Write Your C Extension for Ruby

簡煒航 (Jian Weihang)
@tonytonyjan

Bonjour

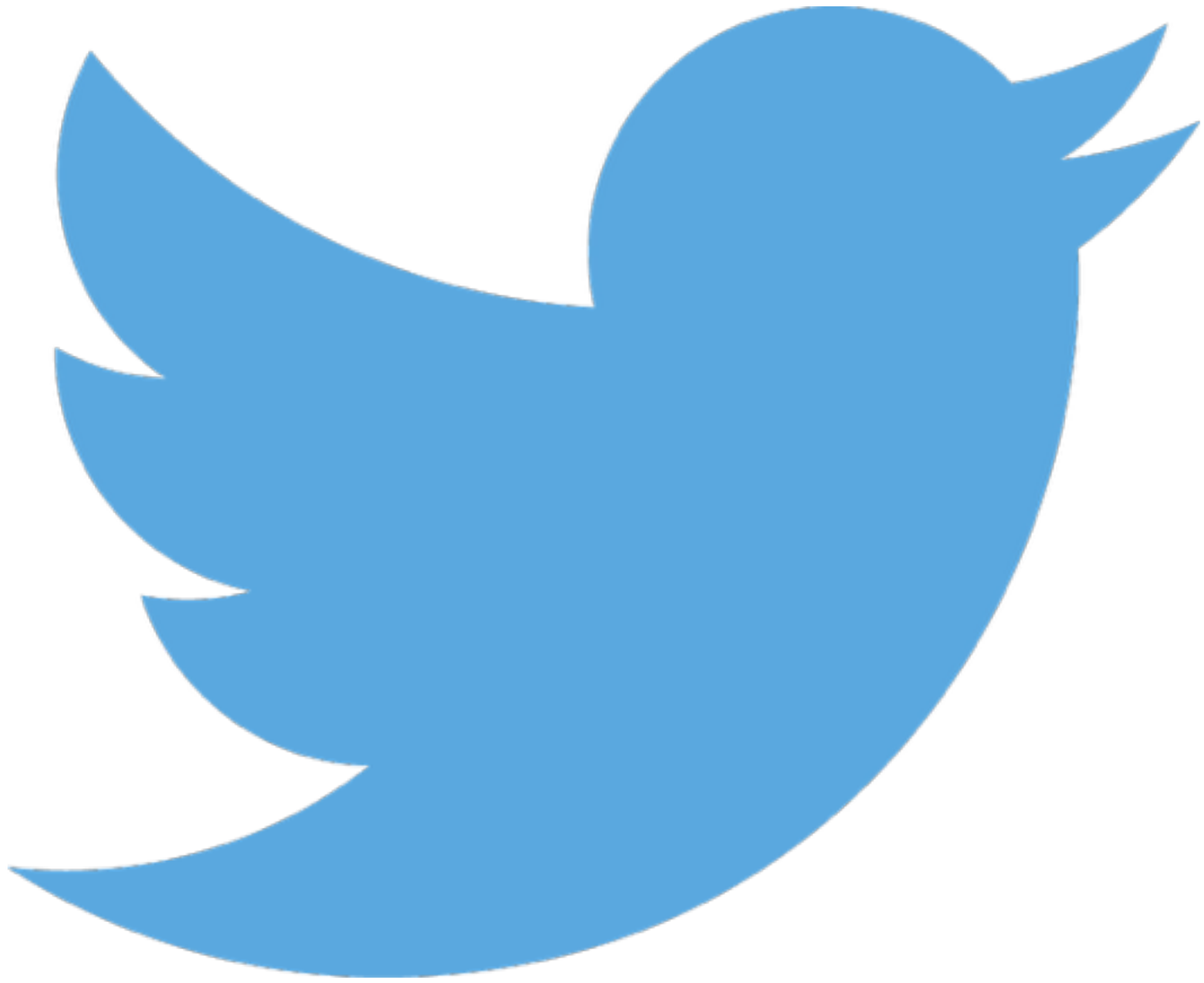
簡煒航

Jian, Weihang

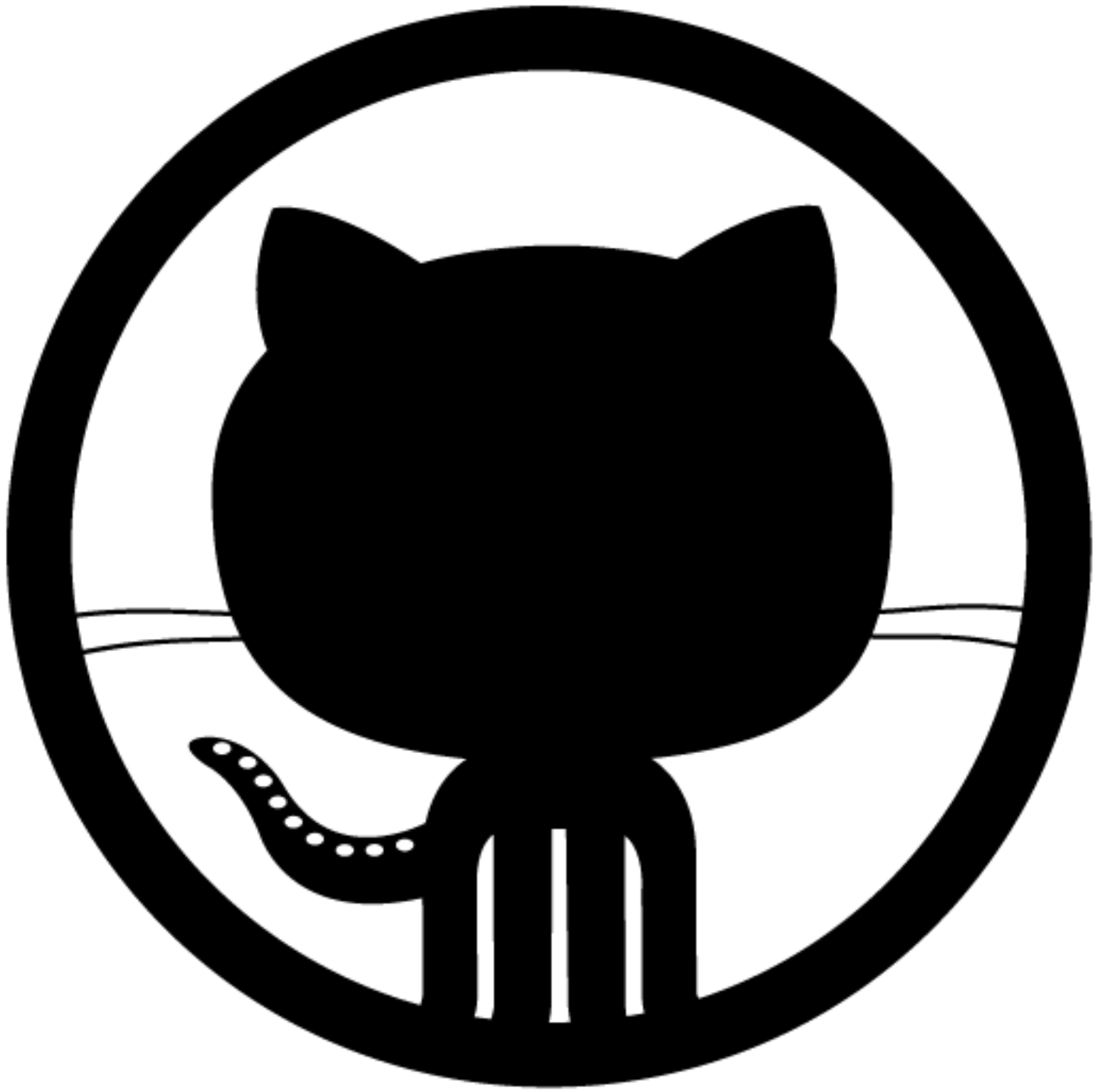
tonytonyjan.net

tonytonyjan

tonytonyjan



tonytonyjan



tonytonyjan

tonytonyjan

tonytonyjan

YouTube

Double Keyboard Player

Ruby

Postgraduate

Freelancer

Book Writer

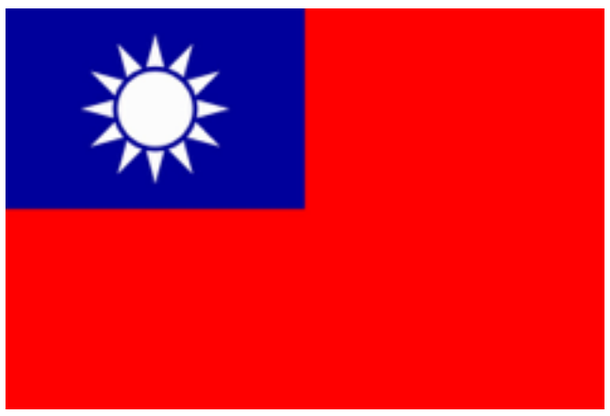
Coach of Rails Girls Taipei

Startup

