Ruby trunk - Feature #5454

keyword arguments

10/18/2011 01:21 AM - mame (Yusuke Endoh)

Status:	Rejected	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	
Target version:		

Description

遠藤です。

キーワード引数を実装してみました。

```
$ cat t.rb
def foo(str: "foo", num: 424242)
p [str, num]
end
foo(str: "bar")

$ ./ruby t.rb
["bar", 424242]
```

細かいことはまだ気にしていませんが、キーワード引数が 2.0 に入ることはかなり確定的なので、とりあえず trunk に入れて考えるというのはどうでしょうか。

一応私の環境では make check は通ります (キーワード引数のテストは書いてないですが) 。パーサに conflict はないです。
既存のベンチマークを見ると、一部気になる項目もありますが (vm1_rescue とか) 、全体的には特に速度劣化しないようです。

以下は未検討・未実装の細かいことです。

- 残りのキーワード引数を受け取る方法が必要?

  ```
  def foo(str: "foo", num: 424242, **h)
  p h
  end
  foo(str: "bar", check: true)  #=> { :check => true }
  ```

- nil が渡されたらやはり nil になるべき?

  ```
  def foo(str: "foo")
  p str
  end
  foo(str: nil) #=> nil ? (今は手抜きで "foo")
  ```

- rest なしで知らないキーワードが来たら例外・警告を発して欲しい?

  ```
  def foo(str: "foo", num: 424242)
  end
  foo(check: true) #=> ArgumentError?
  ```

- Hash のように振る舞うオブジェクトもキーワード引数として解釈すべき? (現在は速度重視で T_HASH だけ見ている)

- 引数の順番はこれでよいか?

  ```
  def foo(
  a, b, c,     # mandatory arguments
  opt = 1,     # optional arguments
  *rest,       # rest argument
  ```

```
    x, y, z,     # post mandatory arguments
    k1: 1, k2:2, # keyword arguments
    **kh,        # keyword rest argument
    &blk         # block argument
    )
```

- □□□□□

benchmark results:
name ruby 1.9.4dev (2011-10-16 trunk 33471) [i686-linux] ruby 1.9.4dev (2011-10-16 trunk 33471) [i686-linux] average difference
app_answer 0.259 0.131 -0.127
app_erb 2.699 2.681 -0.018
app_factorial 5.236 5.320 0.084
app_fib 1.576 1.731 0.154
app_mandelbrot 4.324 3.911 -0.413
app_pentomino 40.882 40.768 -0.114
app_raise 0.966 0.982 0.017
app_strconcat 3.143 3.102 -0.041
app_tak 2.192 2.140 -0.052
app_tarai 1.682 1.748 0.066
app_uri 1.870 1.767 -0.103
io_file_create 2.589 2.570 -0.019
io_file_read 6.169 5.838 -0.330
io_file_write 1.966 1.976 0.010
io_select 3.066 3.111 0.044
io_select2 3.647 3.623 -0.025
io_select3 0.051 0.051 0.000
loop_for 3.302 3.219 -0.082
loop_generator 1.111 1.102 -0.009
loop_times 2.959 2.960 0.001
loop_whileloop 1.437 1.422 -0.016
loop_whileloop2 0.298 0.297 -0.001
so_ackermann 1.862 1.816 -0.046
so_array 3.317 3.473 0.155
so_binary_trees 0.891 0.900 0.009
so_concatenate 9.538 8.696 -0.842
so_count_words 0.412 0.407 -0.005
so_exception 2.185 2.177 -0.008
so_fannkuch 3.338 3.334 -0.004
so_fasta 5.182 5.061 -0.121
so_k_nucleotide 3.425 3.324 -0.101
so_lists 2.239 2.312 0.073
so_mandelbrot 11.479 10.660 -0.820
so_matrix 2.206 2.177 -0.030
so_meteor_contest 11.479 11.634 0.155
so_nbody 8.006 7.925 -0.081
so_nested_loop 2.768 2.614 -0.154
so_nsieve 6.163 6.184 0.021
so_nsieve_bits 6.618 6.850 0.231
so_object 1.807 1.899 0.091
so_partial_sums 11.057 10.537 -0.520
so_pidigits 2.944 3.011 0.066
so_random 1.962 1.992 0.031
so_reverse_complement 3.136 3.036 -0.099
so_sieve 2.224 2.203 -0.021
so_spectralnorm 7.981 7.591 -0.390
vm1_block* 3.790 5.112 1.306
vm1_const* 0.756 0.757 -0.015
vm1_ensure* 0.487 0.601 0.099
vm1_ivar* 1.634 1.863 0.213
vm1_ivar_set* 1.625 1.661 0.021
vm1_length* 1.312 1.366 0.038
vm1_neq* 1.152 0.979 -0.189
vm1_not* 0.589 0.589 -0.015
vm1_rescue* 0.133 0.750 0.601

```
vm1_simplereturn*       2.467   3.601   1.119
vm1_swap*       1.199   1.716   0.502
vm2_array*      1.644   1.643   -0.001
vm2_case*       0.324   0.348   0.023
vm2_defined_method*     8.829   9.003   0.173
vm2_eval*       39.557  37.726  -1.832
vm2_method*     5.255   4.113   -1.142
vm2_mutex*      3.043   3.175   0.131
vm2_poly_method*        6.044   5.340   -0.705
vm2_poly_method_ov*     0.493   0.675   0.181
vm2_proc*       1.297   1.391   0.093
vm2_regexp*     2.419   2.557   0.137
vm2_send*       0.662   0.903   0.241
vm2_super*      1.107   1.016   -0.093
vm2_unif1*      0.688   0.816   0.127
vm2_zsuper*     1.210   1.068   -0.143
vm3_clearmethodcache    0.785   0.770   -0.014
vm3_gc  2.487   2.453   -0.033
vm_thread_alive_check1  0.389   0.422   0.033
vm_thread_create_join   6.064   6.181   0.118
vm_thread_mutex1        2.349   2.546   0.197
vm_thread_mutex2        10.123  7.755   -2.368
vm_thread_mutex3        4.428   5.390   0.962
vm_thread_pass  0.692   0.754   0.063
vm_thread_pass_flood    0.167   0.170   0.003
```

