

Ruby trunk - Feature #3447

argument delegation

06/16/2010 06:57 PM - nobu (Nobuyoshi Nakada)

Status:	Assigned
Priority:	Normal
Assignee:	matz (Yukihiro Matsumoto)
Target version:	

Description

日本語

<http://www.rubyist.net/~matz/20100615.html#p01> 日本語

foo(...)foo(..)foo(..)

```
diff --git i/compile.c w/compile.c
index 4621cd9..d769c56 100644
--- i/compile.c
+++ w/compile.c
@@ -2729,7 +2729,6 @@ defined_expr(rb_iseq_t *iseq, LINK_ANCHOR *ret,
     return 1;

     case NODE_SUPER:
-     case NODE_ZSUPER:
     ADD_INSN(ret, nd_line(node), putnil);
     ADD_INSN3(ret, nd_line(node), defined, INT2FIX(DEFINED_ZSUPER), 0,
               needstr);
@@ -2919,6 +2918,67 @@ setup_args(rb_iseq_t *iseq, LINK_ANCHOR *args, NODE *argn, unsigned long *f
lag)
     POP_ELEMENT(args);
     break;
 }
+ case NODE_DELEGATE: {
+     int i;
+     rb_iseq_t *liseq = iseq->local_iseq;
+
+     if (argn->nd_state) *flag |= VM_CALL_SUPER_BIT;
+     argc = INT2FIX(liseq->argc);
+
+     /* normal arguments */
+     for (i = 0; i < liseq->argc; i++) {
+         int idx = liseq->local_size - i;
+         ADD_INSN1(args, nd_line(argn), getlocal, INT2FIX(idx));
+     }
+
+     if (!liseq->arg_simple) {
+         if (liseq->arg_opts) {
+             /* optional arguments */
+             int j;
+             for (j = 0; j < liseq->arg_opts - 1; j++) {
+                 int idx = liseq->local_size - (i + j);
+                 ADD_INSN1(args, nd_line(argn), getlocal, INT2FIX(idx));
+             }
+             i += j;
+             argc = INT2FIX(i);
+         }
+     }
+
+     if (liseq->arg_rest != -1) {
+         /* rest argument */
+         int idx = liseq->local_size - liseq->arg_rest;
+         ADD_INSN1(args, nd_line(argn), getlocal, INT2FIX(idx));
+         argc = INT2FIX(liseq->arg_rest + 1);
+         *flag |= VM_CALL_ARGS_SPLAT_BIT;
+     }
+ }
```

```

+
+     if (liseq->arg_post_len) {
+         /* post arguments */
+         int post_len = liseq->arg_post_len;
+         int post_start = liseq->arg_post_start;
+
+         if (liseq->arg_rest != -1) {
+             int j;
+             for (j=0; j<post_len; j++) {
+                 int idx = liseq->local_size - (post_start + j);
+                 ADD_INSN1(args, nd_line(argn), getlocal, INT2FIX(idx));
+             }
+             ADD_INSN1(args, nd_line(argn), newarray, INT2FIX(j));
+             ADD_INSN (args, nd_line(argn), concatarray);
+             /* argc is setted at above */
+         }
+         else {
+             int j;
+             for (j=0; j<post_len; j++) {
+                 int idx = liseq->local_size - (post_start + j);
+                 ADD_INSN1(args, nd_line(argn), getlocal, INT2FIX(idx));
+             }
+             argc = INT2FIX(post_len + post_start);
+         }
+     }
+ }
+
+     break;
+ }
+ default: {
+     rb_bug("setup_arg: unknown node: %s\n", ruby_node_name(nd_type(argn)));
+ }
@@ -4115,8 +4175,7 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)
+ }
+ break;
+ }
-     case NODE_SUPER:
-     case NODE_ZSUPER:{
+     case NODE_SUPER: {
DECL_ANCHOR(args);
VALUE argc;
unsigned long flag = 0;
@@ -4124,72 +4183,12 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)
d)
INIT_ANCHOR(args);
iseq->compile_data->current_block = Qfalse;
- if (nd_type(node) == NODE_SUPER) {
-     argc = setup_args(iseq, args, node->nd_args, &flag);
- }
- else {
-     /* NODE_ZSUPER */
-     int i;
-     rb_iseq_t *liseq = iseq->local_iseq;
-
-     argc = INT2FIX(liseq->argc);
-
-     /* normal arguments */
-     for (i = 0; i < liseq->argc; i++) {
-         int idx = liseq->local_size - i;
-         ADD_INSN1(args, nd_line(node), getlocal, INT2FIX(idx));
-     }
-
-     if (!liseq->arg_simple) {
-     if (liseq->arg_opts) {
-         /* optional arguments */
-         int j;

