Ruby trunk - Feature #10617

Change multiple assignment in conditional from parse error to warning

12/18/2014 11:07 AM - recursive-madman (Recursive Madman)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

There is currently an inconsistency between regular and multiple assignment in conditionals.
Regular assignment causes a **warning**, multiple assignment causes a **parse error**.

The historical reason for this is that in 1.8 multiple assignment would always return an Array, but since 1.9 it returns whatever the RHS evaluates to.

Examples:

```
a, b = nil  #=> nil
a, b = []   #=> []      (but a and b are both nil)
a, b = 1,2  #=> [1, 2]
```

Since multiple assignment behavior has changed, it makes sense to remove the (artificial) parse error for multiple assignments.

That makes it possible to test the return value of a method used for multiple assignment without having to use a temporary variable.

Example:

```
# CURRENTLY WORKING CODE:
tmp = some_method_returning_array_or_nil
a, b = tmp
if tmp
  # method returned an array (possibly empty)
else
  # method returned nil.
end

# PROPOSED WORKING CODE:
if(a, b = some_method_returning_array_or_nil)
  # method returned an array (possibly empty)
else
  # method returned nil
end
```

(the parenthesis are needed due to LALR limitations, as discussed in #10450)

Attached is a patch that does the necessary change.

Associated revisions

Revision 134d1ce8 - 04/13/2016 05:36 AM - nobu (Nobuyoshi Nakada)

parse.y: massign in cond

- parse.y (assign_in_cond): allow multiple assignment in conditional expression. [Feature #10617]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@54558 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 54558 - 04/13/2016 05:36 AM - nobu (Nobuyoshi Nakada)

parse.y: massign in cond

- parse.y (assign_in_cond): allow multiple assignment in conditional expression. [Feature #10617]

Revision 54558 - 04/13/2016 05:36 AM - nobu (Nobuyoshi Nakada)

parse.y: massign in cond

- parse.y (assign_in_cond): allow multiple assignment in conditional expression. [Feature #10617]

Revision 54558 - 04/13/2016 05:36 AM - nobu (Nobuyoshi Nakada)

parse.y: massign in cond

- parse.y (assign_in_cond): allow multiple assignment in conditional expression. [Feature #10617]

Revision 54558 - 04/13/2016 05:36 AM - nobu (Nobuyoshi Nakada)

parse.y: massign in cond

- parse.y (assign_in_cond): allow multiple assignment in conditional expression. [Feature #10617]

History

#1 - 12/30/2014 12:04 AM - recursive-madman (Recursive Madman)

I'm not sure if it's decent to push this - I understand 2.2.0 release is taking up a lot of time on the core devs side and holidays are also in progress in some parts of the world, so no hurry here.

Anyway, I feel I should ask:

- are there any known reasons not to accept this patch?
- does the issue need more clarification? #10450 has more information (I know that issue's discussion got a bit heated and confusing, which is why I opened this one to limit it to the actual change that was requested)

If there's anything more I should do to make this go through (maybe write a test or something?), I'm willing to do.

Ahoy.

#2 - 12/31/2014 07:58 AM - duerst (Martin Dürst)

Just in private:

- Adding some tests to the patch is a good idea.

- If there's no more action, I'd wait for the next call
  for proposals for features in Ruby 2.3,
  then produce a slide summarizing your proposal
  (see e.g. Normalization.pdf on
  https://bugs.ruby-lang.org/issues/10084).
  Such calls normally occur somewhere in the middle
  of the year.

Regards, Martin.

#3 - 12/10/2015 06:05 PM - bughit (bug hit)

Why hasn't this been accepted? There is no good reason that an expression that could be nil should not be testable.

#4 - 01/24/2016 03:42 AM - avit (Andrew Vit)

I think it's too confusing or ambiguous to allow multiple assignment in conditional.
It's very easy to just do the assignment on a previous line.

This could be confusing, especially if it comes from splat values:

```
a, b = [false, false]
```

This has been rejected before:
https://bugs.ruby-lang.org/issues/10450

#5 - 01/24/2016 04:35 AM - bughit (bug hit)

Andrew Vit wrote:

> I think it's too confusing

There is nothing intrinsically confusing about testing an expression that may be truthy or falsy. One would only be confused if one did not know that it could be either, in which case one should find out and so unconfuse oneself.

The restriction is not there to deal with confusion, it's historical cruft from the time when multiple assignment expression was always truthy, as Yusuke Endoh explained:

> I'll tell you guys the rationale: multiple assignment always used to return an array.
>
> ```
> $ ./ruby -ve 'p((a, b = nil))'
> ruby 1.8.7 (2013-06-27 patchlevel 374) [x86_64-linux]
> [nil]
> ```
>
> This behavior changed since 1.9.0, so the restriction is indeed meaningless now.

#6 - 03/01/2016 11:00 AM - mikecmpbll (Mike Campbell)

I find this really unintuitive and share some of 'bug hit's exasperation. It's clear that the return value, like everything, is either truthy or falsey, why treat this differently to everything else in Ruby?

It's particularly annoying because it prevents well known patterns from being used with tuples, for instance:

```
while (a = queue.pop(true) rescue nil)
  # do something
end
```

I see that quite a lot for exhausting a queue. However, when you want to use the same pattern with tuples Ruby arbitrarily restricts you:

```
while (a, b = queue.pop(true) rescue nil)
  # do something
end
# SyntaxError: (irb):4: multiple assignment in conditional
```

The work-around with the throwaway variable is simple enough, but unintuitive. I understand from reading the other bug issue that this has historical relevance, but clearly this is no longer the case so I don't see any reason to object to it's removal.

#7 - 04/13/2016 05:36 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r54558.

---

parse.y: massign in cond

- parse.y (assign_in_cond): allow multiple assignment in conditional expression. [Feature #10617]

#8 - 10/28/2017 11:30 AM - Eregon (Benoit Daloze)

This is now allowed, but produces a warning:

```
$ ruby -e 'if (a, b = 1, 2); puts "y"; end'
-e:1: warning: found = in conditional, should be ==
y

$ ruby -W0 -e 'if (a, b = 1, 2); puts "y"; end'
y
```

Is that expected?

It seems to only happen with literals though:

```
$ ruby -e 'if (a, b = 1, 2); puts "y"; end'
-e:1: warning: found = in conditional, should be ==
y

$ ruby -e 'if (a, b = 1, 2.itself); puts "y"; end'
y
```

Files

0001-turn-parse-error-on-multiple-assignment-into-warning.patch	752 Bytes	12/18/2014	recursive-madman (Recursive Madman)