vm_thread_pipe 2.437   2.540   0.103

average total difference is -3.4518661499023438

--
Yusuke Endoh mame@tsg.ne.jp

Related issues:

Related to Ruby trunk - Feature #227: rb_scan_args() for keyword arguments	**Closed**

History

#1 - 10/18/2011 07:55 AM - matz (Yukihiro Matsumoto)

|細かいことはまだ気にしていませんが、キーワード引数が 2.0 に入る
|ことはかなり確定的なので、とりあえず trunk に入れて考えるという
|のはどうでしょうか。

賛成です。trunkに入れちゃっていいのかな。そういう機能の入っ
ていない1.9ブランチが欲しい人は急いで切ったほうがいいと思い
ますよ。

|以下は未実装・懸案です。
|
| - 残りのキーワード引数を受け取る方法が必要か
|
| def foo(str: "foo", num: 424242, **h)
| p h
| end
| foo(str: "bar", check: true) #=> { :check => true }

メソッドの移譲のために必要だと思います。

| - nil が渡されたらやはり nil になるべき?
|
| def foo(str: "foo")
| p str
| end
| foo(str: nil) #=> nil ? (今は手抜きで "foo")

nilになるべきでしょう。

| - rest なしで知らないキーワードが来たら例外・警告を発して
| 欲しい?
|

```
|   def foo(str: "foo", num: 424242)
|   end
|   foo(check: true) #=> ArgumentError?
```

ここは議論の余地があるところですね。現時点では、**で受け取ら
ない限り例外にするべきだと考えていますが、実際にいろいろな局
面で使ってみると意見が変わるかもしれません。

```
| - Hash のように振る舞うオブジェクトもキーワード引数として
|   解釈すべき? (現在は速度重視で T_HASH だけ見ている)
```

とりあえずto_hashを使ってください。新設のto_kwのようなメソッ
ドを使うべきなのかもしれませんが。

```
| - 引数の順番はよいか
|
|   def foo(
|     a, b, c,     # mandatory arguments
|     opt = 1,     # optional arguments
|     *rest,       # rest argument
|     x, y, z,     # post mandatory arguments
|     k1: 1, k2:2, # keyword arguments
|     **kh,        # keyword rest argument
|     &blk         # block argument
|   )
```

これで良いと思います。

```
                                        まつもと ゆきひろ  /:|)
```

#2 - 10/18/2011 08:23 AM - Anonymous

This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.

