# Ruby trunk - Bug #10450

## multiple assignment in conditional

10/29/2014 03:19 PM - bughit (bug hit)

| | | | |
|---|---|---|---|
| **Status:** | Rejected | | |
| **Priority:** | Normal | | |
| **Assignee:** | | | |
| **Target version:** | | | |
| **ruby -v:** | ruby 2.1.4p265 (2014-10-27 revision 48166) [x86_64-linux] | **Backport:** | 2.0.0: UNKNOWN, 2.1: UNKNOWN |

**Description**

multiple assignment is an expression whose value can be truthy (array) or falsy (nil, false), so why is there a restriction on its use in conditionals? A warning perhaps is justified, but a syntax error, why?

```
irb(main):001:0> if a, b = nil then 1 else 0 end
SyntaxError: (irb):1: syntax error, unexpected ',', expecting keyword_then or ';' or '\n'
if a, b = nil then 1 else 0 end
^
(irb):1: syntax error, unexpected keyword_then, expecting end-of-input
if a, b = nil then 1 else 0 end
^
from /home/alex/.rbenv/versions/2.1.4/bin/irb:11:in <main>'
irb(main):002:0> if (a, b = nil) then 1 else 0 end
SyntaxError: (irb):2: multiple assignment in conditional
from /home/alex/.rbenv/versions/2.1.4/bin/irb:11:in'
irb(main):003:0> (a, b = nil) ? 1 : 0
SyntaxError: (irb):3: multiple assignment in conditional
from /home/alex/.rbenv/versions/2.1.4/bin/irb:11:in `'
irb(main):004:0> (a, b = nil)
=> nil
```

## History

**#1 - 10/30/2014 08:56 AM - jballanc (Joshua Ballanco)**

Which of the multiple values assigned would you have used as the test for the conditional?

```
if (a, b = true, false)
  puts "Should this run?"
else
  puts "Or this?"
end
```

**#2 - 11/01/2014 10:03 PM - bughit (bug hit)**

Joshua Ballanco wrote:

> Which of the multiple values assigned would you have used as the test for the conditional?
>
> ```
> if (a, b = true, false)
>   puts "Should this run?"
> else
>   puts "Or this?"
> end
> ```

multiple assignment is an expression whose value can be truthy (array) or falsy (nil, false)

**#3 - 11/03/2014 10:11 AM - jballanc (Joshua Ballanco)**

Yes, but consider:

```
if a = (b, c = false, false)
  puts "a is #{a}, b is #{b}, c is #{c} and everything together is true"
end
```

So if you were hoping to use multiple assignment to check the &&-ed boolean values of the individual conditions (or even the ||-ed value), then you would be rudely surprised. In fact:

```
def gimmie_nil
  nil
end

if a = (b, c = gimmie_nil(), gimmie_nil())
  puts "Still true..."
end
```

...there is no way that the conditional would ever evaluate to false!

**#4 - 11/03/2014 02:42 PM - bughit (bug hit)**

Joshua Ballanco wrote:

> Yes, but consider:
>
> ```
> if a = (b, c = false, false)
>   puts "a is #{a}, b is #{b}, c is #{c} and everything together is true"
> end
> ```
>
> So if you were hoping to use multiple assignment to check the &&-ed boolean values of the individual conditions (or even the ||-ed value), then you would be rudely surprised. In fact:
>
> ```
> def gimmie_nil
>   nil
> end
>
> if a = (b, c = gimmie_nil(), gimmie_nil())
>   puts "Still true..."
> end
> ```
>
> ...there is no way that the conditional would ever evaluate to false!

I am not hoping for anything, nor advocating for the change in the semantics of the multiple assignment expression.  As I pointed out twice it is already an expression that can be truthy or falsy, and it should be usable wherever other expressions are.

**#5 - 11/03/2014 06:59 PM - jballanc (Joshua Ballanco)**

Ok, yes...but if your intention with this feature request is simply to treat the multiple-assignment statement as a whole as the test condition, you can already do that like so:

```
if _ = (a, b = *multi_ret_val_meth)
  puts "Multi-return statement is true"
end
```

Personally, I think this is a small price to pay (one underscore, one equals, two parens) to get the semantics you desire. The error condition for a naked multi-assignment in a conditional seems like a sensible guard for newcomers who don't realize that multi-assignment can have odd semantics depending on the rhs structure.