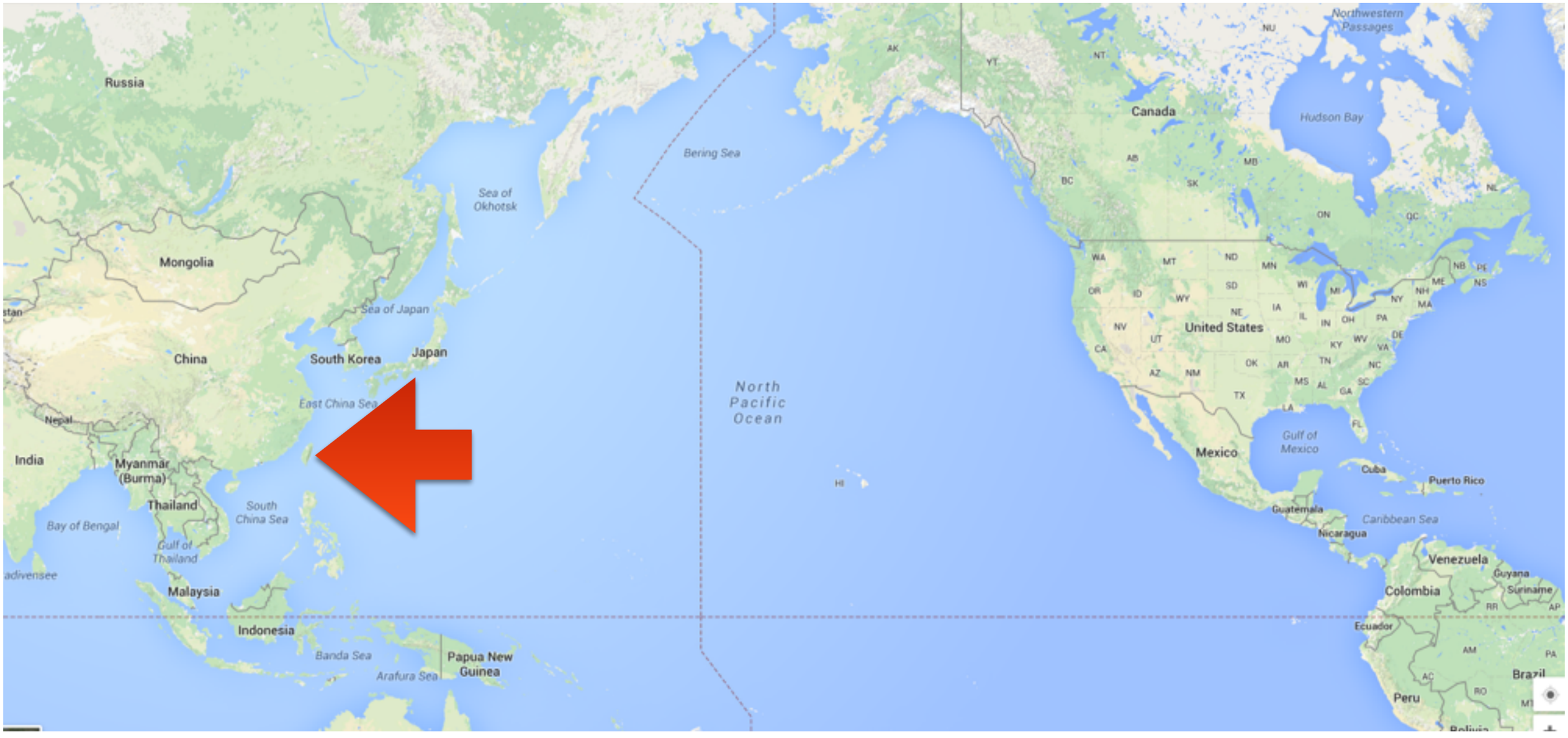
brainana.com

5xruby.tw

Taiwan



臺灣（Taiwan）


Russia
Mongolia
China
South Korea
Japan
Sea of Japan
East China Sea
South China Sea
Bering Sea
Sea of Okhotsk
North Pacific Ocean
Nepal
India
Myanmar (Burma)
Thailand
Bay of Bengal
Gulf of Thailand
Malaysia
Indonesia
Banda Sea
Arafura Sea
Papua New Guinea
AK
YT
NT
NU
Canada
Northwestern Passages
Hudson Bay
BC
AB
SK
MB
ON
QC
NL
United States
Mexico
Gulf of Mexico
Cuba
Puerto Rico
Guatemala
Nicaragua
Caribbean Sea
Venezuela
Guyana
Suriname
Colombia
Ecuador
Peru
Brazil

臺灣（Taiwan）


평양
South Korea
Nagoya
名古屋
Japan
Gwangju
광주
Busan
부산
Tokyo
東京
China
Osaka
大阪
Shanghai
上海
Fukuoka
福岡
Chengdu
成都
East China Sea
Hangzhou
杭州
Chongqing
重庆
Guangzhou
广州
Taiwan
Hanoi
Hong Kong
香港
Laos
Thailand
Kalusunan
Manila
South
China Sea
Vietnam
Cambodia
Philippines
Kabisayaan
Gulf of
Ho Chi