--

Evan Phoenix // evan@phx.io

#3 - 10/18/2011 11:59 AM - Anonymous

Hi,

2011年10月18日8:13 Evan Phoenix evan@phx.io:

> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.
>
> I and NaHi translated the mails.

http://ruby-dev.info/posts/44602

--
yhara (Yutaka HARA)

#4 - 10/18/2011 01:23 PM - matz (Yukihiro Matsumoto)

Hi,

I should have posted in English at first.
Thank you guys for translation.

```
                                matz.
```

In message "Re: [ruby-core:40191] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
on Tue, 18 Oct 2011 11:56:29 +0900, Yutaka Hara yutaka.hara@gmail.com writes:
|
|Hi,
|
|2011年10月18日8:13 Evan Phoenix evan@phx.io:
|> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.
|>
|I and NaHi translated the mails.
|
|http://ruby-dev.info/posts/44602
|

|--
|yhara (Yutaka HARA)

#5 - 10/18/2011 02:26 PM - Anonymous

Do I need to learn Ruby first to learn Rails?

Loving Ruby and want to learn it.

Thanks

2011/10/18 Yukihiro Matsumoto matz@ruby-lang.org

> Hi,
>
> I should have posted in English at first.
> Thank you guys for translation.
>
> ```
> matz.
> ```
>
> In message "Re: [ruby-core:40191] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
> on Tue, 18 Oct 2011 11:56:29 +0900, Yutaka Hara yutaka.hara@gmail.com writes:
> |
> |Hi,
> |
> |2011□10□18□8:13 Evan Phoenix evan@phx.io:
> |> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.
> |>
> |I and NaHi translated the mails.
> |
> |http://ruby-dev.info/posts/44602
> |
> |--
> |yhara (Yutaka HARA)

#6 - 10/18/2011 02:31 PM - Anonymous

Thanks for the translation!

I've got a question about the proposal that will help me understand how it would be implemented.

Given a method:

```
def foo(name, age)
p [name, age]
end
```

And then code that calls this method:

```
arg = { :name => "Evan", :age => 32 }
foo(arg)
```

Would this print out ["Evan", 32]?

If so, this seems like a very big change that a lot of code written against 1.9 will not work with.

Also, the example code in the proposal uses a form to call foo like this:

```
foo(name: "Evan", age: 32)
```

This reuses the 1.8/1.9 behavior of the implicit hash parameter. Does this mean that the implementation creates a hash on the caller side, which is then pulled apart by the called method? If so, these seems just like my first example, and would end up breaking a lot of code.

Because 1.9 introduces the new hash syntax, which programmers have begun using with the implicit hash parameter, I don't think that the syntax can be used again for keyword arguments without introducing an incompatible change.

So, I'd like to suggest an alternative to this which uses a different (and thusly unambiguous) syntax:

```
foo(name="Evan", age=32)
```

I know that this is valid ruby code in 1.9, but it is extremely rare, much much more rare than the implicit hash parameter. Thusly it is safer and will not break 1.9 code.

This form also allows for an implementation of keyword parameters that don't require creating a Hash object, which makes the code faster and more useful.

If a truly new syntax must be introduced to be 100% backward compatible, I'd suggest:

foo(name := "Evan", age := 32)

:= is already known in computing as a bind operator, and thus it is suited well to this task.

Thank you for considering my proposal.

- Evan

--

Evan Phoenix // evan@phx.io

On Monday, October 17, 2011 at 9:08 PM, Yukihiro Matsumoto wrote:

> Hi,
>
> I should have posted in English at first.
> Thank you guys for translation.
>
> matz.
>
> In message "Re: [ruby-core:40191] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
> on Tue, 18 Oct 2011 11:56:29 +0900, Yutaka Hara writes:
> |
> |Hi,
> |
> |2011□10□18□8:13 Evan Phoenix :
> |> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.
> |>
> |I and NaHi translated the mails.
> |
> |http://ruby-dev.info/posts/44602
> |
> |--
> |yhara (Yutaka HARA)

#7 - 10/18/2011 02:31 PM - rkh (Konstantin Haase)

Please ask such question at the Rails mailing list, the ruby-talk mailing list, or Stack Overflow. The core list is about developing Ruby, not developing *in* Ruby.

