Ruby - Bug #12427

Defining methods with the same name to both Fixnum and Bignum classes could cause SEGV in C extensions since Feature #12005

05/25/2016 07:23 AM - frsyuki (Sadayuki Furuhashi)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.0dev (2016-05-21 trunk 55091) [x86_64-darwin14]	**Backport:**	2.1: DONTNEED, 2.2: DONTNEED, 2.3: DONTNEED

Description

My gem (msgpack.gem) includes C extension with following pseudo code.
This code is working well with released ruby versions. But it caused SEGV with ruby 2.4.0-dev trunk 55091.

```
static VALUE Fixnum_to_msgpack(int argc, VALUE* argv, VALUE self)
{
    long v = FIX2LONG(self);
    printf("%ld", v);  // work with v
}

static VALUE Bignum_to_msgpack(int argc, VALUE* argv, VALUE self)
{
    if (RBIGNUM_POSITIVE_P(self)) {
        uint64_t positive = rb_big2ull(self);
        printf("%llu", positive);  // work with positive
    } else {
        int64_t negative = rb_big2ll(self);
        printf("%lld", negative);  // work with negative
    }
}

void Init_msgpack(void)
{
    rb_define_method(rb_cFixnum, "to_msgpack", Fixnum_to_msgpack, -1);
    rb_define_method(rb_cBignum, "to_msgpack", Bignum_to_msgpack, -1);
}
```

Apparently, since Feature #12005 (https://bugs.ruby-lang.org/issues/12005), both rb_cFixnum and rb_cBignum are reference to rb_cInteger (rb_cFixnum == rb_cBignum). Therefore, the later rb_define_method call on rb_cBignum overwrites the method with same name on rb_cFixnum. So if to_msgpack method is called against an integer in the range of fixnum, Bignum_to_msgpack function is called against a fixnum object. Then, RBIGNUM_POSITIVE_P is called against a fixnum object. However, because RBIGNUM_POSITIVE_P expects bignum object strictly despite of underlaying unified class definition with fixnum, it causes SEGV.
This issue happens differently if the order of rb_define_method against rb_cFixnum and rb_cBignum is opposite.

This test code reproduces the problem:
https://github.com/msgpack/msgpack-ruby/pull/115/commits/26183b718963f882309d52c167dc866ba5260272

I added following fix to the C extension. It succeeded to avoid the problem:
https://github.com/msgpack/msgpack-ruby/pull/115/files

Concern 1 is that this issue seems a common problem with C extensions and hard to debug unless the author is aware of changes of Feature #12005 including its influence on C API.
Concern 2 is that C extensions want to use a macro instead of runtime if (rb_cFixnum == rb_cBignum) check to branch code at compile time.
I hope that documents or release note of ruby 2.4 includes mention for those concerns and guidelines for their solution.

Related issues:

Related to Ruby - Feature #12005: Unify Fixnum and Bignum into Integer	**Closed**

Associated revisions

Revision c071c052292168592de091cdb4c22dc3fbfd1af1 - 06/13/2016 11:34 AM - nobu (Nobuyoshi Nakada)

Integer unification macro

- include/ruby/ruby.h (RUBY_INTEGER_UNIFICATION): macro to tell if
  Integer is integrated. [ruby-core:75718][Bug #12427]
- include/ruby/backward.h, internal.h (rb_cFixnum, rb_cBignum):
  fallback to rb_cInteger.
- bignum.c, numeric.c, ext/json/generator/generator.{c,h}: use the
  macro.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55394 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c071c052 - 06/13/2016 11:34 AM - nobu (Nobuyoshi Nakada)

Integer unification macro

- include/ruby/ruby.h (RUBY_INTEGER_UNIFICATION): macro to tell if
  Integer is integrated. [ruby-core:75718][Bug #12427]
- include/ruby/backward.h, internal.h (rb_cFixnum, rb_cBignum):
  fallback to rb_cInteger.
- bignum.c, numeric.c, ext/json/generator/generator.{c,h}: use the
  macro.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55394 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 5bddb944926aadea6f4a7cda25a0053593153652 - 06/14/2016 02:43 PM - nobu (Nobuyoshi Nakada)

remove backward macros

- include/ruby/backward.h (rb_cFixnum, rb_cBignum): remove the
  backward compatibility macros, to fail incompatible extension
  libraries earily. [Bug #12427]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55413 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 5bddb944 - 06/14/2016 02:43 PM - nobu (Nobuyoshi Nakada)

remove backward macros

- include/ruby/backward.h (rb_cFixnum, rb_cBignum): remove the
  backward compatibility macros, to fail incompatible extension
  libraries earily. [Bug #12427]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@55413 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 05/25/2016 08:30 AM - nobu (Nobuyoshi Nakada)

- Description updated

- Status changed from Open to Feedback

https://github.com/ruby/ruby/compare/trunk...nobu:bug/12427-integer-integration
This patch does:

> Concern 1 is that this issue seems a common problem with C extensions and hard to debug unless the author is aware of changes of Feature #12005 including its influence on C API.

Show warnings if rb_cFixnum or rb_cBignum is used,

> Concern 2 is that C extensions want to use a macro instead of runtime if (rb_cFixnum == rb_cBignum) check to branch code at compile time.

Add a macro RUBY_INTEGER_INTEGRATION.

#2 - 06/13/2016 11:35 AM - nobu (Nobuyoshi Nakada)

- Status changed from Feedback to Closed

Applied in changeset r55394.

Integer unification macro

- include/ruby/ruby.h (RUBY_INTEGER_UNIFICATION): macro to tell if Integer is integrated. [ruby-core:75718][Bug #12427]
- include/ruby/backward.h, internal.h (rb_cFixnum, rb_cBignum): fallback to rb_cInteger.
- bignum.c, numeric.c, ext/json/generator/generator.{c,h}: use the macro.

#3 - 06/15/2016 04:44 AM - naruse (Yui NARUSE)

- Related to Feature #12005: Unify Fixnum and Bignum into Integer added

#4 - 08/04/2016 06:08 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN to 2.1: DONTNEED, 2.2: DONTNEED, 2.3: DONTNEED