CommonRuby - Feature #9678

New heredoc syntax

03/26/2014 01:27 PM - alexeymuranov (Alexey Muranov)

Status:	Feedback	
Priority:	Normal	
Assignee:		
Target version:		

Description

For whatever it is worth, i've just had this idea of a new heredoc syntax for some programming language:

```
class C
  def f(x)
    print >>Message1 + >>Message2
Message1:
> Literal text
> ------------
> Here text.
>
Message2:
>> Text with interpolation and escapes
>> ----------------------------------
>> Here text with interpolation: #{ x }.
>>
  end
end
```

History

#1 - 03/26/2014 01:40 PM - nobu (Nobuyoshi Nakada)

And the expected output?

#2 - 03/26/2014 01:47 PM - alexeymuranov (Alexey Muranov)

Sorry, of course:

```
f(42)
```

i would expect to output

```
Literal text
------------
Here text.

Text with interpolation and escapes
-----------------------------------
Here text with interpolation: 42.
```

(There should be one more blank line at the end.)

#3 - 03/26/2014 01:52 PM - alexeymuranov (Alexey Muranov)

IMO, this would allow a program with heredocs to be easily readable without having to indent heredocs. Especially if editor's syntax highlighting will highlight > and >>.

#4 - 03/27/2014 01:42 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Feedback

Could you summarize your proposal concretely?

#5 - 03/27/2014 01:36 PM - alexeymuranov (Alexey Muranov)

I try to summarize again, hopefully better, and with a bit different syntax.

Example

```
class C
  def f(x)
    print <<Message1: + <<Message2: + <<Message3:
Message1:
> 1. Some text
>    without any intepolation (#{ x }) or escape sequences (\n)
>
Message2:
>> 2. Some text\
>>    with interpolation (#{ x }) and escape sequences.\n
Message3:
>> 3. Some mixed text: \
> #{ x } is replaced with
>> #{ x }
  end
end

C.new.f(42)
```

should print

```
1. Some text
   without any intepolation (#{ x }) or escape sequences (\n)

2. Some text   with interpolation (42) and escape sequences.

3. Some mixed text: #{ x } is replaced with
42
```

Explanation

In each line preceeded with single >, the leading > and one space are
removed, and the rest is interpreted as a single-quoted string:

```
> <line content>
```

is the same as

```
'<line content>
'
```

In each line preceeded with >>, the leading >> and one space are
removed, and the rest is interpreted as a double-quoted string:

```
>> <line content>
```

is the same as

```
"<line content>
"
```

Then lines are concatenated.

I hope my proposal is more clear now.

Update. The only difference in interpretation with quoted strings will probably be that quotes would not need to be escaped.

#6 - 03/27/2014 07:02 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

Is it possible to start the '>' anywhere besides the first char?

```
def my_method
  a = <<Message1:
  Message1:
    > Some
    > content
end
```

Or even something anonymous like:

```
def my_method
  a = <<:
    > Some
    > content
end
```

#7 - 03/27/2014 07:08 PM - alexeymuranov (Alexey Muranov)

Rodrigo Rosenfeld Rosas wrote:

> Is it possible to start the '>' anywhere besides the first char?

Good point, why not. My motivation was however to be able to write heredocs without indentation in a nice and clear way.

I have just thought of proposing exactly the same "anonymous" syntax.

#8 - 03/27/2014 07:12 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

What should the scope for interpolation be?

```
def my_method
  a = 1
  b = <<:message
end

a = 2
message:
>> #{a}

my_method # what is the output?
```

#9 - 03/27/2014 08:13 PM - alexeymuranov (Alexey Muranov)

Rodrigo Rosenfeld Rosas wrote:

> What should the scope for interpolation be?

I had thought about it, but i did not see a better option than to require the heredoc to be defined immediately after the line where it is used, as usual.

#10 - 03/27/2014 08:16 PM - alexeymuranov (Alexey Muranov)

Yes, it has to be defined immediately, otherwise it would be impossible to reuse the same heredoc "identifier" (Message:), and it may conflict with program identifiers.

#11 - 03/28/2014 10:56 AM - alexeymuranov (Alexey Muranov)

Some use cases for fun:

```
system <<:
> ./configure
> make
> make install

eval <<:
> 10.times do
>   puts <<:
>   > Who is the silly person who wrote this program?
> end
```

#12 - 04/04/2014 08:16 PM - alexeymuranov (Alexey Muranov)

I have just realized this would cause a difficulty with pasting code into irb. This would not be the first, however, -- the following cannot be pasted into irb either:

```
class C
  def f
    1
  end
end

puts C.new
  .f
```