Konstantin

On Oct 17, 2011, at 22:14 , Praveer Dikshit wrote:

> Do I need to learn Ruby first to learn Rails?
>
> Loving Ruby and want to learn it.
>
> Thanks
>
> 2011/10/18 Yukihiro Matsumoto matz@ruby-lang.org
> Hi,
>
> I should have posted in English at first.
> Thank you guys for translation.
>
> ```
>                                                 matz.
> ```
>
> In message "Re: [ruby-core:40191] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
> on Tue, 18 Oct 2011 11:56:29 +0900, Yutaka Hara yutaka.hara@gmail.com writes:
> |
> |Hi,
> |
> |2011□10□18□8:13 Evan Phoenix evan@phx.io:

|> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.
|>
|I and NaHi translated the mails.
|
|http://ruby-dev.info/posts/44602
|
|--
|yhara (Yutaka HARA)

#8 - 10/18/2011 02:53 PM - rkh (Konstantin Haase)

From the current patch it seems to me that this would raise an ArgumentError, as it does now. Neither name nor age are "keyword arguments". There is no way to define keyword arguments without a default.

Also, these are really named arguments in a Python sense, not keyword arguments in a Smalltalk sense.

matz (Yukihiro Matsumoto):

```
def foo(bar: 42) bar end
foo bar: nil
```

The above should return nil IMHO, since this is also what I'd expect if it would be an options hash.

Konstantin

On Oct 17, 2011, at 22:27 , Evan Phoenix wrote:

> Thanks for the translation!
>
> I've got a question about the proposal that will help me understand how it would be implemented.
>
> Given a method:
>
> def foo(name, age)
> p [name, age]
> end
>
> And then code that calls this method:
>
> arg = { :name => "Evan", :age => 32 }
> foo(arg)
>
> Would this print out ["Evan", 32]?
>
> If so, this seems like a very big change that a lot of code written against 1.9 will not work with.
>
> Also, the example code in the proposal uses a form to call foo like this:
>
> foo(name: "Evan", age: 32)
>
> This reuses the 1.8/1.9 behavior of the implicit hash parameter. Does this mean that the implementation creates a hash on the caller side, which is then pulled apart by the called method? If so, these seems just like my first example, and would end up breaking a lot of code.
>
> Because 1.9 introduces the new hash syntax, which programmers have begun using with the implicit hash parameter, I don't think that the syntax can be used again for keyword arguments without introducing an incompatible change.
>
> So, I'd like to suggest an alternative to this which uses a different (and thusly unambiguous) syntax:
>
> foo(name="Evan", age=32)
>
> I know that this is valid ruby code in 1.9, but it is extremely rare, much much more rare than the implicit hash parameter. Thusly it is safer and will not break 1.9 code.
>
> This form also allows for an implementation of keyword parameters that don't require creating a Hash object, which makes the code faster and more useful.
>
> If a truly new syntax must be introduced to be 100% backward compatible, I'd suggest:
>
> foo(name := "Evan", age := 32)
>
> := is already known in computing as a bind operator, and thus it is suited well to this task.
>
> Thank you for considering my proposal.

- Evan

--

Evan Phoenix // evan@phx.io

On Monday, October 17, 2011 at 9:08 PM, Yukihiro Matsumoto wrote:

> Hi,
>
> I should have posted in English at first.
> Thank you guys for translation.
>
> matz.
>
> In message "Re: [ruby-core:40191] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
> on Tue, 18 Oct 2011 11:56:29 +0900, Yutaka Hara writes:
> |
> |Hi,
> |
> |2011□10□18□8:13 Evan Phoenix :
> |> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed.
> |>
> |I and NaHi translated the mails.
> |
> |http://ruby-dev.info/posts/44602
> |
> |--
> |yhara (Yutaka HARA)

#9 - 10/18/2011 02:53 PM - Anonymous

See below.