2015-02-18

Montreal	Taipei
-18°C	19°C

Happy Chinese New Year

It’s Year of the ~~RAM~~ Ram

Write Your C Extension for Ruby

簡煒航 (Jian Weihang)
@tonytonyjan

Overview

- Compilation
- File Structure
- Basic MRI API
- Pointer Wrapper

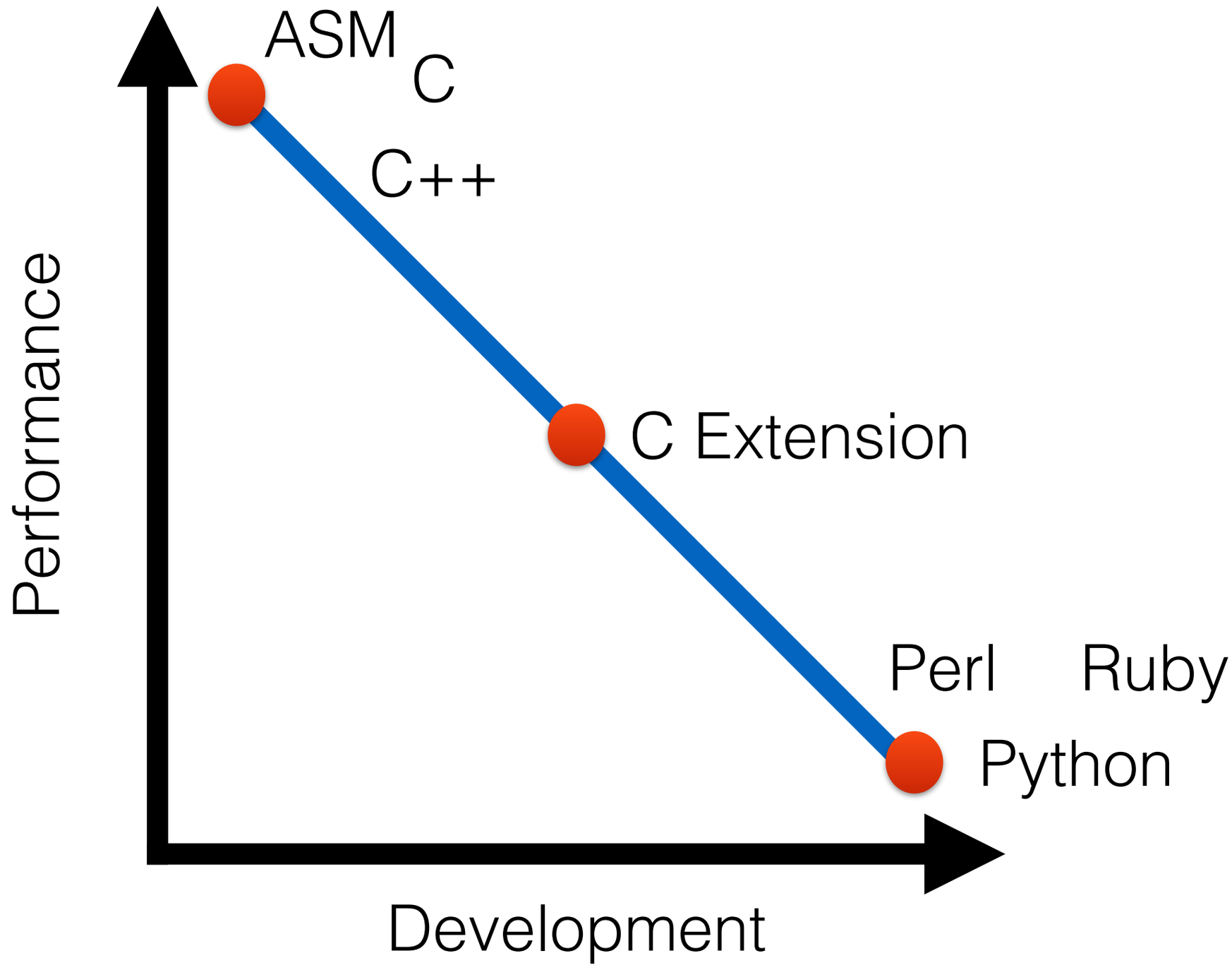
Why C Extension?

Beast from “Kung Fu Hustle”

“What a fast code!”

“You code fast!”

Code Performance
vs
Development Efficiency


ASM
C
C++
C Extension
Perl
Ruby
Python
Performance
Development

First step?

Profiling Tool

gem install ruby-prof

Jaro-winkler Distance
Pure Ruby Implementation

1.6918 ms

```
Total Time: 1.691791
Sort by: total_time

 %total   %self      total       self      wait     child             calls  Name
------------------------------------------------------------------------------
 100.00%  27.55%      1.692      0.466     0.000     1.226                 1  Global#[No method]
                      1.226      0.016     0.000     1.210               1/1  Integer#times
------------------------------------------------------------------------------
                      1.226      0.016     0.000     1.210               1/1  Global#[No method]
  72.45%   0.92%      1.226      0.016     0.000     1.210                 1  Integer#times
                      1.210      0.118     0.000     1.092     10000/10000  JaroWinkler#distance
------------------------------------------------------------------------------
                      1.210      0.118     0.000     1.092     10000/10000  Integer#times
  71.52%   6.98%      1.210      0.118     0.000     1.092           10000  JaroWinkler#distance
                      0.881      0.133     0.000     0.748     10000/10000  JaroWinkler#jaro_distance
                      0.092      0.013     0.000     0.079     10000/60000  Enumerable#each_with_index
                      0.042      0.014     0.000     0.028     10000/10000  Hash#merge
```

