Ruby trunk - Feature #9098

Indent heredoc against the left margin by default when "indented closing identifier" is turned on.

11/10/2013 06:18 AM - sikachu (Prem Sichanugrist)

Status:	Closed
Priority:	Normal
Assignee:	nobu (Nobuyoshi Nakada)
Target version:	

Description

tl;dr: I would like to port +String#strip_heredoc+ (http://api.rubyonrails.org/classes/String.html#method-i-strip_heredoc) from Rails and enable it on <<- heredoc.

Hi,

I've been using here document (heredoc) for a while, and I've found out that it's very annoying that the content is always treated as flushed left. Per syntax doc:

```
Note that the while the closing identifier may be indented, the content is
always treated as if it is flush left.  If you indent the content those spaces
will appear in the output.
```

So, this is the current result if you use heredoc in Ruby:

```
class FancyHello
  def self.hello
    puts <<-README.inspect
      Hello
        World!
    README
  end
end

FancyHello.hello # => "      Hello\n        World!\n"
```

In Rails, the core team has implemented +String#strip_heredoc+, which handles this scenario by remove leading white spaces from the string. However, not many user know that Rails has that method, and instead doing this to get around it:

```
class FancyHello
  def self.hello
    puts <<-README.inspect
Hello
  World!
    README
  end
end

FancyHello.hello # => "Hello\n  World!\n"
```

I really think that we could do better on this by removing the leading white spaces, matching +String#strip_heredoc+ method.

So, after the change, I want to be able to do this, and get this result:

```
class FancyHello
  def self.hello
    puts <<-README.inspect
      Hello
        World!
    README
  end
end

FancyHello.hello # => "Hello\n  World!\n"
```

Note: the behavior on <<HEREDOC will stay the same, as I feel like in that case you really want the input to be flushed left.

I'll write up a patch and submit it if you think this is a good idea. Please let me know any question or concern you may have.

Thank you,

Prem

Associated revisions

Revision 9a28a29b - 12/07/2015 02:39 PM - nobu (Nobuyoshi Nakada)

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52916 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 52916 - 12/07/2015 02:39 PM - nobu (Nobuyoshi Nakada)

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

Revision 52916 - 12/07/2015 02:39 PM - nobu (Nobuyoshi Nakada)

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

Revision 52916 - 12/07/2015 02:39 PM - nobu (Nobuyoshi Nakada)

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

Revision 52916 - 12/07/2015 02:39 PM - nobu (Nobuyoshi Nakada)

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

Revision 52916 - 12/07/2015 02:39 PM - nobu (Nobuyoshi Nakada)

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

Revision 10952d16 - 12/08/2015 03:08 AM - nobu (Nobuyoshi Nakada)

NEWS: indented heredoc [ci skip]

- NEWS: mention about indented here document. [Feature #9098]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@52935 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 52935 - 12/08/2015 03:08 AM - nobu (Nobuyoshi Nakada)

NEWS: indented heredoc [ci skip]

- NEWS: mention about indented here document. [Feature #9098]

Revision 52935 - 12/08/2015 03:08 AM - nobu (Nobuyoshi Nakada)

NEWS: indented heredoc [ci skip]

- NEWS: mention about indented here document. [Feature #9098]

Revision 52935 - 12/08/2015 03:08 AM - nobu (Nobuyoshi Nakada)

NEWS: indented heredoc [ci skip]

- NEWS: mention about indented here document. [Feature #9098]

Revision 52935 - 12/08/2015 03:08 AM - nobu (Nobuyoshi Nakada)

NEWS: indented heredoc [ci skip]

- NEWS: mention about indented here document. [Feature #9098]

Revision 52935 - 12/08/2015 03:08 AM - nobu (Nobuyoshi Nakada)

NEWS: indented heredoc [ci skip]

- NEWS: mention about indented here document. [Feature #9098]

History

#1 - 11/13/2013 03:21 PM - drbrain (Eric Hodel)

- Status changed from Open to Assigned

- Assignee changed from sikachu (Prem Sichanugrist) to matz (Yukihiro Matsumoto)

- Target version set to Next Major

Why not make your content flush left?