--

Evan Phoenix // evan@phx.io

On Monday, October 17, 2011 at 10:33 PM, Haase, Konstantin wrote:

> From the current patch it seems to me that this would raise an ArgumentError, as it does now. Neither name nor age are "keyword arguments". There is no way to define keyword arguments without a default.
> Ah! Ok, so keyword arguments are completely separate than normal arguments. This feature would basically be "Hash restructuring", ie letting the VM know what keys in the Hash your interested in and having it pull them out for you.

That seems quite interesting, but less useful that being to invoke a method and bind a value to a normal argument via it's name.

- Evan
  > > Also, these are really named arguments in a Python sense, not keyword arguments in a Smalltalk sense. > > matz (Yukihiro Matsumoto): > > def foo(bar: 42) bar end > foo bar: nil > > The above should return nil IMHO, since this is also what I'd expect if it would be an options hash. > > Konstantin > > On Oct 17, 2011, at 22:27 , Evan Phoenix wrote: > > > Thanks for the translation! > > > > I've got a question about the proposal that will help me understand how it would be implemented. > > > > Given a method: > > > > def foo(name, age) > > p [name, age] > > end > > > > And then code that calls this method: > > > > arg = { :name => "Evan", :age => 32 } > > foo(arg) > > > > Would this print out ["Evan", 32]? > > > > If so, this seems like a very big change that a lot of code written against 1.9 will not work with. > > > > Also, the example code in the proposal uses a form to call foo like this: > > > > foo(name: "Evan", age: 32) > > > > This reuses the 1.8/1.9 behavior of the implicit hash parameter. Does this mean that the implementation creates a hash on the caller side, which is then pulled apart by the called method? If so, these seems just like my first example, and would end up breaking a lot of code. > > > > > > Because 1.9 introduces the new hash syntax, which programmers have begun using with the implicit hash parameter, I don't think that the syntax can be used again for keyword arguments without introducing an incompatible change. > > > > So, I'd like to suggest an alternative to this which uses a different (and thusly unambiguous) syntax: > >

> > foo(name="Evan", age=32) > >
> > I know that this is valid ruby code in 1.9, but it is extremely rare, much much more rare than the implicit hash parameter. Thusly it is safer and will not break 1.9 code. > >
> > This form also allows for an implementation of keyword parameters that don't require creating a Hash object, which makes the code faster and more useful. > >
> > If a truly new syntax must be introduced to be 100% backward compatible, I'd suggest: > >
> > foo(name := "Evan", age := 32) > >
> > := is already known in computing as a bind operator, and thus it is suited well to this task. > >
> > Thank you for considering my proposal. > >
> > - Evan > >
> > --
> > Evan Phoenix // evan@phx.io (mailto:evan@phx.io) > >
> >
> > On Monday, October 17, 2011 at 9:08 PM, Yukihiro Matsumoto wrote: > >
> > > Hi, > > >
> > > I should have posted in English at first. > > > Thank you guys for translation. > > >
> > > matz. > > >
> > > In message "Re: [ruby-core:40191] Re: [Ruby 2.0 - Feature #5454] keyword arguments" > > > on Tue, 18 Oct 2011 11:56:29 +0900, Yutaka Hara writes: > > > | > > > |Hi, > > > | > > > |2011□10□18□8:13 Evan Phoenix : > > > |> This looks very interesting! Would someone be willing to translate to english? I've only got a vague idea of what is being discussed. > > > |> > > > |I and NaHi translated the mails. > > > | > > > | http://ruby-dev.info/posts/44602 > > > | > > > |-- > > > |yhara (Yutaka HARA)

#10 - 10/19/2011 12:23 AM - jballanc (Joshua Ballanco)

On Tue, Oct 18, 2011 at 1:33 AM, Haase, Konstantin <
Konstantin.Haase@student.hpi.uni-potsdam.de> wrote:

> Also, these are really named arguments in a Python sense, not keyword
> arguments in a Smalltalk sense.