```

```

-         for (j = 0; j < liseq->arg_opts - 1; j++) {
-             int idx = liseq->local_size - (i + j);
-             ADD_INSN1(args, nd_line(node), getlocal, INT2FIX(idx));
-         }
-         i += j;
-         argc = INT2FIX(i);
-     }
-
-     if (liseq->arg_rest != -1) {
-         /* rest argument */
-         int idx = liseq->local_size - liseq->arg_rest;
-         ADD_INSN1(args, nd_line(node), getlocal, INT2FIX(idx));
-         argc = INT2FIX(liseq->arg_rest + 1);
-         flag |= VM_CALL_ARGS_SPLAT_BIT;
-     }
-
-     if (liseq->arg_post_len) {
-         /* post arguments */
-         int post_len = liseq->arg_post_len;
-         int post_start = liseq->arg_post_start;
-
-         if (liseq->arg_rest != -1) {
-             int j;
-             for (j=0; j<post_len; j++) {
-                 int idx = liseq->local_size - (post_start + j);
-                 ADD_INSN1(args, nd_line(node), getlocal, INT2FIX(idx));
-             }
-             ADD_INSN1(args, nd_line(node), newarray, INT2FIX(j));
-             ADD_INSN (args, nd_line(node), concatarray);
-             /* argc is setteled at above */
-         }
-         else {
-             int j;
-             for (j=0; j<post_len; j++) {
-                 int idx = liseq->local_size - (post_start + j);
-                 ADD_INSN1(args, nd_line(node), getlocal, INT2FIX(idx));
-             }
-             argc = INT2FIX(post_len + post_start);
-         }
-     }
- }
+ argc = setup_args(iseq, args, node->nd_args, &flag);
+
+ /* dummy receiver */
+ ADD_INSN1(ret, nd_line(node), putobject,
-         nd_type(node) == NODE_ZSUPER ? Qfalse : Qtrue);
+         (flag & VM_CALL_SUPER_BIT) ? Qfalse : Qtrue);
+ flag &= ~VM_CALL_SUPER_BIT;
+ ADD_SEQ(ret, args);
+ ADD_INSN3(ret, nd_line(node), invokesuper,
+         argc, parent_block, LONG2FIX(flag));
diff --git i/gc.c w/gc.c
index 58e4550..0d5fbad 100644
--- i/gc.c
+++ w/gc.c
@@ -1671,7 +1671,7 @@ gc_mark_children(rb_objspace_t *objspace, VALUE ptr, int lev)
     goto again;

     case NODE_ZARRAY: /* - */
-     case NODE_ZSUPER:
+     case NODE_DELEGATE:
     case NODE_VCALL:
     case NODE_GVAR:
     case NODE_LVAR:
diff --git i/insns.def w/insns.def
index f75007d..6c1efdc 100644

```

```

--- i/insns.def
+++ w/insns.def
@@ -993,10 +993,10 @@ send
 {
     const rb_method_entry_t *me;
     VALUE recv, klass;
-    rb_block_t *blockptr = 0;
+    rb_block_t *blockptr = (op_flag & VM_CALL_SUPER_BIT) ? GET_BLOCK_PTR() : 0;
     int num = caller_setup_args(th, GET_CFP(), op_flag, (int)op_argc,
                               (rb_iseq_t *)blockiseq, &blockptr);
-    rb_num_t flag = op_flag;
+    rb_num_t flag = op_flag & ~VM_CALL_SUPER_BIT;
     ID id = op_id;

     /* get receiver */
diff --git i/node.c w/node.c
index 65bc541..f290d3 100644
--- i/node.c
+++ w/node.c
@@ -408,10 +408,17 @@ dump_node(VALUE buf, VALUE indent, int comment, NODE *node)
     F_NODE(nd_args, "arguments");
     break;

-    case NODE_ZSUPER:
-    ANN("super invocation with no argument");
-    ANN("format: super");
-    ANN("example: super");
+    case NODE_DELEGATE:
+    if (node->nd_state) {
+    ANN("argument delegation with block");
+    ANN("format: ...");
+    ANN("example: foo(...)");
+    }
+    else {
+    ANN("argument delegation without block");
+    ANN("format: ..");
+    ANN("example: foo(..)");
+    }
     break;

     case NODE_ARRAY:
diff --git i/node.h w/node.h
index f8cf7de..74320c0 100644
--- i/node.h
+++ w/node.h
@@ -96,8 +96,8 @@ enum node_type {
 #define NODE_VCALL      NODE_VCALL
     NODE_SUPER,
 #define NODE_SUPER      NODE_SUPER
-    NODE_ZSUPER,
-#define NODE_ZSUPER      NODE_ZSUPER
+    NODE_DELEGATE,
+#define NODE_DELEGATE    NODE_DELEGATE
     NODE_ARRAY,
 #define NODE_ARRAY      NODE_ARRAY
     NODE_ZARRAY,
@@ -414,7 +414,7 @@ typedef struct RNode {
 #define NEW_FCALL(m, a) NEW_NODE(NODE_FCALL, 0, m, a)
 #define NEW_VCALL(m) NEW_NODE(NODE_VCALL, 0, m, 0)
 #define NEW_SUPER(a) NEW_NODE(NODE_SUPER, 0, 0, a)
-#define NEW_ZSUPER() NEW_NODE(NODE_ZSUPER, 0, 0, 0)
+#define NEW_DELEGATE(b) NEW_NODE(NODE_DELEGATE, 0, 0, b)
 #define NEW_ARGS(m, o) NEW_NODE(NODE_ARGS, o, m, 0)
 #define NEW_ARGS_AUX(r, b) NEW_NODE(NODE_ARGS_AUX, r, b, 0)
 #define NEW_OPT_ARG(i, v) NEW_NODE(NODE_OPT_ARG, i, v, 0)
diff --git i/parse.y w/parse.y
index 9fac5bd..735d1bf 100644

```

```

--- i/parse.y
+++ w/parse.y
@@ -2399,6 +2399,20 @@ opt_paren_args : none

  opt_call_args : none
    | call_args
+   | tDOT2
+   {
+     /*%%*/
+     $$ = NEW_DELEGATE(0);
+     /*%
+     %*/
+   }
+   | tDOT3
+   {
+     /*%%*/
+     $$ = NEW_DELEGATE(1);
+     /*%
+     %*/
+   }
+
+   ;

  call_args : command
@@ -3647,7 +3661,7 @@ method_call : operation paren_args
  | keyword_super
  {
    /*%%*/
-   $$ = NEW_ZSUPER();
+   $$ = NEW_SUPER(NEW_DELEGATE(1));
    /*%
    $$ = dispatch0(zsuper);
    %*/

diff --git i/test/ruby/test_method.rb w/test/ruby/test_method.rb
index 7be70b0..a04a285 100644
--- i/test/ruby/test_method.rb
+++ w/test/ruby/test_method.rb
@@ -345,4 +345,38 @@ class TestMethod < Test::Unit::TestCase
  obj.extend(m)
  assert_equal([:m1, :a], obj.public_methods(false), bug)
end

+
+ def test_argument_delegate
+   class << (o = Object.new)
+     def foo(*args)
+       yield(*args)
+     end
+     def foo1(*)
+       foo(...)
+     end
+     def foo2(*)
+       foo1(...)
+     end
+     def bar(*args, &block)
+       [args, block]
+     end
+     def bar1(*)
+       bar(..)
+     end
+     def bar2(*)
+       bar1(..)
+     end
+   end
+   called = nil
+   assert_equal([42], o.foo1(42) {|*x| called = x})
+   assert_equal([42], called)
+   called = nil
+   assert_equal([42], o.foo2(42) {|*x| called = x})

