# Ruby - Bug #1801

## parse error on variable/method collision

07/21/2009 11:13 PM - coatl (caleb clausen)

| | | | |
|---|---|---|---|
| **Status:** | Rejected | | |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Target version:** | | | |
| **ruby -v:** | 1.9.1 | **Backport:** | |

**Description**

=begin
Consider this ruby program:

def a; 11; end
def e; 44; end
def ep(x); 99 end
ep=556
p(ep (e) /a)

The correct output is 9. Actual output is 99.

The last line should be parsed like this: p(ep((e)/a))
However, it appears to get parsed like this: p((ep(e))/a)

In other words, the /a should be inside the arg list to ep(), however, it actually gets put outside.

The rules that apply to this case are a little involved. Parentheses following a method name are usually the argument list of the method. However, if a space separates the parentheses from the method name, they are considered grouping parentheses instead. (Except if there are any commas or splats directly within the parens, in which case they go back to being argument parens. Not the case here.) The example above appears to violate those rules, the inner parens should be considered grouping parens, whereas they're actually treated as arg list parens.

Deleting the assignment to the local variable ep makes this problem go away. It's as if the presence of a local variable of the same name as the method confuses the parser, even tho it's clearly a method call, since it has an argument list.

This appears to be present in every version of MRI ruby I can get ahold of (on ruby-versions.net) except 1.0 and (curiously) 1.7.1. It's also found in jruby.

Rubinius has an even more severe version of the problem. It outputs 50, which indicates that it not only misparses, it also evaluates the expression 'ep (a)' as a local variable, completely ignoring the arg list!
=end

### History

**#1 - 07/21/2009 11:23 PM - matz (Yukihiro Matsumoto)**

=begin
Hi,

In message "Re: [ruby-core:24476] [Bug #1801] parse error on variable/method collision"
on Tue, 21 Jul 2009 23:13:15 +0900, caleb clausen redmine@ruby-lang.org writes:

|Consider this ruby program:
|
|def a; 11; end
|def e; 44; end
|def ep(x); 99 end
|ep=556
|p(ep (e) /a)
|
|The correct output is 9. Actual output is 99.
|
|The last line should be parsed like this: p(ep((e)/a))
|However, it appears to get parsed like this: p((ep(e))/a)
|

|In other words, the /a should be inside the arg list to ep(), however, it actually gets put outside.

It's intentional.  We cannot always fulfill everyone's expectation
from various background.  When you claim "should" in your bug report,
you have to include your rationale more than mere expectation in the
report.

|Rubinius has an even more severe version of the problem. It outputs 50, which indicates that it not only misparses, it also evaluates the expression 'ep (a)' as a local variable, completely ignoring the arg list!

It seems to be a bug in Rubinius.  You should report to its maintainers.

```
matz.
```

=end

#### #2 - 07/22/2009 12:47 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:24478] Re: [Bug #1801] parse error on variable/method collision"
on Tue, 21 Jul 2009 23:40:20 +0900, Caleb Clausen caleb@inforadical.net writes:

|When I showed this to you at rubyconf, you confirmed that it was a bug.

Ah, sorry for my weak memory.  I remember the conversation now.
Now it's a issue of trade off, between compatibility and
intuitiveness.

```
matz.
```

=end

#### #3 - 08/22/2009 07:27 AM - coatl (caleb clausen)

=begin
I have looked for examples of where this behavior is used in the wild, but I can't find any. Maybe I imagined them.
=end

#### #4 - 08/22/2009 10:00 AM - nobu (Nobuyoshi Nakada)

=begin
Hi,

At Wed, 22 Jul 2009 00:47:03 +0900,
Yukihiro Matsumoto wrote in [ruby-core:24480]:

> Ah, sorry for my weak memory.  I remember the conversation now.
> Now it's a issue of trade off, between compatibility and
> intuitiveness.

At least, this difference doesn't seem like intentional to me.
Is it?

```
$ ./ruby -v -e 'def a; 11; end
def e; 44; end
def ep(x); 99 end
ep = 556
p ep (e)/a'
ruby 1.9.2dev (2009-08-22 trunk 24620) [i386-darwin9.0]
9

$ ./ruby -v -e 'def a; 11; end
def e; 44; end
def ep(x); 99 end
p ep (e)/a'
ruby 1.9.2dev (2009-08-22 trunk 24620) [i386-darwin9.0]
-e:4: warning: (...) interpreted as grouped expression
99
```

--
Nobu Nakada

=end

#### #5 - 06/26/2011 02:04 PM - naruse (Yui NARUSE)

*- Status changed from Open to Rejected*

This is intentional.