Ruby master - Bug #14130

Keyword arguments are ripped from the middle of hash if argument have default value

11/25/2017 02:00 PM - zverok (Victor Shepelev)

Status:	Closed		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:			
ruby -v:		**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

Here is the code:

```
def test1(source = {}, **opts)
  puts "SOURCE: #{source}, OPTS: #{opts}"
end

def test2(source, **opts)
  puts "SOURCE: #{source}, OPTS: #{opts}"
end

puts "No source"
test1(length: 2000)
# 1. SOURCE: {}, OPTS: {:length=>2000} -- OK, it is reasonable.
test2(length: 2000)
# 2. SOURCE: {:length=>2000}, OPTS: {} -- Exactly as expected.

puts
puts "Source is mixed hash"
test1('River name' => 'Mississippi', length: 2000, 'Country' => 'USA')
# 3. SOURCE: {"River name"=>"Mississippi", "Country"=>"USA"}, OPTS: {:length=>2000} -- It is alrea
dy a bit weird
test2('River name' => 'Mississippi', length: 2000, 'Country' => 'USA')
# 4. SOURCE: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}, OPTS: {} -- The most
weird thing!
```

My concern is cases (3) and (4). Ripping keyword argument from what from any logic looks like a middle of a hash (3) is already pretty strange. But the fact that this behavior depends on whether first argument has or has not default value (3 vs 4) clearly looks like some bug?

Checked on several recent versions, including ruby 2.5.0dev (2017-09-11 trunk 59836) [x86_64-linux] (the last that is available on my RVM), behavior is consistent between them.

Related issues:

Related to Ruby master - Feature #14183: "Real" keyword argument	**Closed**

Associated revisions

Revision 3463e831 - 08/31/2019 02:25 AM - jeremyevans (Jeremy Evans)

Warn for keyword to last hash parameter when method has no optional/rest parameters

Previously, there was no warning in this case, even though we will
be changing the behavior in Ruby 3.

Fixes [Bug #14130]

History

#1 - 12/14/2017 07:25 AM - hsbt (Hiroshi SHIBATA)

- Related to Feature #14183: "Real" keyword argument added

#2 - 09/05/2018 03:44 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to nobu (Nobuyoshi Nakada)

This is a bug.

```
test1('River name' => 'Mississippi', length: 2000, 'Country' => 'USA')
# 3. SOURCE: {"River name"=>"Mississippi", "Country"=>"USA"}, OPTS: {:length=>2000} -- It is already a bit wei
rd
```

My understanding is that under no circumstance should a mixed hash be split in two like this.

Ruby should recognize that some keys are no symbols and thus this hash can't be a keyword parameter hash and must be a positional argument. The correct result should thus be:

```
SOURCE: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}, OPTS: {}
```

Note that this should still be true even if the :length key was the last one.

To obtain the current result, one would have to call test1({'River name' => 'Mississippi', 'Country' => 'USA'}, length: 2000)

#3 - 08/30/2019 10:06 PM - jeremyevans0 (Jeremy Evans)

With the changes in #14183, keyword splats support non-symbol keys, so with the master branch you now get:

```
No source
SOURCE: {}, OPTS: {:length=>2000}
SOURCE: {:length=>2000}, OPTS: {}

Source is mixed hash
SOURCE: {}, OPTS: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}
SOURCE: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}, OPTS: {}
```

In Ruby 3, you will get:

```
No source
SOURCE: {}, OPTS: {:length=>2000}
# ArgumentError, because only keywords provided and no positional arguments, and at least one positional argum
ent is required

Source is mixed hash
SOURCE: {}, OPTS: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}
# ArgumentError, because only keywords provided and no positional arguments, and at least one positional argum
ent is required
```

This shows there is still work do to in 2.7 to warn cases where the behavior will change in Ruby 3. In both cases, the calls to test2 should emit a warning, because the method will raise an error in Ruby 3.

#4 - 08/30/2019 11:25 PM - mame (Yusuke Endoh)

jeremyevans0 (Jeremy Evans) Completely agreed.

Matz says that it is a bug to split keywords depending upon whether the key is Symbol or non-Symbol. (The behavior was introduced without matz's confirmation.) So I think it is okay that the behavior of Ruby 2.7 slightly changes.

#5 - 08/31/2019 02:46 AM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

With https://github.com/ruby/ruby/commit/3463e83192215c36bdcebad8be907eaa09593a41, you now get warnings for calls where the behavior will change in Ruby 3:

```
No source
SOURCE: {}, OPTS: {:length=>2000}
file.rb:12: warning: The keyword argument for `test2' (defined at file.rb:5) is passed as the last hash parame
ter
SOURCE: {:length=>2000}, OPTS: {}

Source is mixed hash
SOURCE: {}, OPTS: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}
file.rb:19: warning: The keyword argument for `test2' (defined at file.rb:5) is passed as the last hash parame
ter
SOURCE: {"River name"=>"Mississippi", :length=>2000, "Country"=>"USA"}, OPTS: {}
```