Replace with C Extension

0.5103 ms

```
Total Time: 0.5103410000000004
Sort by: total_time

 %total   %self      total       self      wait     child             calls   Name
------------------------------------------------------------------------------
 100.00%  95.59%      0.510      0.488     0.000     0.022                 1   Global#[No method]
                      0.022      0.013     0.000     0.009               1/1   Integer#times
------------------------------------------------------------------------------
                      0.022      0.013     0.000     0.009               1/1   Global#[No method]
   4.41%   2.61%      0.022      0.013     0.000     0.009                 1   Integer#times
                      0.009      0.009     0.000     0.000       10000/10000   JaroWinkler#distance
------------------------------------------------------------------------------
                      0.009      0.009     0.000     0.000       10000/10000   Integer#times
   1.80%   1.80%      0.009      0.009     0.000     0.000             10000   JaroWinkler#distance

* indicates recursively called methods
```

Ruby EXIF Readers

```
$ ruby exif_benchmark.rb
                    user     system      total        real
mini_exiftool   0.150000   0.050000  12.400000 ( 12.540122)
exifr           0.080000   0.000000   0.080000 (  0.083251)
exif            0.010000   0.000000   0.010000 (  0.009855)
```

- mini_exiftool - CLI wrapper of Exiftool
- exifr - Pure Ruby
- exif - C Extension of libexif

GitHub

tonytonyjan/jaro_winkler

tonytonyjan/exif

Make C Extension

Solutions

- C API of Ruby
- rubyinline - mixing C code into Ruby
- SWIG - Simplified Wrapper and Interface Generator

rubyinline

```ruby
require "inline"
class MyTest
  inline do |builder|
    builder.c "
    long factorial(int max) {
      int i=max, result=1;
      while (i >= 2) { result *= i--; }
      return result;
    }"
  end
end
t = MyTest.new()
factorial_5 = t.factorial(5)
```

SWIG

1 function

1 declaration

foo.c foo.h libfoo.i

$ swig -ruby libfoo.i

foo.c foo.h libfoo_wrap.c

2k lines

clang/gcc

libfoo.so

3x bigger than MRI C API implementation

SWIG is similar to C, but not C

```
/* libfoo.i */
%module libfoo
%{
  #include "libfoo.h"
%}

%typemap(in, numinputs=0) (double *tax, double *rate) {
  $1 = (double *)malloc(1 * sizeof(double));
  $2 = (double *)malloc(1 * sizeof(double));
};
```

MRI API is just Fine

What happen in “require”

```
require 'foo'
```

It will load “foo.rb”

foo.so, foo.o and foo.bundle

RbConfig::CONFIG['DLEXT']

To write a loadable Ruby module is easy.

What about write a native loadable module?

Entry of a C Program

```c
// main.c
#include <stdio.h>

int main(int argc, char const *argv[]){
  printf("Hello World");
  return 0;
}
```

Entry of C Extension

```c
#include <ruby.h>

int Init_foo(){
  printf("Hello World\n");
  return 0;
}
```

```
$ ruby -e 'require "foo"'
Hello World
```

Compilation

How to compile?

- Compile in-line
- autoconf/pkg-config
- MakeMakefile (recommended)

Compile in-line

```
clang \
-I/RUBY/PATH/include/ruby-2.2.0/PLATFORM_NAME \
-I/RUBY/PATH/include/ruby-2.2.0 \
-Wl,-undefined,dynamic_lookup -Wl,-multiply_defined,suppress \
-lpthread -lgmp -ldl -lobjc \
-o foo.bundle foo.c
```

pkg-config

```
export PKG_CONFIG_PATH=/RUBY/PATH/lib/pkgconfig
clang $(pkg-config --libs --cflags ruby-2.2) -o foo.bundle foo.c
```

mkmf (Make Makefile)

```ruby
# extconf.rb
require 'mkmf'
create_makefile('foo')
```

$ ruby extconf.rb
$ make

```
.
├── Makefile
├── extconf.rb
├── foo.bundle
├── foo.c
└── foo.o

0 directories, 5 files
```

installation path

```
create_makefile("foo_ext")
```

```
require 'foo_ext'
```

```
create_makefile("foo/bar")
```

```
require 'foo/bar'
```

custom source path

```
create_makefile("foo_ext" "src")
```

```
.
├── extconf.rb
└── src
    ├── bar.c
    ├── bar.h
    ├── foo.c
    └── foo.h
```

$ ruby extconf.rb
$ make

```
.
├── Makefile
├── bar.o
├── extconf.rb
├── foo.o
├── foo_ext.bundle
└── src
    ├── bar.c
    ├── bar.h
    ├── foo.c
    └── foo.h
```

Conditional Processing

Conditional Processing

```
AC_CHECK_HEADERS([foo.h])
AC_CHECK_HEADERS([bar.h], [], [],
[#ifdef HAVE_FOO_H
# include <foo.h>
#endif
])
```