```

```
+ assert_equal([42], called)
+ called = :not_called
+ assert_equal([42], nil, o.bar1(42) {|*x| called = true})
+ assert_equal(:not_called, called)
+ assert_equal([42], nil, o.bar2(42) {|*x| called = true})
+ assert_equal(:not_called, called)
+ end
end
```

```
--
--- Bug
--- Bug
  
```

Related issues:

Related to Ruby trunk - Feature #11256: anonymous block forwarding

Assigned

History

#1 - 06/16/2010 07:11 PM - matz (Yukihiro Matsumoto)

In message "Re: [ruby-dev:41623] [Feature:trunk] argument delegation"
on Wed, 16 Jun 2010 18:57:40 +0900, Nobuyoshi Nakada nobu@ruby-lang.org writes:

<http://www.rubyist.net/~matz/20100615.html#p01>

```
foo(...)
```

```
foo(..)
```

#2 - 06/17/2010 03:37 AM - mame (Yusuke Endoh)

2010 6 16 18:57 Nobuyoshi Nakada nobu@ruby-lang.org:

<http://www.rubyist.net/~matz/20100615.html#p01>

argument delegation

-
-
- `define_method`

```
foo(a, b, c, &)
```

Yusuke Endoh mame@tsg.ne.jp

#3 - 06/17/2010 06:31 AM - matz (Yukihiro Matsumoto)

In message "Re: [ruby-dev:41625] Re: [Feature:trunk] argument delegation"
on Thu, 17 Jun 2010 03:36:56 +0900, Yusuke ENDOH mame@tsg.ne.jp writes:

argument delegation

- 1.
- 2.
3. `define_method`

000000000000000000

1. 000000000000000000000000000000000000
0000000000(000000000000000000000000
0000000000000000)0000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000

2. 00super000000000000super0000000000
0000000000000000000000000000000000
00000000

3. 000000define_method0000000000000000
0000000000000000000000000000000000

000000000000000000000000000000000000

foo(a, b, c, &)

000000000000000000000000000000000000

000000000000000000000000000000000000
000000000000000000000000000000000000
000000200000000000000000000000000000

0000000000000000&+000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
0000000000

0000 0000 /:|)

#4 - 06/17/2010 10:45 AM - ko1 (Koichi Sasada)

00000000

(2010/06/17 6:31), Yukihiro Matsumoto wrote::

argument delegation 00

- 1. 00000000000000 (00000000) 000000
- 2. 0000000000000000000000000000000000
- 3. define_method 0000000000 (00000000000000)

0000000000000000

1. 000000000000000000000000000000000000
0000000000(000000000000000000000000
0000000000000000)0000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000

2. 00super000000000000super0000000000
0000000000000000000000000000000000
00000000

3. 000000define_method0000000000000000
0000000000000000000000000000000000

000000000000000000000000000000000000
0000

00000000000000000000

000000000000000000000000000000000000
000000000000000000000000foo(...) 000000
000000000000000000000000000000000000

0000

<http://rurema.clear-code.com/query:%28...%29/>

~~~~~

~~~~~

```
foo(a, b, c, &)
```

~~~~~

~~~~~

// SASADA Koichi at atdot dot net

#5 - 06/17/2010 11:09 AM - matz (Yukihiro Matsumoto)

~~~~~

In message "Re: [ruby-dev:41629] Re: [Feature:trunk] argument delegation" on Thu, 17 Jun 2010 10:45:11 +0900, SASADA Koichi [ko1@atdot.net](mailto:ko1@atdot.net) writes: <https://bugs.ruby-lang.org/issues/3447#>

```
~~~~~
foo(...)
~~~~~
```

~~~~~

<http://rurema.clear-code.com/query:%28...%29/>

~~~~~

~~~~~  
... Lua
~~~~~

**#6 - 06/17/2010 12:05 PM - akr (Akira Tanaka)**

2010年6月17日 6:31 Yukihiro Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

1. ~~~~~ (~~~~~) ~~~~~

1. ~~~~~ (~~~~~) ~~~~~

~~~~~

~~~~~  
3 ~~~~~

~~~~~  
ruby-talk:162431

~~~~~  
[ruby-talk:162561](http://ruby-talk:162561) [ruby-dev:30892](http://ruby-dev:30892) ~~~~~

~~~~~  
2 ~~~~~

~~~~~



argument delegation  
super(...)  
super

argument delegation  
super(...)  
super

1. define\_method (define\_method)

1. define\_method

define\_method lambda

...

```
def f(a)
  lambda { |b|
    g(...)
  }
end
```

...

...

...

[Tanaka Akira]

#7 - 06/17/2010 01:59 PM - shugo (Shugo Maeda)

...

argument delegation  
super(...)  
super

2010 6 17 6:31 Yukihiro Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

1. super

super  
argument delegation?