**#6 - 12/06/2014 07:14 PM - olleicua (Antha Auciello)**

It seems wierd to me that this is a syntax error.

```
def foo
  a,b = (1,2)
end
if foo
  put 'hello'
end
```

Is fine but removing the abstraction is a syntax error.  This seems like.. not the point of syntax.  A Warning makes more sense IMO.

**#7 - 12/07/2014 03:34 PM - matz (Yukihiro Matsumoto)**

*- Status changed from Open to Rejected*

It's a limitation of LALR syntax defined by yacc.
Wrap the multiple assignment in parentheses.

```
if (a,b = 1,2)
  puts 'hello'
```

```
end
```

But I am afraid it's meaningless, since multiple assignment always return an array as its value.

Matz.

**#8 - 12/07/2014 07:26 PM - bughit (bug hit)**

Yukihiro Matsumoto wrote:

> But I am afraid it's meaningless, since multiple assignment always return an array as its value.
>
> Matz.

It's absolutely not meaningless, the above assertion is false, multiple assignment does not always return an array.

For the fourth time, multiple assignment is an expression whose value can be truthy (array) or falsy (a, b = nil)

It's therefore perfectly reasonable to test in a conditional.

```
if a, b = method_returning_array_or_nil
  foo
else
  bar
end
```

**#9 - 12/08/2014 09:42 AM - javawizard (Alex Boyd)**

Multiple assignment returns the object on the right hand side, *before* to_ary is called on it:

```
irb(main):006:0> class A; def to_ary; [1, 2]; end; end
=> :to_ary
irb(main):007:0> a, b = A.new
=> #<A:0x007fe273da6260>
```

It will therefore, as bug hit mentions, return nil if the right hand side is nil.

But even wrapping in parentheses doesn't work (ruby -v: ruby 2.2.0dev (2014-12-08 trunk 48728) [x86_64-linux]):

```
irb(main):001:0> if (a, b = nil) then 'yes' else 'no' end
SyntaxError: (irb):1: multiple assignment in conditional
    from bin/irb:11:in `<main>'
```

This patch:

```
--- parse.y
+++ parse.y
@@ -9432,10 +9432,6 @@ static int
 assign_in_cond(struct parser_params *parser, NODE *node)
 {
     switch (nd_type(node)) {
-      case NODE_MASGN:
-       yyerror("multiple assignment in conditional");
-       return 1;
-
       case NODE_LASGN:
       case NODE_DASGN:
       case NODE_DASGN_CURR:
```

seems to work (but, of course, still requires parentheses):

```
irb(main):004:0> if (a, b = nil) then 'yes' else 'no' end
=> "no"
irb(main):005:0> if (a, b = [1, 2]) then 'yes' else 'no' end
=> "yes"
```

I don't myself have any huge interest in being able to do this, but now I'm curious: why was this prohibited in the first place? (There could very well be some unexpected consequence of allowing this that I've overlooked...)

**#10 - 12/09/2014 04:27 AM - bughit (bug hit)**

Yukihiro Matsumoto wrote:

> But I am afraid it's meaningless, since multiple assignment always return an array as its value.

Since the rejection reason is invalid, why would you not reopen?  Is there's some other reason?

**#11 - 12/14/2014 08:34 PM - bughit (bug hit)**

bug hit wrote:

> Yukihiro Matsumoto wrote:
>
> > But I am afraid it's meaningless, since multiple assignment always return an array as its value.
>
> Since the rejection reason is invalid, why would you not reopen?  Is there's some other reason?

Can some one on the ruby core give a reason for this?  Bugs should not be rejected with invalid excuses.