AutoConfig - Generic Header Checks

Conditional Processing

mkmf methods	Compilor Optoins
have_header(“foo”)	-DHAVE_FOO_H
have_library(“foo”)	-lexif
have_type(“foo”)	-DHAVE_TYPE_FOO
have_var(“foo”)	-DHAVE_FOO
have_struct_member('struct foo', 'bar')	-DHAVE_STRUCT_FOO_BAR
have_func(“foo”)	-DHAVE_FOO

It’s just DSL

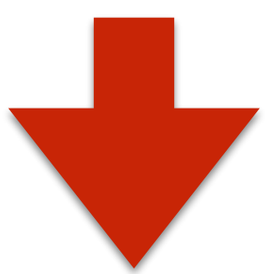
```
have_func 'PQconnectionUsedPassword' or
  abort "Your PostgreSQL is too old. Either install an older version " +
        "of this gem or upgrade your database."
have_func 'PQisthreadsafe'
have_func 'PQprepare'
have_func 'PQexecParams'
have_func 'PQescapeString'
```

Configurable Target Path

```
gem install mysql2 -- --with-mysql-dir=prefix
gem install mysql2 -- --with-mysql-include=lib --with-mysql-lib=dir
```

dir_config

```
dir_config('foo')
```



```
gem install foo -- --with-foo-dir=prefix
gem install foo -- --with-foo-include=lib --with-foo-lib=dir
```

Setup Gemspec

```ruby
# foo.gemspec
Gem::Specification.new 'foo', '1.0.0' do |s|
  s.name = 'foo'
  s.summary = 'foo'
  s.authors = %w[tonytonyjan]
  s.files = %w[foo.c]
  s.extensions = %w[extconf.rb]
end
```

That’s it

Yon’t need autotools,
mkmf gives you the best.

File Structure

```
$ bundle gem NAME --ext
```

Typical File Structure

```
.
├── Gemfile
├── Rakefile
├── ext
│   └── foo
│       ├── extconf.rb
│       ├── foo.c
│       └── foo.h
├── foo.gemspec
└── lib
    ├── foo
    │   └── version.rb
    └── foo.rb

4 directories, 8 files
```

```ruby
# ext/extconf.rb
require "mkmf"
create_makefile("foo/foo")
```

```ruby
# lib/foo.rb
require "foo/version"
require "foo/foo"
```

mysql2, nokogiri, sqlite3

```
require 'foo/foo'
```

It's ambiguous!

Better File Structure

```
.
├── Gemfile
├── Rakefile
├── ext
│   ├── extconf.rb
│   ├── foo.c
│   └── foo.h
├── foo.gemspec
└── lib
    ├── foo
    │   └── version.rb
    └── foo.rb

3 directories, 8 files
```

```
# ext/extconf.rb
require "mkmf"
create_makefile("foo_ext")
```

```
# lib/foo.rb
require "foo/version"
require "foo_ext"
```

pg, bcrypt, eventmachine

Development Workflow

`gem install` will generate Makefile & build automatically.

While developing...

```
$ cd ext/
$ ruby extconf.rb
$ make
$ cd ..
$ ruby -I ext -r foo_ext -e '...'
```

It's too tedious.

Life can be easier.

gem install rake-compiler

```
# Rakefile
require "rake/extensiontask"


Rake::ExtensionTask.new("foo_ext") do |ext|
  ext.ext_dir = "ext"
end
```

$ rake -D

```
rake clean
    Remove any temporary products.

rake clobber
    Remove any generated file.

rake compile
    Compile all the extensions

rake compile:foo_ext
    Compile foo_ext
```

Using rake-compiler

$ rake compile test

rake-compiler is for development.

It’s nothing to do with gem installation.

Basic C API

Key Knowledge

- Ruby is OO, C is not.
- C variables have types but data don't.
- Ruby variables have no types but data do.
- Data in Ruby are represented by C type "VALUE", and "VALUE" data has its own data-type.

Define Module/Class

Ruby

```ruby
module Foo; end
class Bar; end
```

```ruby
Bar.superclass #=> Object
```

C

```c
void
Init_foo_ext(void)
{
  VALUE rb_mFoo = rb_define_module("Foo");
  VALUE rb_cBar = rb_define_class("Bar", rb_cObject);
}
```

Nested Class/Module

Ruby

```ruby
module Foo
  class Bar; end
  module Buz; end
end
```

C

```c
void Init_foo_ext(void){
  VALUE rb_mFoo = rb_define_module("Foo");
  VALUE rb_cBar = rb_define_class_under(rb_mFoo, "Bar", rb_cObject);
  VALUE rb_mBuz = rb_define_module_under(rb_mFoo, "Buz");
}
```

```c
VALUE rb_define_class(const char *name, VALUE super);
VALUE rb_define_module(const char *name);
VALUE rb_define_class_under(VALUE outer, const char *name, VALUE super);
VALUE rb_define_module_under(VALUE outer, const char *name);
```

Define Method

Ruby

```ruby
class Bar
  def self.hello(name); puts "Hello, #{name}" end
  def world; puts "World" end
end
```

