Ruby - Feature #6083

Hide a Bignum definition

02/25/2012 02:02 PM - ko1 (Koichi Sasada)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	3.0	

Description

Now, the struct RBignum which is a definition of Bignum in C is located in include/ruby/ruby.h. It means we can't change implementation of Bignum. For example, using GMP as Bignum representation.

I propose to move the struct RBignum definition from include/ruby/ruby.h to bignum.c. I believe no one use struct RBignum directly (except core).

It has possibility to break binary compatibility.

Associated revisions

Revision ab9bc151e828843b9d9170754f4ba4d92ef85750 - 02/14/2014 03:29 PM - akr (Akira Tanaka)

- include/ruby/ruby.h,
  internal.h,
  ext/-test-/bignum/bigzero.c: Hide a Bignum definition.
  [ruby-core:42891] [Feature #6083]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@44957 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision ab9bc151 - 02/14/2014 03:29 PM - akr (Akira Tanaka)

- include/ruby/ruby.h,
  internal.h,
  ext/-test-/bignum/bigzero.c: Hide a Bignum definition.
  [ruby-core:42891] [Feature #6083]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@44957 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 02/25/2012 05:17 PM - naruse (Yui NARUSE)

Binary hackers can handle C structs even if it is a hidden private struct.
So it should be enough simply to declare "struct RBignunm is not a public API".

#2 - 02/25/2012 06:23 PM - kosaki (Motohiro KOSAKI)

> Binary hackers can handle C structs even if it is a hidden private struct.
> So it should be enough simply to declare "struct RBignunm is not a public API".

I disagree. As far as we consider there is backdoor, every binary
compatibility effort are
not meaningful. But I believe it clearly has a worth.

#3 - 03/30/2012 01:11 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

ko1 wrote:

> I propose to move the struct RBignum definition from include/ruby/ruby.h to bignum.c. I believe no one use struct RBignum directly (except core).

Agreed.
I hope C API would be organized. This can be preliminary case.

naruse wrote:

> Binary hackers can handle C structs even if it is a hidden private struct.
> So it should be enough simply to declare "struct RBignunm is not a public API".

If you think so, it could be simpler.
README.EXT has no description about RBignum.
So we don't *need* to do anything.
But we can *kindly* do something.

--
Yusuke Endoh mame@tsg.ne.jp

#4 - 07/25/2012 07:08 PM - ko1 (Koichi Sasada)

Matz, what do you think about it?

#5 - 10/31/2012 02:06 AM - matz (Yukihiro Matsumoto)

- Target version changed from 2.0.0 to 2.6

I am sorry but it's too late. I admit my fault.

Matz.

#6 - 10/31/2012 03:53 PM - kosaki (Motohiro KOSAKI)

> Target version changed from 2.0.0 to next minor
> I am sorry but it's too late. I admit my fault.

Matz, I don't like this decision. I believe we should choose next *major*
or 2.0.0. minor release should not have binary incompatibility as far
as possible.
Is this typo? If no, I'd like to propose pull it back. declaration
moving is not big task.

- kosaki

#7 - 10/31/2012 10:36 PM - ko1 (Koichi Sasada)

(2012/10/31 0:31), KOSAKI Motohiro wrote:

> > Target version changed from 2.0.0 to next minor
> > I am sorry but it's too late. I admit my fault.
>
> Matz, I don't like this decision. I believe we should choose next *major*
> or 2.0.0. minor release should not have binary incompatibility as far
> as possible.
> Is this typo? If no, I'd like to propose pull it back. declaration
> moving is not big task.

Maybe, we don't define what is major and what is minor.

On 2.0.x, we shouldn't change binary compatibility.
But 2.1.x or later, I think it is acceptable.
(and some people don't accept :))

--
// SASADA Koichi at atdot dot net

#8 - 11/01/2012 03:53 AM - kosaki (Motohiro KOSAKI)

On Wed, Oct 31, 2012 at 9:26 AM, SASADA Koichi ko1@atdot.net wrote:

> (2012/10/31 0:31), KOSAKI Motohiro wrote:

> Target version changed from 2.0.0 to next minor
> I am sorry but it's too late. I admit my fault.