#2 - 11/14/2013 03:26 AM - sikachu (Prem Sichanugrist)

drbrain (Eric Hodel) I could, but then the code would look ugly when the non-heredoc section is well-indented, and especially if the heredoc is pretty long. Consider this example code:

```
module Rails
  class CommandsTasks # :nodoc:
    attr_reader :argv

    HELP_MESSAGE = <<-EOT
Usage: rails COMMAND [ARGS]

The most common rails commands are:
 generate    Generate new code (short-cut alias: "g")
 console     Start the Rails console (short-cut alias: "c")
 server      Start the Rails server (short-cut alias: "s")
 dbconsole   Start a console for the database specified in config/database.yml
             (short-cut alias: "db")
 new         Create a new Rails application. "rails new my_app" creates a
             new application called MyApp in "./my_app"

In addition to those, there are:
 application  Generate the Rails application code
 destroy      Undo code generated with "generate" (short-cut alias: "d")
 plugin new   Generates skeleton for developing a Rails plugin
 runner       Run a piece of code in the application environment (short-cut alias: "r")

All commands can be run with -h (or --help) for more information.
EOT

    COMMAND_WHITELIST =
%(plugin generate destroy console server dbconsole application runner new version help)
```

Compared to this:

```
module Rails
  class CommandsTasks # :nodoc:
    attr_reader :argv

    HELP_MESSAGE = <<-EOT
      Usage: rails COMMAND [ARGS]

      The most common rails commands are:
       generate    Generate new code (short-cut alias: "g")
       console     Start the Rails console (short-cut alias: "c")
       server      Start the Rails server (short-cut alias: "s")
       dbconsole   Start a console for the database specified in config/database.yml
                   (short-cut alias: "db")
       new         Create a new Rails application. "rails new my_app" creates a
                   new application called MyApp in "./my_app"

      In addition to those, there are:
```

```
      application  Generate the Rails application code
      destroy      Undo code generated with "generate" (short-cut alias: "d")
      plugin new   Generates skeleton for developing a Rails plugin
      runner       Run a piece of code in the application environment (short-cut alias: "r")

     All commands can be run with -h (or --help) for more information.
   EOT

   COMMAND_WHITELIST =
%(plugin generate destroy console server dbconsole application runner new version help)
```

From the code example, you could see that it looks much nicer when it's properly indented based on the level. The heredoc doesn't feel foreign and stand out-of-place anymore.

#3 - 11/14/2013 08:08 AM - drbrain (Eric Hodel)

I can see the second is worse, not better, because it runs over 80 columns on more lines which wrap in my editor.

#4 - 11/14/2013 08:10 AM - sikachu (Prem Sichanugrist)

I can see that. I actually just indented the example in without looking at the line width. My bad. :)

If the second example is *not* over 80 columns, would that still be consider worse for you?

#5 - 11/14/2013 10:45 AM - drbrain (Eric Hodel)

I prefer Avdi's idea in [ruby-core:58322] as it does not introduce incompatibility.

#6 - 11/14/2013 10:53 AM - avdi (Avdi Grimm)

On Sat, Nov 9, 2013 at 4:18 PM, sikachu (Prem Sichanugrist) s@sikac.hu
wrote:

I've been using here document (heredoc) for a while, and I've found out

> that it's very annoying that the content is always treated as flushed left.

This has actually bugged me for a very long time. Other languages have
triple-quoted strings where the indent of the opening triple-quote is taken
into account, and automatically stripped from the resulting text. In Ruby
if I don't want leading spaces I have to break the visual flow of my
indentation (or resort to something like #strip_heredoc).

I don't agree with changing the existing heredocs to have new semantics.
But I could see adding e.g. a "squiggly-heredoc":

```
class Foo
  BAR = <<~EOF
  This line is
  cleanly indented
  EOF
end
Foo::BAR # => "This line is\ncleanly indented"
```

Note that the way I imagine it, this would work a little different than
triple-quoted strings in other languages, since the indentation wouldn't be
determined by the location of the opening signifier. Instead, it would be
determined by indentation of the first non-whitespace character inside the
heredoc.
--
Avdi Grimm
http://avdi.org

I only check email twice a day. to reach me sooner, go to
http://awayfind.com/avdi

#7 - 11/14/2013 12:10 PM - sikachu (Prem Sichanugrist)

Oh, that's an awesome suggestion, @avdi. I like it. That way, it wouldn't be introducing backward incompatibility. Maybe we can target next minor (since it's a new feature) as well?

drbrain (Eric Hodel), so I think I'll want to change this ticket to be adding <<~ to do this functionality instead. Should I go ahead and try to implement it, or do I have to wait for matz (Yukihiro Matsumoto) to +1?