**#12 - 12/16/2014 08:06 AM - duerst (Martin Dürst)**

bug hit wrote:

> bug hit wrote:
>
> > Yukihiro Matsumoto wrote:
> >
> > > But I am afraid it's meaningless, since multiple assignment always return an array as its value.
> >
> > Since the rejection reason is invalid, why would you not reopen?  Is there's some other reason?
>
> Can some one on the ruby core give a reason for this?  Bugs should not be rejected with invalid excuses.

Yukihiro Matsumoto also wrote:

> It's a limitation of LALR syntax defined by yacc.

Bugs have to be rejected if there's a valid reason for rejection, even if another reason also was given that may or may not be valid.

Maybe if you can show how to implement what you want (i.e. how to get around the LALR limitation), then you can open another issue.

**#13 - 12/16/2014 03:50 PM - bughit (bug hit)**

Martin Dürst wrote:

> bug hit wrote:
>
> > bug hit wrote:
> >
> > > Yukihiro Matsumoto wrote:
> > >
> > > > But I am afraid it's meaningless, since multiple assignment always return an array as its value.
> > >
> > > Since the rejection reason is invalid, why would you not reopen?  Is there's some other reason?
> >
> > Can some one on the ruby core give a reason for this?  Bugs should not be rejected with invalid excuses.
>
> Yukihiro Matsumoto also wrote:
>
> > It's a limitation of LALR syntax defined by yacc.
>
> Bugs have to be rejected if there's a valid reason for rejection, even if another reason also was given that may or may not be valid.
>
> Maybe if you can show how to implement what you want (i.e. how to get around the LALR limitation), then you can open another issue.

You also don't read before posting, ok let me redigest for you.  The LALR limitation is about use without parens, that's not what this bug is about.  If it worked with parens I would not have even bothered filing it.  Alex Boyd already posted a patch that lifts the unreasonable restriction.  As of now, no legitimate (non false) reason has been given for rejection.

**#14 - 12/16/2014 05:01 PM - recursive-madman (Recursive Madman)**

It is indeed weird, that this isn't allowed.

A few lines below where the parse error is thrown, there is

```
parser_warn(node->nd_value, "found = in conditional, should be ==");
```

for regular conditionals.

It seems weird to me, that one is a warning and the other is an error, when the parser has at that point already recognized the multiple assignment correctly.

The warning suggests that assignment in conditionals is discouraged. That doesn't explain though, why one is an error and the other is not.

**#15 - 12/16/2014 05:18 PM - mame (Yusuke Endoh)**

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.

(It is really great that no one notices this; we truly graduated from 1.8!)

--
Yusuke Endoh mame@ruby-lang.org

**#16 - 12/16/2014 06:00 PM - recursive-madman (Recursive Madman)**

So... that means this issue will be reopened and Alex Boyd's patch applied?

**#17 - 12/16/2014 06:04 PM - javawizard (Alex Boyd)**

Ah, historical holdover... Those are fun. :-)

That being the case, any chance of getting my patch applied?

Martin Dürst wrote:

> Bugs have to be rejected if there's a valid reason for rejection, even if another reason also was given that may or may not be valid.
>
> Maybe if you can show how to implement what you want (i.e. how to get around the LALR limitation), then you can open another issue.

To clarify bug hit's point, the LALR limitation was only one facet of this ticket. The other one was permitting multiple assignment *with* the requisite use of parentheses, which the parser is perfectly capable of but which was being filtered out by hand in assign_in_cond due to what Yusuke Endoh explained was historical behavior that no longer exists. The patch I included in my comment would be sufficient to lift this restriction.

**#18 - 12/16/2014 10:48 PM - javawizard (Alex Boyd)**

I've submitted https://github.com/ruby/ruby/pull/786 to keep this on the radar.

**#19 - 12/17/2014 06:46 AM - duerst (Martin Dürst)**

It seems that both Matz and I got confused. That's not too surprising because the first example has no parentheses.

I suggest the following:

- Create a new issue that strictly concentrates on the case WITH parentheses.
- Beef up the pull request with the necessary changes to tests.

Please note that Matz can still reject the issue if he doesn't like it (e.g. because he feels it's too confusing,...).

**#20 - 12/17/2014 09:33 AM - javawizard (Alex Boyd)**

Martin Dürst wrote:

> It seems that both Matz and I got confused. That's not too surprising because the first example has no parentheses.