C

```c
rb_define_singleton_method(rb_cBar, "hello", hello, 1);
rb_define_method(rb_cBar, "world", helllo, 0);
```

function pointer

argc

self

```c
VALUE hello(VALUE instance, VALUE name){
  printf("hello %s\n", StringValueCStr(name));
  return Qnil;
}
```

Convert Ruby String to C String

Singleton Instance of rb_cNilClass
(there are also Qfalse, Qtrue)

```c
VALUE world(VALUE klass){
  printf("world\n");
  return Qnil;
}
```

self

Ruby data <-> C data

	Fixnum	Numeric	String
int	FIX2INT(value) INT2FIX(i)	NUM2INT(value) INT2NUM(I)	
long	FIX2LONG(value) LONG2FIX(l)	NUM2LONG(value) LONG2NUM(l)	
double		NUM2DBL(value) rb_float_new(f)	
char*			StringValueCStr(value) rb_str_new_cstr(s)

Steps of implementation

```
VALUE func(VALUE self, VALUE argv...){
  // 1. Check types of arguments
  // 2. convert arguments to C data
  // 3. process C data
  // 4. return VALUE object
}
```

Type Checking

Ruby

```ruby
case obj
when Modeul then # ...
when Class then # ...
else raise 'not valid'
end
```

C

```c
switch (TYPE(rb_mFoo)){
  case T_MODULE:
    // process Module
    break;
  case T_CLASS:
    // process Class
    break;
  default:
    rb_raise(rb_eTypeError, "not valid value");
    break;
}
```

internal VALUE Types

T_NIL	T_STRING	T_STRUCT	T_FILE
T_OBJECT	T_REGEXP	T_BIGNUM	T_TRUE
T_CLASS	T_ARRAY	T_FIXNUM	T_FALSE
T_MODULE	T_HASH	T_COMPLEX	T_DATA
T_FLOAT	T_SYMBOL	T_RATIONAL	