```
def foo(&b)
  b = lambda { puts "lambda in foo" }
  bar(...)
end
```

...

Shugo Maeda

#8 - 06/17/2010 02:53 PM - matz (Yukihiro Matsumoto)

...

In message "Re: [ruby-dev:41634] Re: [Feature:trunk] argument delegation"  
on Thu, 17 Jun 2010 13:58:53 +0900, Shugo Maeda [shugo@ruby-lang.org](mailto:shugo@ruby-lang.org) writes:

argument delegation  
super(...)  
super

super

-

- `super` の呼び出し

Ruby の `super` の呼び出しは、  
 呼び出し元オブジェクトの `superclass` を  
 呼び出す。

例として、

- `super` の呼び出し
- `super` の呼び出し

呼び出し元オブジェクトが `super` を呼び出す (呼び出し)

- `super` の呼び出し
- `super` の呼び出し

呼び出し元オブジェクトが `super` を呼び出す `super(...)` の呼び出し

- `super` の呼び出し
- `super` の呼び出し

引数継承 (argument delegation) の呼び出し  
`super` / `super()` の呼び出し

`super` の呼び出しと引数継承の呼び出しの違いは？

```
def foo(&b)
  b = lambda { puts "lambda in foo" }
  bar(...)
end
```

呼び出し

`super` の呼び出しと引数継承の呼び出しの違いは？  
 YARV の呼び出し

```
puts " /: | )"
```

**#9 - 06/17/2010 03:03 PM - ko1 (Koichi Sasada)**

(2010/06/17 14:53), Yukihiro Matsumoto wrote::

```
super の呼び出しと引数継承の呼び出しの違いは？  

YARV の呼び出し
```

呼び出しと引数継承の呼び出しの違いは？

# `zsuper` の呼び出しと `eval` の呼び出し

バージョン 1.9 の呼び出し

// SASADA Koichi at atdot dot net

**#10 - 06/17/2010 03:15 PM - matz (Yukihiro Matsumoto)**

呼び出し

In message "Re: [ruby-dev:41631] Re: [Feature:trunk] argument delegation"  
 on Thu, 17 Jun 2010 12:05:22 +0900, Tanaka Akira [akr@fsij.org](mailto:akr@fsij.org) writes:

```
呼び出しと引数継承の呼び出しの違いは？  

2 の呼び出し
```

```
呼び出しと引数継承の呼び出しの違いは？  

呼び出しと引数継承の呼び出しの違いは？  

呼び出しと引数継承の呼び出しの違いは？
```

```
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
```

```
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
```

```
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
```

000000 define\_method 0 lambda 00000000

... 00000000000000000000000000000000

```
def f(a)
  lambda {|b|
    g(...)}
end
```

000000 ... 00000000000000000000000000

```
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
000000000000000000000000000000000000
```

```
000000000000000000000000000000000000
000000000000000000000000000000000000
```

000000000000000000000000000000000000

```
000000000000000000000000...000000000000
0000000000 ruby-talk:162561 000000000000000000
000000000000000000000000000000000000**0000**obj0000
000000000000000000000000000000000000
```

00000

```
def f(a)
  c = **
  lambda {|b|
    g(**c) # a0000
  }
end
```

```
def f(a)
  lambda {|b|
    g(**) # b0000
  }
end
```

0000000000000000over spec00000000000000

```
0000 0000 /:|)
```

#11 - 06/17/2010 03:17 PM - matz (Yukihiro Matsumoto)

0000 00000000

In message "Re: [ruby-dev:41636] Re: [Feature:trunk] argument delegation" on Thu, 17 Jun 2010 15:03:33 +0900, SASADA Koichi [ko1@atdot.net](mailto:ko1@atdot.net) writes:

```
000000000000000000000000000000000000
000000000000000000000000000000000000
```



- super
- super

super super(...)

5 super

super

super argument delegation

```
def foo(&b)
  b = lambda { puts "lambda in foo" }
  bar(...)
end
```

super YARV

#

Shugo Maeda

#14 - 06/17/2010 07:06 PM - matz (Yukihiro Matsumoto)

In message "Re: [ruby-dev:41641] Re: [Feature:trunk] argument delegation" on Thu, 17 Jun 2010 17:05:10 +0900, Shugo Maeda [shugo@ruby-lang.org](mailto:shugo@ruby-lang.org) writes:

initialize super() super/super()

super: 391 super(): 292

lib/\*\*/\*.rb super() super 66 super 243

5 super

super

/:|)

#15 - 06/17/2010 10:29 PM - mame (Yusuke Endoh)

2010 6 17 6:31 Yukihiro Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

# 1. オブジェクト指向プログラミング (オブジェクト) の基礎

## 1. (D) オブジェクト指向プログラミングの基礎

```
delegate オブジェクト  
オブジェクト指向プログラミング
```

```
lib/ オブジェクト指向プログラミング (オブジェクト  
指向プログラミング) の基礎
```

```
オブジェクト指向プログラミング&1オブジェクト指向  
プログラミング  
オブジェクト指向プログラミング  
オブジェクト指向
```

```
オブジェクト指向プログラミング  
オブジェクト指向 183 (オブジェクト 24) の基礎  
& オブジェクト指向プログラミング
```

```
$ ruby check.rb
```

```
lib/drb/drb.rb
```

```
1079: def method_missing(msg_id, *a, &b)
```

```
1083: return obj.__send__(msg_id, *a, &b)
```

```
lib/test/unit/assertions.rb
```

```
22: def assert_raise(*args, &b)
```

```
23: assert_raises(*args, &b)
```

```
lib/rake.rb
```

```
743: def create_rule(*args, &block)
```

```
744: Rake.application.create_rule(*args, &block)
```

```
lib/rake.rb
```

```
844: def task(*args, &block)
```

```
845: Rake::Task.define_task(*args, &block)
```

```
lib/rake.rb
```

```
862: def file(*args, &block)
```

```
863: Rake::FileTask.define_task(*args, &block)
```

```
lib/rake.rb
```

```
868: def file_create(args, &block)
```

```
869: Rake::FileCreationTask.define_task(args, &block)
```

```
lib/rake.rb
```

```
892: def multitask(args, &block)
```

```
893: Rake::MultiTask.define_task(args, &block)
```

```
lib/rake.rb
```

```
918: def rule(*args, &block)
```

```
919: Rake::Task.create_rule(*args, &block)
```

```
lib/scanf.rb
```

```
607: def scanf(str, &b)
```

```
608: return block_scanf(str, &b) if b
```

```
lib/scanf.rb
```

```
687: def scanf(fstr, &b)
```

```
689: block_scanf(fstr, &b)
```

```
lib/scanf.rb
```

```
716: def scanf(fs, &b)
```

```
717: STDIN.scanf(fs, &b)
```

```
lib/rubygems/user_interaction.rb
```

```
62: def use_ui(new_ui, &block)
```

```
63: Gem::DefaultUserInteraction.use_ui(new_ui, &block)
```

```
lib/optparse.rb
```

```
832: def accept(*args, &blk) top.accept(*args, &blk) end
```

```
836: def self.accept(*args, &blk) top.accept(*args, &blk) end
```

```

lib/optparse.rb
1202: def on(*opts, &block)
1203:   define(*opts, &block)