I just wanted to point out that MacRuby (in order to support compatibility with Obj-C which uses Smalltalk style keyword arguments) already uses the hash syntax for defining/calling methods, but the semantics are different from what is being proposed here, if I understand correctly. To summarize:

```
# Ruby 2.0 proposal:
def foo(a: 1, b: 2, c: 3)
    puts [a, b, c]
end
foo(a: 'one', b: 'two', c: 'three') #=> ["one", "two", "three"]
```

```
# Currently in MacRuby:
def foo(a: first, b: second, c: third)
    puts [first, second, third]
end
foo(a: 'one', b: 'two', c: 'three') #=> ["one", "two", "three"]
```

With the current proposal for keyword arguments, I think MacRuby would not be able to implement this feature.

#11 - 10/19/2011 12:53 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:40207] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
on Wed, 19 Oct 2011 00:14:13 +0900, Joshua Ballanco jballanc@gmail.com writes:

|With the current proposal for keyword arguments, I think MacRuby would not
|be able to implement this feature.

I know. But I have warned MacRuby guys for years.

```
                          matz.
```

#12 - 10/19/2011 12:53 AM - mame (Yusuke Endoh)

Hello,

2011/10/19 Joshua Ballanco jballanc@gmail.com:

> Â Â # Ruby 2.0 proposal:
> Â Â def foo(a: 1, b: 2, c: 3)
> Â Â Â Â puts [a, b, c]
> Â Â end

Â Â foo(a: 'one', b: 'two', c: 'three') #

#13 - 10/19/2011 12:53 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:40209] Re: [Ruby 2.0 - Feature #5454] keyword arguments"
on Wed, 19 Oct 2011 00:31:44 +0900, Yusuke Endoh mame@tsg.ne.jp writes:

|Interesting. Matz, which do you prefer?

I prefer the former (keyword hash) way. MacRuby keyword arguments are
solely for bridging with Objective-C, so that they can translate

obj.foo(1, bar: 2)

into

[obj foo: 1 bar: 2]

in Objective-C.

```
matz.
```

#14 - 10/19/2011 12:53 AM - mame (Yusuke Endoh)

Hello,

2011/10/19 Yukihiro Matsumoto matz@ruby-lang.org:

> |With the current proposal for keyword arguments, I think MacRuby would not
> |be able to implement this feature.
>
> I know. Â But I have warned MacRuby guys for years.

Okay, I see you prefer the current proposal.

Still, I guess MacRuby will serve as a useful reference
when we discuss this proposal, especially, the corner cases.

--
Yusuke Endoh mame@tsg.ne.jp

#15 - 10/19/2011 12:53 AM - jballanc (Joshua Ballanco)

On Tue, Oct 18, 2011 at 11:31 AM, Yukihiro Matsumoto matz@ruby-lang.orgwrote:

> Hi,
>
> In message "Re: [ruby-core:40207] Re: [Ruby 2.0 - Feature #5454] keyword
> arguments"
> on Wed, 19 Oct 2011 00:14:13 +0900, Joshua Ballanco <
> jballanc@gmail.com> writes:
>
> |With the current proposal for keyword arguments, I think MacRuby would not
> |be able to implement this feature.
>
> I know. But I have warned MacRuby guys for years.

Heh, indeed. I just spoke briefly with lrz, and he's optimistic that we can
make it work. One question I have, though, would be if you intend to allow
variables as default values or only literals. For example, should this work:

```
x = 30
def foo(a: 1, b: 2, c: x)
    puts [a, b, c]
end
foo(a: 10, b: 20) #=>  [10, 20, 30]
```

?

#16 - 10/19/2011 01:23 AM - mame (Yusuke Endoh)

Hello,

2011/10/19 Joshua Ballanco :

> For example, should this work:
> x = 30
> def foo(a: 1, b: 2, c: x)
> puts [a, b, c]
> end
> foo(a: 10, b: 20) #=> ¨Â±°²°³°> ?

Though the example will not work because x is in the different
variable scope, you can write any expression as a default value,
like:

```
def foo(str: File.read("/etc/passwd"))
p str
end
foo(str: "bar") #=> "bar"
foo             #=> the contents of /etc/passwd
```

... currently it does not work correctly because of bug of my
implementation, though :-)
I'll update the patch tomorrow.