#8 - 11/14/2013 01:01 PM - drbrain (Eric Hodel)

- Target version changed from Next Major to 2.6

Creating a patch will help, but it can't be committed without the approval of matz.

#9 - 11/14/2013 07:41 PM - rosenfeld (Rodrigo Rosenfeld Rosas)

+1 to Avdi's idea. I mostly miss this when I'm writing my specs. Sometimes I have to compare some processed HTML with the expected result and it's not easy to write the expected string without compromising indentation. Usually I end up with something like:

```
[
  'First line',
  'Second line',
  ...
].join "\n"
```

Having something like what Avdi described would help me a lot to write some of my specs.

#10 - 11/15/2013 01:07 AM - sikachu (Prem Sichanugrist)

Sweet. I'll go ahead and start cooking up the patch. Thanks all for the preliminary feedback.

#11 - 04/20/2015 11:42 AM - bjmllr (Ben Miller)

This feature request has been silent for a year, so I submitted a patch for it: https://github.com/ruby/ruby/pull/878

#12 - 11/09/2015 06:36 AM - ko1 (Koichi Sasada)

Discussion: https://docs.google.com/document/d/1D0Eo5N7NE_unlySOKG9lVj_eyXf66BQPM4PKp7NvMyQ/pub

Feel free to continue discussion on this ticket.

#13 - 11/09/2015 06:52 AM - matz (Yukihiro Matsumoto)

Accepted. We have to define the behavior on (hard) tabs. As a UNIX user, I'd like to tak 8 spaces for tab stops.

Matz.

#14 - 11/09/2015 03:00 PM - Ajedi32 (Andrew M)

What do you mean? Isn't the number of spaces a tab is equivalent to only relevant when the first line is indented with tabs, while the following lines are indented with spaces (or vice-versa)?

Shouldn't it just throw an error in that case?

#15 - 11/28/2015 01:31 AM - bjmllr (Ben Miller)

- File dedent_heredoc.patch added

I updated the patch to treat tabs as equal to 8 spaces.

This seems to work fine with the indentation produced by popular text editors (only tabs, only spaces, or tabs-then-spaces), with only one corner case I noticed, which is demonstrated in test_dedented_heredoc_with_inconsistent_indentation_preserves_tab (in test/ruby/test_syntax.rb), in which case we try backing up to the previous tab stop and starting over, to avoid breaking the hard tab. Do we need to handle spaces-then-tabs?

I attached the patch here so that more people are able to see it.

#16 - 11/28/2015 03:14 AM - Ajedi32 (Andrew M)

I'm confused. Why are tabs being treated as equivalent to spaces at all? E.g. If I write:

```
def hello
  puts <<~README.inspect
<tab>Hello

<space><space><space><space><space><space><space><space>World!
    README
  end
end
```

Are you saying that should be accepted by the compiler? Why? Why should that be any less invalid than:

```
def hello
```

```
  puts <<~README.inspect
<tab>Hello

<space><space><space><space>World!
    README
  end
end
```

or

```
def hello
  puts <<~README.inspect
<space><space><space><space>Hello

<space><space>World!
    README
  end
end
```

Shouldn't we just throw an error in all of those cases? Is there ever a legitimate reason why you'd want to allow inconsistent indentation in one of these blocks? What happens when someone has their editor set to display tabs as 4 spaces, and writes:

```
def hello
  puts <<~README.inspect
<space><space><space><space>Hello

<tab>World!
    README
  end
end
```

Why should that result in:

```
hello #=> "    Hello\n\n\tWorld!"
```

I certainly wouldn't expect that result intuitively. In such a case, wouldn't a well-written error message explaining that I'm mixing tabs and spaces be much more helpful for me as a developer?

(Note: I also cross-posted to the PR. I'm not really sure where the canonical place is for discussion on this feature.)

#17 - 11/28/2015 03:37 AM - matz (Yukihiro Matsumoto)

Because we can not distinguish tabs and spaces on most editors, while we can distinguish 4 spaces and 8 spaces.
It's weird but has historical reason.

Matz.