lib/optparse.rb
1216: def on_head(*opts, &block)
1217:   define_head(*opts, &block)

lib/optparse.rb
1230: def on_tail(*opts, &block)
1231:   define_tail(*opts, &block)

lib/optparse.rb
1425: def visit(id, *args, &block)
1427:   el.send(id, *args, &block)

lib/csv.rb
2308: def CSV(*args, &block)
2309:   CSV.instance(*args, &block)

lib/erb.rb
334:   def percent_line(line, &block)
336:   return @scan_line.call(line, &block)

lib/erb.rb
334:   def percent_line(line, &block)
341:   @scan_line.call(line, &block)

lib/matrix.rb
1457: def map2(v, &block) # :yield: e1, e2
1459:   els = collect2(v, &block)

lib/set.rb
613: def initialize(*args, &block) # :nodoc:
615:   initialize(*args, &block)

lib/delegate.rb
141: def method_missing(m, *args, &block)
144:   target.respond_to?(m) ? target.__send__(m, *args, &block) :
super(m, *args, &block)

lib/open-uri.rb
27: def open(name, *rest, &block) # :doc:
35:   open_uri_original_open(name, *rest, &block)

count: 24

$ cat check.rb
count = 0
Dir.glob("lib/**/*.rb").each do |path|
  next unless File.file?(path)
  open(path) do |file|
    while line = file.gets
      if line =~ /^(\s*)def\s+[\d\w]+(\.(.*&.*\))/ # 000000000000
      #if line =~ /^(\s*)def\s+[\d\w]+(\.(.*&.*\))/ # 0000000000000000
        sig = line
        lineno = file.lineno
        re1 = /^{\s#{ $1.size }}end/
        re2 = Regexp.new(Regexp.quote($2))
        while line = file.gets
          break if re1 =~ line
          if re2 =~ line
            puts path
            puts "#{ lineno }:#{ sig }"
            puts "#{ file.lineno }:#{ line }"
            puts
            count += 1
          end
        end
      end
    end
  end
end

end
end
puts "count: #{ count }"

```

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

#16 - 06/18/2010 05:43 AM - matz (Yukihiko Matsumoto)

0000 000000

In message "Re: [ruby-dev:41644] Re: [Feature:trunk] argument delegation"  
on Thu, 17 Jun 2010 22:29:19 +0900, Yusuke ENDOH [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp) writes:

```
0000000000000000000000000000000000
00000000 183 00 (00 24 000000) 0000000000
& 0000000000000000000000
```

0000000000000000000000000000000000
0000

#17 - 06/18/2010 08:34 AM - shugo (Shugo Maeda)

00000

2010 6 17 19:06 Yukihiko Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

```
00000000 initialize 00000000000000 super()00000000
0000000000000000000000000000000000
00000000000000000000 super/super()000000000000
```

```
super: 39100
super(): 29200
```

000000000000000000000000000000

```
00000000 lib/**/.rb 00000000 super()000000000000
0000000000 super 66 0000000000000000000002
000000000000000000 super 000000000000000000
0000000000 super 243 00000000000000000000
0000000000000000000000000000000000
00
```

~/src/ruby/\*\*/.rb 000000000000
000000 Tk 00000000...0

```
5000000000000000000000000000000000
0000000000000000000000 super 00000000
0000000000000000000000000000000000
```

000000(00000) super 0000000000

0000000000000000000000000000000000
0000000000000000000000000000000000
00000000

Shugo Maeda

#18 - 08/25/2010 12:30 AM - mame (Yusuke Endoh)

00000

2010 6 17 6:31 Yukihiko Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

```
000000000000000000000000&0+000000000000
0000000000000000000000000000000000
0000000000000000000000000000000000
00000000000
```

0000000000000000000000000000000000

```
def foo
```



```
yield
end
```

```
def bar
  foo(&)
end
```

```
bar { p 1 } #=> 1
```

```
#####
```

```
foo &
bar
```

```
##### foo & bar #### foo(&); bar #####
##### foo & bar #####
#####conflict #####
```

```
diff --git a/compile.c b/compile.c
index 2fd804c..3f8c331 100644
--- a/compile.c
+++ b/compile.c
@@ -2863,8 +2863,13 @@ setup_args(rb_iseq_t *iseq, LINK_ANCHOR *args,
NODE *argn, unsigned long *flag)
    INIT_ANCHOR(arg_block);
    INIT_ANCHOR(args_splat);
    if (argn && nd_type(argn) == NODE_BLOCK_PASS) {
-   COMPILER(arg_block, "block", argn->nd_body);
-   *flag |= VM_CALL_ARGS_BLOCKARG_BIT;
+   if ((VALUE)argn->nd_body != (VALUE)-1) {
+     COMPILER(arg_block, "block", argn->nd_body);
+     *flag |= VM_CALL_ARGS_BLOCKARG_BIT;
+   }
+   else {
+     *flag |= VM_CALL_ARGS_BLOCKTRGH_BIT;
+   }
    argn = argn->nd_head;
  }
}
```

```
diff --git a/node.c b/node.c
index 65bc541..85574b4 100644
--- a/node.c
+++ b/node.c
@@ -621,7 +621,12 @@ dump_node(VALUE buf, VALUE indent, int comment, NODE *node)
  ANN("example: foo(x, &blk)");
  F_NODE(nd_head, "other arguments");
  LAST_NODE;
- F_NODE(nd_body, "block argument");
+ if ((VALUE)node->nd_body != (VALUE)-1) {
+   F_NODE(nd_body, "block argument");
+ }
+ else {
+   F_MSG(nd_body, "block argument", "-1 (anonymous)");
+ }
  break;
}
```