An understandable mistake to make.

> I suggest the following:
>
> * Create a new issue that strictly concentrates on the case WITH parentheses.

Will do (or if bug hit gets around to it first, I'll follow his).

> * Beef up the pull request with the necessary changes to tests.

Aw, tests? You're going to make me write tests? :-)

Will do.

> Please note that Matz can still reject the issue if he doesn't like it (e.g. because he feels it's too confusing,...).

Of course. I just want a potential rejection to be with all of the facts understood correctly, and it sounds like that's the case now.

**#21 - 12/17/2014 05:38 PM - bughit (bug hit)**

Martin Dürst wrote:

> I suggest the following:
>
> * Create a new issue that strictly concentrates on the case WITH parentheses.

This bug is about the restriction on the use of multiple assignment in conditionals, it already adequately describes the problem and contains a solution. That it can't be made to work without parens is merely a parenthetical caveat. Why ask for pointless busywork, when it simply needs to be reopened?

**#22 - 12/18/2014 04:52 AM - mame (Yusuke Endoh)**

I think that the restriction is no longer meaningful, but it does not mean that this is a bug. Whatever the reason is, the behavior has been the spec. It is not harmful. It is trivial to work around. So I consider this ticket as a feature request.

I can reopen this ticket, but it has many noises and too long. I also recommend you to open a new ticket to feature request tracker.

Note that a feature request proposer has a duty to persuade matz. You must give a clear, short, and attractive explanation. You should not assume that he could read the whole explanation. He has no time. At worst, he may skip any English sentences and read only the first example. So, you should choose the best example carefully.

--
Yusuke Endoh mame@ruby-lang.org

**#23 - 12/18/2014 05:15 AM - bughit (bug hit)**

Yusuke Endoh wrote:

> I think that the restriction is no longer meaningful, but it does not mean that this is a bug. Whatever the reason is, the behavior has been the spec. It is not harmful. It is trivial to work around. So I consider this ticket as a feature request.
>
> I can reopen this ticket, but it has many noises and too long. I also recommend you to open a new ticket to feature request tracker.
>
> Note that a feature request proposer has a duty to persuade matz. You must give a clear, short, and attractive explanation. You should not assume that he could read the whole explanation. He has no time. At worst, he may skip any English sentences and read only the first example. So, you should choose the best example carefully.

If you can reopen, then please do, I think it's better not to have two tickets for the same issue. I disagree that this is a feature. Recursive composablility of expressions is and should be the rule. It is the exceptions to this rule that require justification. Back when multiple assignment was always truthy, the exception was justified, now it's not, making the current behavior unjustified, illogical, arbitrary, i.e a bug.

**#24 - 12/18/2014 06:12 AM - duerst (Martin Dürst)**

bug hit wrote:

> Yusuke Endoh wrote:
>
> > I think that the restriction is no longer meaningful, but it does not mean that this is a bug. Whatever the reason is, the behavior has been the spec. It is not harmful. It is trivial to work around. So I consider this ticket as a feature request.
> >
> > I can reopen this ticket, but it has many noises and too long. I also recommend you to open a new ticket to feature request tracker.

> If you can reopen, then please do, I think it's better not to have two tickets for the same issue.  I disagree that this is a feature.

It would be helpful if you could defer the triage into bugs and features to people such as Yusuke who have a lot of experience with how Ruby development works. Also, while it's better to not have two tickets for the same issue, it's better to have a new ticket because, as we had to find out, the current ticket mixes up two issues (allowing multiple assignment in conditions and removing parentheses), is confusing, and has accumulated a lot of dust.