--
Yusuke Endoh

#17 - 10/23/2011 02:52 PM - mame (Yusuke Endoh)

- File keyword-argument-patch-20111023.zip added

- Assignee changed from matz (Yukihiro Matsumoto) to ko1 (Koichi Sasada)

Hello,

I've updated the patch (keyword-argument-patch-20111023.zip).
It modifies the core much.
So I'd like to commit it after koichi's review.

The rough strategy of the implementation is to interpret:

```
def foo(str: "foo", num: 424242, **h)
...
end
```

as:

```
def foo(*r)
h = r.last.respond_to?(:to_hash) ? r.last.to_hash.dup : {}
str = h.key?(:str) ? h.delete(:str) : "foo"
num = h.key?(:num) ? h.delete(:num) : 424242
...
end
```

> | - Do we need a way to receive all the keyword arguments?
> |
> | def foo(str: "foo", num: 424242, **h)
> | p h
> | end
> | foo(str: "bar", check: true) #=> { :check => true }
>
> It would be necessary for delegating arguments to another method.

Done.

```
$ ./ruby -e '
def foo(str: "foo", num: 4242, **h)
p h
end
foo(str: "bar", check: true)
```

'
{:check=>true}

| - When nil is passed explicitly, should the value be nil?
|
| def foo(str: "foo")
| p str
| end
| foo(str: nil) #=> nil ? (not implemented yet, "foo" is assigned now)

It should be nil.

Done.

```
$ ./ruby -e '
def foo(str: "foo", num: 4242)
p str
end
foo(str: nil)
'
nil
```

| - When unknown keyword is specified and there is no **rest,
| should we raise exception or warning?
|
| def foo(str: "foo", num: 424242)
| end
| foo(check: true) #=> ArgumentError?

There is a room for discussion. Currently I think it should raise an exception unless receiving with **rest, but I may change my opinion after using it in several situations in actual scripts.

Done.

```
$ ./ruby -e '
def foo(str: "foo", num: 4242)
end
foo(str: "bar", check: true)

'
-e:5:in `': unknown keyword (TypeError)
```

| - Should we accept an object behaves like Hash as keyword arguments?
| (Current implementation only checks T_HASH for speed)

Please use to_hash for now. Maybe we should define new method, something like to_kw.

Done.

```
$ ./ruby -e '
def foo(str: "foo", num: 4242)
p str
end
h = Object.new
def h.to_hash; { str: "bar" }; end
foo(h)
'
"bar"
```

| - Is order of arguments ok?
|
| def foo(
| a, b, c, # mandatory arguments
| opt = 1, # optional arguments
| *rest, # rest argument
| x, y, z, # post mandatory arguments
| k1: 1, k2:2, # keyword arguments
| **kh, # keyword rest argument
| &blk # block argument
|)

I think this order is OK.

Thanks!

--
Yusuke Endoh mame@tsg.ne.jp

#18 - 10/23/2011 05:21 PM - naruse (Yui NARUSE)

- Project changed from CommonRuby to Ruby trunk

- Target version deleted (2.6)

#19 - 10/23/2011 07:25 PM - mame (Yusuke Endoh)

- Status changed from Assigned to Rejected

すみません、メールがあちこちに言って議論が混乱しそうですね。
一度 close して立て直します。

Sorry for scattering the mails and confusing the discussion.
I close this once and I'll create a new ticket.

--
Yusuke Endoh mame@tsg.ne.jp

Files

0001-remove-NEW_LAMBDA-hack.patch	1.51 KB	10/18/2011	mame (Yusuke Endoh)
0002-use-struct-rb_args_info-instead-of-NODEs.patch	9.33 KB	10/18/2011	mame (Yusuke Endoh)
0003-implement-a-parser-part-for-basic-keyword-arguments.patch	23.2 KB	10/18/2011	mame (Yusuke Endoh)
0004-rough-implementation-of-keyword-arguments.patch	5.43 KB	10/18/2011	mame (Yusuke Endoh)
0005-temporal-fix-for-ripper-test.patch	717 Bytes	10/18/2011	mame (Yusuke Endoh)
keyword-argument-patch-20111023.zip	11.6 KB	10/23/2011	mame (Yusuke Endoh)