```
case NODE_DEFN:
diff --git a/parse.y b/parse.y
index e085088..b0b946b 100644
--- a/parse.y
+++ b/parse.y
@@ -2459,6 +2459,14 @@ block_arg : tAMPER arg_value
    $$ = $2;
    %*/
  }
+ | tAMPER
+ {
+   /*%%*/
+   $$ = NEW_BLOCK_PASS(-1);
+   /*%
+   $$ = dispatch0(anonymous_block_arg);
+   %*/
+ }
;
;
```

```
opt_block_arg : ',' block_arg
diff --git a/vm_core.h b/vm_core.h
index 7676b2f..f0fa3a3 100644
--- a/vm_core.h
+++ b/vm_core.h
@@ -544,6 +544,7 @@ typedef struct {
 #define VM_CALL_TAILRECURSION_BIT (0x01 << 6)
 #define VM_CALL_SUPER_BIT (0x01 << 7)
 #define VM_CALL_OPT_SEND_BIT (0x01 << 8)
+#define VM_CALL_ARGS_BLOCKTRGH_BIT (0x01 << 9)

 #define VM_SPECIAL_OBJECT_VMCORE 0x01
 #define VM_SPECIAL_OBJECT_CBASE 0x02
diff --git a/vm_inshelper.c b/vm_inshelper.c
index 985a2fb..2a127d7 100644
--- a/vm_inshelper.c
+++ b/vm_inshelper.c
@@ -261,6 +261,10 @@ caller_setup_args(const rb_thread_t *th,
 rb_control_frame_t *cfp, VALUE flag,
     *block = blockptr;
 }
}
+ else if (flag & VM_CALL_ARGS_BLOCKTRGH_BIT) {
+     rb_control_frame_t *reg_cfp = cfp;
+     *block = GET_BLOCK_PTR();
+ }
+
 else if (blockiseq) {
     blockptr = RUBY_VM_GET_BLOCK_PTR_IN_CFP(cfp);
     blockptr->iseq = blockiseq;
```

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

**#19 - 08/25/2010 03:23 AM - ko1 (Koichi Sasada)**

XXXXXXXX

XXXXXXXXXXXXXXXXXXXX

(2010/08/25 0:30), Yusuke ENDOH wrote:

```
--- a/vm_core.h
+++ b/vm_core.h
@@ -544,6 +544,7 @@ typedef struct {
 #define VM_CALL_TAILRECURSION_BIT (0x01 << 6)
 #define VM_CALL_SUPER_BIT (0x01 << 7)
 #define VM_CALL_OPT_SEND_BIT (0x01 << 8)
+#define VM_CALL_ARGS_BLOCKTRGH_BIT (0x01 << 9)

 #define VM_SPECIAL_OBJECT_VMCORE 0x01
 #define VM_SPECIAL_OBJECT_CBASE 0x02
diff --git a/vm_inshelper.c b/vm_inshelper.c
index 985a2fb..2a127d7 100644
--- a/vm_inshelper.c
+++ b/vm_inshelper.c
@@ -261,6 +261,10 @@ caller_setup_args(const rb_thread_t *th,
 rb_control_frame_t *cfp, VALUE flag,
     *block = blockptr;
 }
}
+ else if (flag & VM_CALL_ARGS_BLOCKTRGH_BIT) {
+     rb_control_frame_t *reg_cfp = cfp;
+     *block = GET_BLOCK_PTR();
+ }
+
 else if (blockiseq) {
     blockptr = RUBY_VM_GET_BLOCK_PTR_IN_CFP(cfp);
     blockptr->iseq = blockiseq;
```

XXXXXXVM\_CALL\_ARGS\_BLOCKTRGH\_BIT XXXXXXXXXX  
XXVM\_CALL\_ARGS\_BLOCKARG\_BIT XXXXXblockiseqXXXXXXXXXXXX  
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

--  
// SASADA Koichi at atdot dot net

#20 - 08/25/2010 09:07 PM - mame (Yusuke Endoh)

□□□□

2010□8□25□3:23 SASADA Koichi [ko1@atdot.net](mailto:ko1@atdot.net):

```
□□□□□ VM_CALL_ARGS_BLOCKTRGH_BIT □□□□□□□□□□  
□□ VM_CALL_ARGS_BLOCKARG_BIT □□□□□ blockiseq □□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
□□□□□□□□□□□□□□□□□□□□□□□□□□□□  
□□□□□ VM_CALL_ARGS_BLOCKTRGH_BIT □□□□□□ 2 □□□□□□□□□□  
□□□□□ (□□□□□ Proc □□□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□) □□□□□□□□□□□□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
□□ blockiseq -1 □□□□□□ BLOCKTRGH □□□□□□□□□□□□□□□□  
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□  
# □ trunk □□□□□□□□□□□ test-all □□□□□□□□□□
```

```
diff --git a/compile.c b/compile.c  
index 2fd804c..1c04640 100644  
--- a/compile.c  
+++ b/compile.c  
@@ -1411,11 +1411,18 @@ iseq_set_sequence(rb_iseq_t *iseq, LINK_ANCHOR *anchor)  
    case TS_ISEQ: /* iseq */  
        {  
            VALUE v = operands[j];  
-           rb_iseq_t *block = 0;  
+           VALUE block = 0;  
            if (v) {  
-           GetISegPtr(v, block);  
+           if (v == Qundef) {  
+               block = (VALUE)-1;  
+           }  
+           else {  
+               rb_iseq_t *blk;  
+               GetISegPtr(v, blk);  
+               block = (VALUE)blk;  
+           }  
-           generated_iseq[pos + 1 + j] = (VALUE)block;  
+           generated_iseq[pos + 1 + j] = block;  
            break;  
        }  
    case TS_VALUE: /* VALUE */  
@@ -2853,7 +2860,7 @@ add_ensure_iseq(LINK_ANCHOR *ret, rb_iseq_t *iseq, int is_return)  
    }  
  