#18 - 11/28/2015 05:37 PM - Ajedi32 (Andrew M)

Right. I guess what I'm saying is that tabs shouldn't be treated as equivalent to spaces at all, since the number of spaces they are "equivalent" to is different for every developer, depending on what they have their editor set to display.

This can and will result in unexpected behavior (aka bugs) when tabs and spaces are mixed in indentation by mistake. I'd much rather these bugs be caught by the compiler and brought to the developer's attention with a clearly-worded error message then simply ignored and left for the developer to find out about later.

Tabs should be treated as tabs, and spaces should be treated as spaces. I'm not really sure what the argument is for treating one character as equivalent to another in the first place. Is there any situation where a developer might want that behavior? What's the use-case here? I honestly can't think of one.

To me, this just feels like a way to make it easy for developers to unknowingly create bugs in their code.

#19 - 12/01/2015 06:20 AM - nobu (Nobuyoshi Nakada)

"8 spaces for tab stops" is not enough to make the spec.

When tabs and spaces are mixed, e.g., there are"\t"*2+"2 tabs" line and " "*10+"10 spaces" line, what should be expected?

1. dedent at the greatest indent column where all lines have a white space.

   a tab which includes the column will remain, but spaces between previous tab stop and the column will be removed.
   this is the implementation by bjmllr (Ben Miller), and results
   "\t"+"2 tabs" line and " "*2+"10 spaces" line.

2. dedent at the greatest indent column exclusively tabs

a tab which includes the column will remain, and also spaces before the column will be removed.
this is like kill-rectangle of Emacs, and I've thought this, and results
"\t"+"2 tabs" line and "10 spaces" line.

3. expand tabs first

   tabs are expanded first before dedent, and results
   " "*6+"2 tabs" line and "10 spaces" line.

4. or, like Andrew M claims, dedent of byte-wise common parts can be an option, I think now.

#20 - 12/07/2015 06:45 AM - matz (Yukihiro Matsumoto)

- Assignee changed from matz (Yukihiro Matsumoto) to nobu (Nobuyoshi Nakada)

Alternative 2 looks good to me.

Matz.

#21 - 12/07/2015 02:40 PM - nobu (Nobuyoshi Nakada)

- Status changed from Assigned to Closed

Applied in changeset r52916.

---

parse.y: indented hereoc

- parse.y: add heredoc <<~ syntax. [Feature #9098]

#22 - 12/07/2015 03:19 PM - Ajedi32 (Andrew M)

Ah, for some reason I was under the impression that the plan was to use the indentation from the first non-blank line to determine how much to dedent the entire string.

If the plan instead is to use the least-indented non-blank line for that, then the other options make much more sense IMO since there's no way to determine definitively whether leading indention was intended to be removed by Ruby vs used in the string. I actually like this way better come to think of it, since it makes it possible to indent the first line in the string, whereas with the method I was envisioning that would be impossible.

For the record, I'm still in favor of option 4 (or rather, some combination of 4 and 2 which throws an error when their behaviors differ) as I think it's the least likely to result in unexpected behavior for developers, but if Matz has made up his mind I guess there's not much I can do about it. I suppose I'll just have to add a rule to my linter to help prevent those kinds of mistakes.

The results from methods 1, 2, and 4 should always be the same in practice. If they're not, it's almost certainly a case of the developer doing something they didn't intend (a bug).

#23 - 12/08/2015 12:10 AM - usa (Usaku NAKAMURA)

Hey, nobu, write NEWS!!

#24 - 12/08/2015 03:43 AM - mame (Yusuke Endoh)

This change actually introduces a syntactic incompatibility. Is this okay? I have not seen any real-life use-case that suffers from this incompatibility, though.

```
def foo
  42
end
BAR = -10
p foo <<~BAR
```

```
$ ruby -v
ruby 2.2.3p173 (2015-08-18 revision 51636) [x86_64-linux]

$ ruby t.rb
21504

$ ./miniruby -v
ruby 2.3.0dev (2015-12-08 master 52936) [x86_64-linux]

$ ./miniruby t.rb
t.rb:5: can't find string "BAR" anywhere before EOF
```

```
t.rb:5: syntax error, unexpected end-of-input
```

Ruby syntax is a chaos.

--
Yusuke Endoh mame@ruby-lang.org

Files

dedent_heredoc.patch	12.8 KB	11/28/2015	bjmllr (Ben Miller)