Making Ruby?

ゆるふわRuby生活

Nobuyoshi Nakada / 中田伸悦

Salseforce.com / Heroku

Self-introduction

Fulltime Ruby Committer @
Salesforce.com / Heroku
(2011~)

So called Matz team

- Matz
- Nobu
- ~~Ko1~~

日常/Daily

- デバッグ/Debugging
- 新機能/New features
- バグ/Bug making
- 家事・育児/Housekeeping etc

About Ruby development

- Repository
- Issues
- Developers’ meeting

Repository

Subversion

svn+ssh://svn@ci.ruby-lang.org/ruby
https://svn.ruby-lang.org/cgi-bin/viewvc.cgi/

Git mirror

https://github.com/ruby/ruby

Why not Git?

- ruby is older than Git
- moving to git needs some works
- hash is not clear as revision number
- Windows is not supported **officially**

Not enough advantage

Issues

- Redmine
  https://bugs.ruby-lang.org/projects/ruby-trunk/issues
- Mailing List
  - ruby-core@ruby-lang.org (en)
  - ruby-dev@ruby-lang.org (ja)

Developers' meeting

- Once per month
- In Tokyo (usually)
  in Kyoto 2016/9
- Taking inventory of bug tickets

How to build Ruby (from tarball)

Similar to other OSS

configure

+

make

Out-of-Place build

- Various configure options
- Virtual machines

  Linux, Windows, …
- GNU Makefile to build at once

  https://github.com/nobu/build-files/blob/master/Ruby.mk

Various configure options

Many build directories by combination

- --enable-shared
- --with-arch
- optflags
- etc

How to Build Ruby (from repo)

- subversion / git(mirror)
- autoconf
- bison
- gperf
- ruby

To build Ruby, you need Ruby

Ruby

- BASERUBY

  pre-installed ruby

- MINIRUBY

  ruby made during the build

BASERUBY

Ruby (maybe old) to generate source files

- parse.y → parse.c, …
- defs/id.def → id.h, id.c
- insns.def → vm.inc, insns.inc, …
- etc…

MINIRUBY

To generate Makefiles for extension libraries, and others

MINIRUBY’s feature

No dynamic loading

↓

Runnable alone

- No LD_LIBRARY_PATH
- Convenient for debugging

MINIRUBY’ s limitation

Feature can be also a limitation

- Unable to load extension libraries
  - b/c restriction of Windows DLL
    - can’ t share exts with normal ruby
  - built-in encodings only
    - ASCII-8BIT, US-ASCII, UTF-8
    - for -K option: EUC-JP, Shift_JIS

Building encodings

To generate Makefile by erb

no details

Similar to exts but bit simpler

Building extension libraries

- execute extconf.rb files under ext and gems directories

```
Dir.glob("{ext,gems}/**/extconf.rb") do |file|
  load(file)
end
```

- generate dedicated Makefile (exts.mk)

Parallel build (~2.4)

- building miniruby ⇒ parallel
  (w/ GNU make)
- building extension libraries ⇒ parallel
- each extconf.rb ⇒ sequential
  - ⇒ making exts.mk is slow

Parallel build (2.5)

- run each directories underneath ext and gems
  - no dependencies each other
  - depends on the parent only
- composite each exts.mk files
  - ⇒ faster configuration

Problem

No headers and libraries are installed at build

- C headers provided by ruby
  ruby.h, etc
- library files provided by ruby
  libruby.so, etc

Solution

Mimic global variables used in mkmf.rb by **trace_var**

- $extmk
- $ruby

trace_var

Hook changes of a global variable

```
  trace_var(symbol, cmd )             -> nil
  trace_var(symbol) {|val| block }    -> nil

Controls tracing of assignments to global variables. The parameter symbol
identifies the variable (as either a string name or a symbol identifier).
cmd (which may be a string or a Proc object) or block is executed
whenever the variable is assigned. The block or Proc object receives the
variable's new value as a parameter. Also see Kernel::untrace_var.

  trace_var :$_, proc {|v| puts "$_ is now '#{v}'" }
  $_ = "hello"
  $_ = ' there'

produces:

  $_ is now 'hello'
  $_ is now ' there'
```

$extmk

Flag to handle bundled exts in mkmf.rb

- set source directory from build directory
- set built extension directory
- chain $ruby hook

$ruby

Path to ruby to run

- set up RbConfig configurations
- set $ruby path

Bug Report

[ruby-list:50578]

w/o local variable

```
# p = 2
p (-1.3).abs #=> 1.3
```

w/ local variable

```
p = 2
p (-1.3).abs #=> -1.3
```

Exactly Not-A-Bug

Ancient Spec

at least 1.1

Just-size bug

- “Demon Castle parse.y” by mame
- “Monstrous” lex_state
- But not so hard

NOT SO HARD?

-w option

```
$ ruby -w -e 'p=2; p (-1.3).abs'
-e:1: warning: don't put space
 before argument parentheses
```

parser_yylex()

the lexical analysis

```
case '(':
    // ...
    else if (lex_state == (EXPR_END|EXPR_LABEL) && space_seen) {
        rb_warning0("don't put space before argument parentheses");
    }
    // ...
    SET_LEX_STATE(EXPR_BEG|EXPR_LABEL);
```

What’s space_seen?

a space was seen just before the current token?

```
p (-1.3).abs
 ^-----------!Here!
```

lex_state

state of lexer

(EXPR_END|EXPR_LABEL) …

What's EXPR_END?

Able to end an expression

- just after right paren of method
- just after method name w/o paren
- just after method arg w/o paren
- …

What’s EXPR_LABEL?

Able to place a label

- just after left paren of method
- just after method name w/o paren
- just after method arg w/o paren

In parse_ident()

parse an identifier starts with a lower letter (local variable / method)

```
ident = tokenize_ident(parser, last_state);
if (!IS_lex_state_for(last_state, EXPR_DOT|EXPR_FNAME) &&
    (result == tIDENTIFIER) && /* not EXPR_FNAME, not attrasgn */
    lvar_defined(ident)) {
    SET_LEX_STATE(EXPR_END|EXPR_LABEL);
}
```

What’ s lvar_defined(ident)?

Prediction to tell “whether the name referred as ident (p here) is
defined as a local variable in the current scope”

Rules

W/o variable

```
primary                 : tLPAREN_ARG
```

W/ variable

```
paren_args      : '(' opt_call_args rparen
```

How to Fix?

Remove the condition by lvar_defined

I consider it a bug, but…

literal symbol by intern

- compile.c (iseq_compile_each0): literal symbol should not be affected by redefinition of String#intern method.
- vm_insnhelper.c (rb_vm_str_intern): intern a string into a symbol directly.

literal symbol by intern

```
:"#{foo}"
```

[Feature #13812]

Refinements can’ t affect string interpolation

Difference

Conversion is explicitly visible or not

New Features

> *No eye-catcher in 2.5*

Such as &. in 2.3

“Unicode case” in 2.4

Approved

- `rescue inside do/end`
- `Array#append, prepend`
- `Hash#transform_keys`
- `Kernel#yield_self`
- …

Rejected

- neko ^..^ operator (in Perl6)
- User-defined operator

Under Discussion

- Method extraction operator
  - Kernel#method -> Method instance
- Rightward assignment

Write Ruby

Wouldn’ t you write **New Ruby**?