    static VALUE  
-setup_args(rb_iseq_t *iseq, LINK_ANCHOR *args, NODE *argn, unsigned long *flag)  
+setup_args(rb_iseq_t *iseq, LINK_ANCHOR *args, NODE *argn, VALUE *blockiseq, unsigned long *flag)  
    {  
        VALUE argc = INT2FIX(0);  
        int nsplat = 0;  
@@ -2863,7 +2870,13 @@ setup_args(rb_iseq_t *iseq, LINK_ANCHOR *args, NODE *argn, unsigned long *flag)  
        INIT_ANCHOR(arg_block);  
        INIT_ANCHOR(args_splat);  
        if (argn && nd_type(argn) == NODE_BLOCK_PASS) {  
-        COMPILE(arg_block, "block", argn->nd_body);  
+        if ((VALUE)argn->nd_body != (VALUE)-1) {  
+            COMPILE(arg_block, "block", argn->nd_body);  
+        }  
+        else {  
+            arg_block = 0;  
+            if (blockiseq) *blockiseq = Qundef;  
+        }  
        *flag |= VM_CALL_ARGS_BLOCKARG_BIT;  
        argn = argn->nd_head;
```

```

    }
@@ -2932,7 +2945,7 @@ setup_args(rb_iseq_t *iseq, LINK_ANCHOR *args, NODE *argn, unsigned long *flag)
    ADD_SEQ(args, args_splat);
    }

-   if (*flag & VM_CALL_ARGS_BLOCKARG_BIT) {
+   if ((*flag & VM_CALL_ARGS_BLOCKARG_BIT) && arg_block) {
    ADD_SEQ(args, arg_block);
    }
    return argc;
@@ -3776,7 +3789,7 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)
    boff = 1;
    default:
        INIT_ANCHOR(args);
-       argc = setup_args(iseq, args, node->nd_args->nd_head, &flag);
+       argc = setup_args(iseq, args, node->nd_args->nd_head, 0, &flag);
        ADD_SEQ(ret, args);
    }
    ADD_INSN1(ret, nd_line(node), dupn, FIXNUM_INC(argc, 1 + boff));
@@ -4083,7 +4096,7 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)

    /* args */
    if (nd_type(node) != NODE_VCALL) {
-       argc = setup_args(iseq, args, node->nd_args, &flag);
+       argc = setup_args(iseq, args, node->nd_args, &parent_block, &flag);
    }
    else {
        argc = INT2FIX(0);
@@ -4121,7 +4134,7 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)
    INIT_ANCHOR(args);
    iseq->compile_data->current_block = Qfalse;
    if (nd_type(node) == NODE_SUPER) {
-       argc = setup_args(iseq, args, node->nd_args, &flag);
+       argc = setup_args(iseq, args, node->nd_args, &parent_block, &flag);
    }
    else {
        /* NODE_ZSUPER */
@@ -4294,7 +4307,7 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)
    }

    if (node->nd_head) {
-       argc = setup_args(iseq, args, node->nd_head, &flag);
+       argc = setup_args(iseq, args, node->nd_head, 0, &flag);
    }
    else {
        argc = INT2FIX(0);
@@ -4906,7 +4919,7 @@ iseq_compile_each(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE * node, int popped)

    INIT_ANCHOR(recv);
    INIT_ANCHOR(args);
-   argc = setup_args(iseq, args, node->nd_args, &flag);
+   argc = setup_args(iseq, args, node->nd_args, 0, &flag);

    if (node->nd_recv == (NODE *) 1) {
        flag |= VM_CALL_FCALL_BIT;
diff --git a/insns.def b/insns.def
index fcd97ae..10b99be 100644
--- a/insns.def
+++ b/insns.def
@@ -989,7 +989,7 @@ DEFINE_INSN
send
(ID op_id, rb_num_t op_argc, ISEQ blockiseq, rb_num_t op_flag, IC ic)
(...)
-(VALUE val) // inc += - (int)(op_argc + ((op_flag & VM_CALL_ARGS_BLOCKARG_BIT) ? 1 : 0));
+(VALUE val) // inc += - (int)(op_argc + (((op_flag & VM_CALL_ARGS_BLOCKARG_BIT) && (VALUE)blockiseq != (VALUE)
)-1) ? 1 : 0));
{
    const rb_method_entry_t *me;
    VALUE recv, klass;
@@ -1017,7 +1017,7 @@ DEFINE_INSN
invokesuper
(rb_num_t op_argc, ISEQ blockiseq, rb_num_t op_flag)
(...)
-(VALUE val) // inc += - (int)(op_argc + ((op_flag & VM_CALL_ARGS_BLOCKARG_BIT) ? 1 : 0));
+(VALUE val) // inc += - (int)(op_argc + (((op_flag & VM_CALL_ARGS_BLOCKARG_BIT) && (VALUE)blockiseq != (VALUE)

```

```

)-1) ? 1 : 0));
{
    rb_block_t *blockptr = !(op_flag & VM_CALL_ARGS_BLOCKARG_BIT) ? GET_BLOCK_PTR() : 0;
    int num = caller_setup_args(th, GET_CFP(), op_flag,
diff --git a/node.c b/node.c
index 65bc541..85574b4 100644
--- a/node.c
+++ b/node.c
@@ -621,7 +621,12 @@ dump_node(VALUE buf, VALUE indent, int comment, NODE *node)
    ANN("example: foo(x, &blk)");
    F_NODE(nd_head, "other arguments");
    LAST_NODE;
-   F_NODE(nd_body, "block argument");
+   if ((VALUE)node->nd_body != (VALUE)-1) {
+       F_NODE(nd_body, "block argument");
+   }
+   else {
+       F_MSG(nd_body, "block argument", "-1 (anonymous)");
+   }
    break;

    case NODE_DEFN:
diff --git a/parse.y b/parse.y
index e085088..b0b946b 100644
--- a/parse.y
+++ b/parse.y
@@ -2459,6 +2459,14 @@ block_arg      : tAMPER arg_value
                $$ = $2;
                %*/
                }
+       | tAMPER
+       {
+           /*%%*/
+           $$ = NEW_BLOCK_PASS(-1);
+           /*%
+           $$ = dispatch0(anonymous_block_arg);
+           %*/
+       }
+       ;

    opt_block_arg : ',' block_arg
diff --git a/tool/instruction.rb b/tool/instruction.rb
index 4fd2127..4ce219f 100755
--- a/tool/instruction.rb
+++ b/tool/instruction.rb
@@ -66,13 +66,12 @@ class RubyVM
    ret = "int inc = 0;\n"

