2018 09:02 AM - kou (Kouhei Sutou)

- Status changed from Assigned to Closed

Thanks for your report.
I've fixed it at master: https://github.com/ruby/csv/commit/71d66af9824e4a30ed616668fc4ce8d9b68a5026