I'm proposing a method called Proc#call_under (the name could be discussed) which both unifies instance_eval and instance_exec, and makes it possible to call a Proc with a block and a scope:

Proc#call_under(self, *args, &blk):

```
proc  { self }.call_under(1)  #=> 1
proc  { |a| self + a }.call_under(1, 2)  #=> 3
proc  { |&b| self + b.call }.call_under(2)  { 2 }  #=> 4
```

**Related issues:**
Related to Ruby master - Feature #6298: Proc#+  Rejected  04/15/2012

---

What is self here, or what should it be? Such that:

```ruby
module M
  A = Proc.new{ self }
end

bc = BlockCollection.new(A)
bc.call  #=> M

'F'.instance_eval(&bc)  #=> 'F'
```

---

Hello,

I made a proof-of-concept patch.
Not tested yet. Please try it and find a bug.
It (and some related functions) seem to need some refactoring work because it calls directly invoke_block_from_c which is very internal function.

```ruby
$ ./miniruby -e 'p proc { self }.call_under(1)
p proc { |a| self + a }.call_under(1, 2)
p proc { |&b| self + b.call }.call_under(2) { 2 }
'
1
3
4
```

diff --git a/proc.c b/proc.c
index d44e8d8..7ad490e 100644
--- a/proc.c
+++ b/proc.c
@@ -567,6 +567,22 @@ proc_call(int argc, VALUE *argv, VALUE procval)
    return vret;
+
+VALUE rb_proc_call_under(VALUE procval, VALUE under, VALUE self, VALUE values);
+
+static VALUE
+proc_call_under(int argc, VALUE *argv, VALUE procval) {
+    VALUE self, klass, values;
+    rb_scan_args(argc, argv, "1*", &self, &values);
+    if (SPECIAL_CONST_P(self)) {
+        klass = Qnil;
+    } else {
+        klass = rb_singleton_class(self);
+    }
+    return rb_proc_call_under(procval, klass, self, values);
+}
+
+#if SIZEOF_LONG > SIZEOF_INT
+static inline int
+check_argc(long argc)
@@ -2183,6 +2199,7 @@ Init_Proc(void)
     rb_define_method(rb_cProc, "[]", proc_call, -1);
     rb_define_method(rb_cProc, "==="", proc_call, -1);
     rb_define_method(rb_cProc, "yield", proc_call, -1);
+    rb_define_method(rb_cProc, "call_under", proc_call_under, -1);
 #endif
     rb_define_method(rb_cProc, "to_proc", proc_to_proc, 0);
     rb_define_method(rb_cProc, "arity", proc_arity, 0);
-diff --git a/vm_eval.c b/vm_eval.c
index 6c26b97..562a215 100644
--- a/vm_eval.c
+++ b/vm_eval.c
@@ -1262,6 +1262,30 @@ yield_under(VALUE under, VALUE self, VALUE values)
     }
 }

+static inline VALUE
+invoke_block_from_c(rb_thread_t *th, const rb_block_t *block,
+    VALUE self, int argc, const VALUE *argv, const NODE *cref);
+
+VALUE
+rb_proc_call_under(VALUE procval, VALUE under, VALUE self, VALUE values) {
+    rb_thread_t *th = GET_THREAD();
+    rb_block_t *block;
+    NODE *cref;
+    rb_proc_t *proc;
+    GetProcPtr(procval, proc);
+    block = proc->block;
+    block.self = self;
+    cref = vm_cref_push(th, under, NOEX_PUBLIC, &proc->block);
+    cref->flags |= NODE_FL_CREF_PUSHED_BY_EVAL;
+
```

```
```
/* string eval under the class/module context */
static VALUE
eval_under(VALUE under, VALUE self, VALUE src, const char *file, int line)

--
Yusuke Endoh mame@tsg.ne.jp

#4 - 11/20/2012 10:34 PM - mame (Yusuke Endoh)
- Target version set to 2.6

#5 - 06/29/2013 06:18 AM - judofyr (Magnus Holm)
I concur with trans: I don't think this issue will solve #6298. However, please keep that discussion out of this issue.

Would it be possible to get a OK/NG for inclusion of this in 2.1?

Still not sure about the name. Maybe #call_with is better?

#6 - 06/29/2013 06:52 AM - phluid61 (Matthew Kerwin)
judofyr (Magnus Holm) wrote:

Still not sure about the name. Maybe #call_with is better?

I suggest #call_as or #call_bound: "as" because it suggests that "self" in the proc will refer to the first parameter, or possibly "bound" because we're binding the proc to the object (as well as to its existing binding scope). However I don't have strong feelings about any name, except that I like #call_with less than any others.

A side discussion: how does this tie in with currying? For example, do you foresee a Proc#curry_under (or curry_self, or curry_with, or whatever) method, that would cause the proc to be passed to instance_exec when all its parameters are satisfied?

#7 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)
- Target version deleted (2.6)

#8 - 04/19/2018 07:57 AM - nobu (Nobuyoshi Nakada)
- Description updated

Files
| proc_call_under.patch | 2.06 KB | 04/17/2012 | mame (Yusuke Endoh) |