    @opes.each_with_index{|(t, v), i|
-       if t == 'rb_num_t' && ((re = /\b#{v}\b/n) =~ @sp_inc ||
-                               @defopes.any?{|t, val| re =~ val})
+       if ((re = /\b#{v}\b/n) =~ @sp_inc || @defopes.any?{|t, val| re =~ val})
            ret << "        int #{v} = FIX2INT(opes[#{i}]);\n"
        end
    }
    @defopes.each_with_index{|(t, var), val), i|
-       if t == 'rb_num_t' && val != '*' && /\b#{var}\b/ =~ @sp_inc
+       if val != '*' && /\b#{var}\b/ =~ @sp_inc
            ret << "        #{t} #{var} = #{val};\n"
        end
    }
diff --git a/vm_inshelper.c b/vm_inshelper.c
index 985a2fb..f6418b7 100644
--- a/vm_inshelper.c
+++ b/vm_inshelper.c
@@ -240,25 +240,31 @@ caller_setup_args(const rb_thread_t *th, rb_control_frame_t *cfp, VALUE flag,

    if (block) {
        if (flag & VM_CALL_ARGS_BLOCKARG_BIT) {
-           rb_proc_t *po;
-           VALUE proc;
-
-           proc = *(--cfp->sp);
-
-           if (proc != Qnil) {

```

```

-     if (!rb_obj_is_proc(proc)) {
-         VALUE b = rb_check_convert_type(proc, T_DATA, "Proc", "to_proc");
-         if (NIL_P(b) || !rb_obj_is_proc(b)) {
-             rb_raise(rb_eTypeError,
-                 "wrong argument type %s (expected Proc)",
-                 rb_obj_classname(proc));
+         if ((VALUE)blockiseq != (VALUE)-1) {
+             rb_proc_t *po;
+             VALUE proc;
+
+             proc = *(--cfp->sp);
+
+             if (proc != Qnil) {
+                 if (!rb_obj_is_proc(proc)) {
+                     VALUE b = rb_check_convert_type(proc, T_DATA, "Proc", "to_proc");
+                     if (NIL_P(b) || !rb_obj_is_proc(b)) {
+                         rb_raise(rb_eTypeError,
+                             "wrong argument type %s (expected Proc)",
+                             rb_obj_classname(proc));
+                     }
+                     proc = b;
+                 }
-                 proc = b;
+                 GetProcPtr(proc, po);
+                 blockptr = &po->block;
+                 RUBY_VM_GET_BLOCK_PTR_IN_CFP(cfp)->proc = proc;
+                 *block = blockptr;
+             }
-             GetProcPtr(proc, po);
-             blockptr = &po->block;
-             RUBY_VM_GET_BLOCK_PTR_IN_CFP(cfp)->proc = proc;
-             *block = blockptr;
+             }
+             else {
+                 rb_control_frame_t *reg_cfp = cfp;
+                 *block = GET_BLOCK_PTR();
+             }
+         }
+     }
+     else if (blockiseq) {

```

---

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

**#21 - 08/26/2010 11:28 PM - yugui (Yuki Sonoda)**

Yugui

2010/6/17 Yukihiro Matsumoto [matz@ruby-lang.org](mailto:matz@ruby-lang.org):

```

#####&;#####
#####
#####
#####
#####

```

```

#####&block#####
#####&block#####

```

```

#####&#####

```

```

foo(a, b, &yield)

```

```

#####yield block.call#####

```

---

Yuki Sonoda (Yugui)

[yugui@yugui.jp](mailto:yugui@yugui.jp)  
<http://yugui.jp>

**#22 - 09/02/2010 03:06 AM - mame (Yusuke Endoh)**

2010 8 26 23:28 Yugui [yugui@yugui.jp](mailto:yugui@yugui.jp):

日本語のコメントと英語のコメント

日本語のコメント

```
func(foo + bar, baz * qux, i) {  
  & i  
}
```

```
foo(a, b, &yield)
```

```
yield {  
  block.call  
}
```

日本語のコメント

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

#### #23 - 03/18/2012 05:19 PM - shyouhei (Shyouhei Urabe)

- Description updated

- Status changed from Open to Feedback

- 日本語のコメント
- 英語のコメント

#### #24 - 05/24/2012 11:54 PM - nobu (Nobuyoshi Nakada)

- Description updated

#### #25 - 11/20/2012 10:54 PM - mame (Yusuke Endoh)

- Status changed from Feedback to Assigned

- Assignee set to matz (Yukihiko Matsumoto)

- Target version set to 2.6

shyouhei (Shyouhei Urabe) wrote:

- 日本語のコメント

matz 日本語 go ahead 日本語2.0.0 日本語 feature deadline 日本語 next minor 日本語

- 日本語のコメント

日本語 matz 日本語 & 日本語

```
def foo; ... yield ... end  
foo { ... }
```

日本語 foo 日本語 bar 日本語

```
def bar; ... yield ... end  
def foo(&blk); ... bar(&blk) ... end  
foo { ... }
```

日本語 blk 日本語 and/or 日本語 Proc 日本語

Yusuke Endoh [mame@tsg.ne.jp](mailto:mame@tsg.ne.jp)

#### #26 - 02/14/2014 10:10 AM - nobu (Nobuyoshi Nakada)

- Description updated

#### #27 - 06/22/2015 07:44 AM - matz (Yukihiko Matsumoto)

- Related to Feature #11256: anonymous block forwarding added

#### #28 - 12/25/2017 06:14 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)