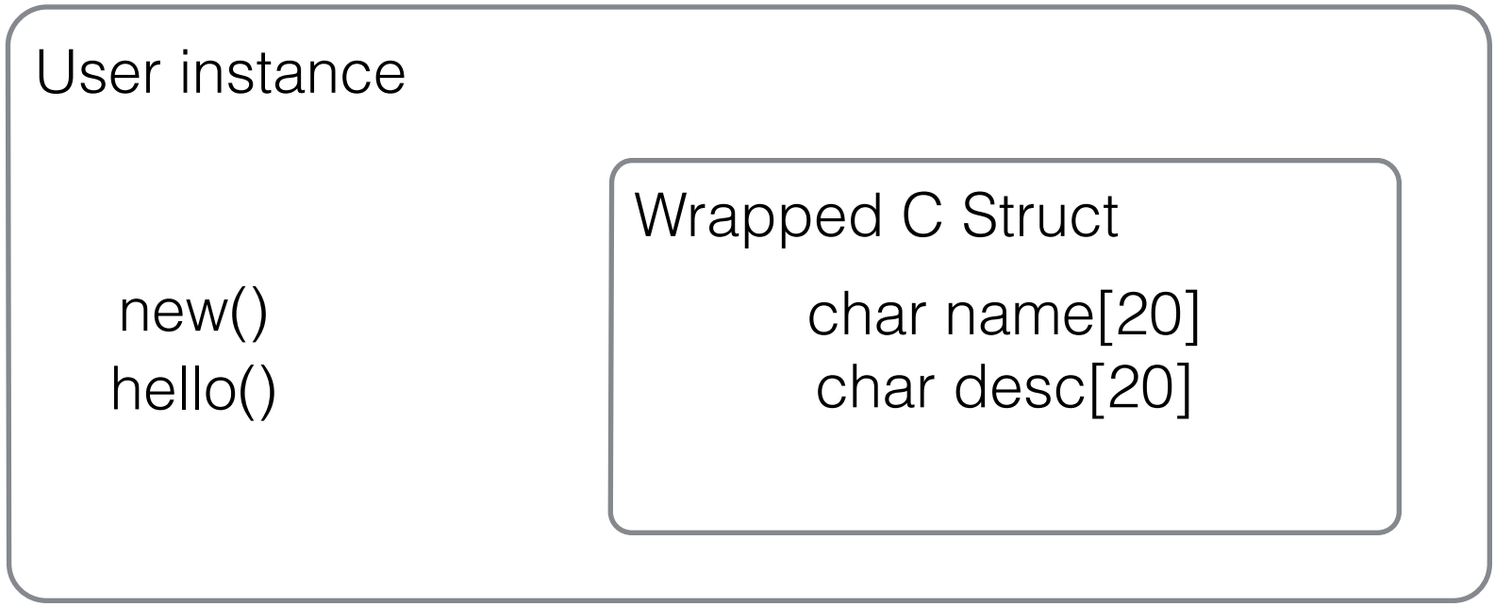
Pointer Wrapper

C is not OOP

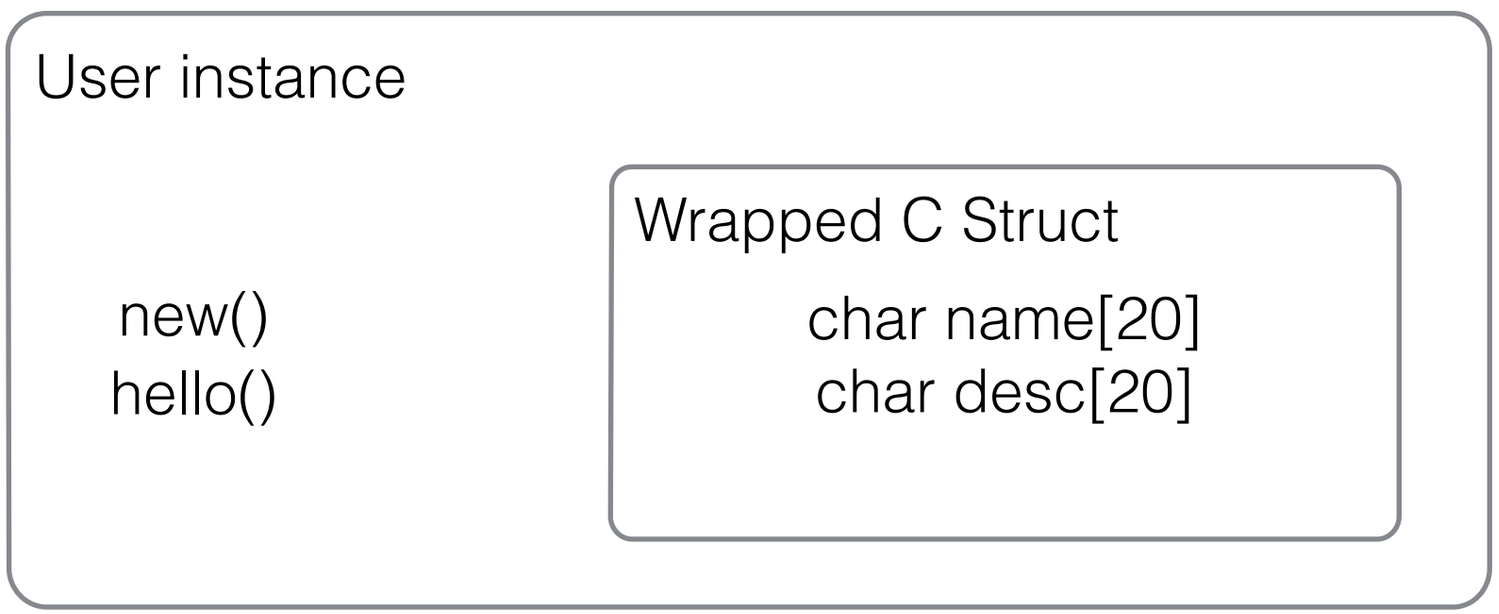
However, there is structure in C.

```ruby
u = User.new("Weihang", "An engineer")
u.hello # => print: "Hi, I'm Weihang"
```

u1 = User.new(…)



u2 = User.new(…)



```c
// user.h
typedef struct {
    char name[20];
    char desc[20];
} User;
```

```c
static VALUE new(VALUE self, VALUE name, VALUE desc);
static VALUE hello(VALUE self);
```

```c
VALUE rb_cBar = rb_define_class_under(rb_mFoo, "Bar", rb_cObject);
rb_define_singleton_method(rb_cBar, "new", new, 2);
rb_define_method(rb_cBar, "hello", hello, 0);
```

User#new(name, desc)

Ruby class C struct return

```
static VALUE new(VALUE self, VALUE name, VALUE desc){
  User* user;
  VALUE rb_user = Data_Make_Struct(self, User, 0, -1, user);
  strncpy(user->name, StringValueCStr(name), 19);
  strncpy(user->desc, StringValueCStr(desc), 19);
  return rb_user;
}
```

mark function pointer

ActiveRecord::Persistence#destroy free function pointer

User#hello

Ruby instance C struct return

```c
static VALUE hello(VALUE self){
  User* user;
  Data_Get_Struct(self, User, user);
  printf("Hi, I'm %s\n", user->name);
  return Qnil;
}
```

What about C++?

```cpp
#include <iostream>
#include "foo.h"
using namespace std;

extern "C" void Init_foo(){
    cout << "Hello, World" << endl;
}
```

That’s it!

Leaning Resources

- Official Guide
  - http://guides.rubygems.org/gems-with-extensions/
- README.EXT in ruby source code
- @tenderlove
  - http://tenderlovemaking.com/2009/12/18/writing-ruby-c-extensions-part-1.html
  - http://tenderlovemaking.com/2010/12/11/writing-ruby-c-extensions-part-2.html
- Source code of pg, sqlite2, mysql2, nokogiri, kgio, bcrypt, etc
- MRI source code

Thanks for your listening