#### #25 - 12/18/2014 07:41 AM - bughit (bug hit)

Martin Dürst wrote:

> bug hit wrote:
>
> > Yusuke Endoh wrote:
> >
> > > I think that the restriction is no longer meaningful, but it does not mean that this is a bug.  Whatever the reason is, the behavior has been the spec.  It is not harmful.  It is trivial to work around.  So I consider this ticket as a feature request.
> > >
> > > I can reopen this ticket, but it has many noises and too long.  I also recommend you to open a new ticket to feature request tracker.
> >
> > If you can reopen, then please do, I think it's better not to have two tickets for the same issue.  I disagree that this is a feature.
>
> It would be helpful if you could defer the triage into bugs and features to people such as Yusuke who have a lot of experience with how Ruby development works. Also, while it's better to not have two tickets for the same issue, it's better to have a new ticket because, as we had to find out, the current ticket mixes up two issues (allowing multiple assignment in conditions and removing parentheses), is confusing, and has accumulated a lot of dust.

Good point about the dust.  There are three entries in this bug that are essential, my initial report, Alex Boyd's patch, and Yusuke Endoh's historical explanation.  All of your posts are indeed dust that can and should be removed, try to resist the urge to dust this bug any further.

There are not two issues here, just one, multiple assignment in conditionals is not possible. In the first post I provided examples to show that the problem is present with and without parens (with different errors), that's a good thing, makes the bug report more thorough.

What horror, the confusion of absorbing two related facts instead of one, why, it's enough make your brain seize up.

#### #26 - 12/18/2014 11:02 AM - phluid61 (Matthew Kerwin)

bug hit wrote:

> Good point about the dust.  There are three entries in this bug that are essential, my initial report, Alex Boyd's patch, and Yusuke Endoh's historical explanation.  All of your posts are indeed dust that can and should be removed, try to resist the urge to dust this bug any further.
>
> There are not two issues here, just one, multiple assignment in conditionals is not possible. In the first post I provided examples to show that the problem is present with and without parens (with different errors), that's a good thing, makes the bug report more thorough.
>
> What horror, the confusion of absorbing two related facts instead of one, why, it's enough make your brain seize up.

I've watched this thread with interest, followed by amusement, and finally disappointment. I could write tomes about the Ruby community's values, about what can be gained by being polite to respected members of the Ruby community (not to mention the rest of us), and the value of taking advice or direction from said community members, but apparently that wouldn't contribute in any meaningful way -- it would only be so much dust.

However I must raise a point of order:

> that's a good thing, makes the bug report more thorough.

In this case it was apparently a bad thing, and made the bug report confusing. What is good in some development communities isn't necessarily good in Ruby, and confusing Matz is just about the worst thing you can do here. (Not to mention insulting his intelligence.)

The best way to get what you want is to listen to people, be patient, and be nice. MINSWAN.

</meta-discussion>

#### #27 - 12/18/2014 11:08 AM - recursive-madman (Recursive Madman)

I've created a feature request with attached patch here: #10617 (the patch differs from Alex Boyd's version in that it prints a warning to be consistent with regular assignment in a conditional)

#### #28 - 12/18/2014 11:51 AM - javawizard (Alex Boyd)

#10617 looks good to me. +1 on the warning.

**#29 - 12/18/2014 12:10 PM - recursive-madman (Recursive Madman)**

Actually, thinking about it again I'm not sure anymore if the warning is warranted. The warning for singular assignment is there to warn about the possibility of a typo (a = b vs a == b). Multiple assignment is unambiguous in this respect though (a, b = c makes sense, a, b == c doesn't).

**#30 - 12/21/2014 08:37 PM - bughit (bug hit)**

Recursive Madman wrote:

> Actually, thinking about it again I'm not sure anymore if the warning is warranted. The warning for singular assignment is there to warn about the possibility of a typo (a = b vs a == b). Multiple assignment is unambiguous in this respect though (a, b = c makes sense, a, b == c doesn't).

True, there's no ambiguity, no warning is needed.  I don't think warnings are intended as a subjective style guide, which is what it would be in this case.

**#31 - 01/06/2015 04:06 PM - enebo (Thomas Enebo)**

We just got a report on JRuby that we DO NOT error if masgn is in a conditional: https://github.com/jruby/jruby/issues/2433#issuecomment-68882716

My main two takeaways:

1. I don't think many people ever try to do this since it took so long for this to be reported in our issue tracker.
2. There are probably lots of weird things you can currently put into an if that are not guarded like this.