Matz, I don't like this decision. I believe we should choose next *major*
or 2.0.0. minor release should not have binary incompatibility as far
as possible.
Is this typo? If no, I'd like to propose pull it back. declaration
moving is not big task.

Maybe, we don't define what is major and what is minor.

On 2.0.x, we shouldn't change binary compatibility.
But 2.1.x or later, I think it is acceptable.
(and some people don't accept :))

Ok, please ignore my last mail then. :)

#9 - 11/01/2012 11:53 PM - Anonymous

Hi,

In message "Re: [ruby-core:48640] Re: [ruby-trunk - Feature #6083] Hide a Bignum definition"
on Wed, 31 Oct 2012 15:31:08 +0900, KOSAKI Motohiro kosaki.motohiro@gmail.com writes:

|> Target version changed from 2.0.0 to next minor
|> I am sorry but it's too late. I admit my fault.
|
|Matz, I don't like this decision. I believe we should choose next *major*
|or 2.0.0. minor release should not have binary incompatibility as far
|as possible.
|Is this typo? If no, I'd like to propose pull it back. declaration
|moving is not big task.

Hiding (or no hiding) Bignum definition would not change binary
compatibility, probably you mean source compatibility?

But keeping compatibility during the release cycle is important, I
admit. So I changed my mind. If there's no big opposition, I can
accept this proposal.

```
matz.
```

#10 - 11/01/2012 11:54 PM - matz (Yukihiro Matsumoto)

- Target version changed from 2.6 to 2.0.0

#11 - 11/02/2012 01:33 AM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to 3.0

> If there's no big opposition, I can accept this proposal.

Sorry, but I must voice big opposition.

Unfortunately, hiding RBignum will cause significant incompatibility.
The macro RBIGNUM_BDIGITS uses RBignum internally. Many extension
libraries actually uses RBIGNUM_BDIGITS to construct a bignum object.

So, to hide RBignum, we must carefully design a new alternative API
to construct a bignum. It will take a time.
Worse, source compatibility will break anyway because it is almost
impossible to make it the same as the current RBIGNUM_BDIGITS.

(Consider replacing the internal representation with GMP.
RBIGNUM_BDIGITS must malloc a BDIGIT array and export GMP into the
array. But no one will free the allocated memory.)

Thus, I'm moving this to "Next Major". Again, I'm sorry.

--
Yusuke Endoh mame@tsg.ne.jp

#12 - 11/02/2012 02:05 AM - shyouhei (Shyouhei Urabe)

+1 for next major.

No one is against this basic concept of hiding Bignums, meseems. It's just a bad timing for us.

#13 - 11/02/2012 02:53 AM - ko1 (Koichi Sasada)

(2012/11/01 11:05), shyouhei (Shyouhei Urabe) wrote:

> +1 for next major.

+1 for next major or next minor.

We need discussion about how to break binary compatibility in another
thread :)

--
// SASADA Koichi at atdot dot net

#14 - 11/02/2012 03:23 AM - mame (Yusuke Endoh)

2012/11/2 SASADA Koichi ko1@atdot.net:

> (2012/11/01 11:05), shyouhei (Shyouhei Urabe) wrote:
>
> > +1 for next major.
>
> +1 for next major or next minor.
>
> We need discussion about how to break binary compatibility in another
> thread :)

Note that this issue (#6083) will break source compatibility.

--
Yusuke Endoh mame@tsg.ne.jp

#15 - 02/12/2014 11:48 AM - akr (Akira Tanaka)

- File hide-bignum-internal.patch added

I made a patch for this issue.

It just moves struct RBignum and related macros from include/ruby/ruby.h to internal.h, not bignum.c.
It is because other files (such as gc.c, marshal.c, etc.) access internal of struct RBignum.

#16 - 02/14/2014 07:59 AM - matz (Yukihiro Matsumoto)

Even though this breaks source compatibility, it is worth changing it (to mark dangerous operation).
So go ahead.

Matz.

#17 - 02/14/2014 03:29 PM - akr (Akira Tanaka)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

Applied in changeset r44957.

- include/ruby/ruby.h,
  internal.h,
  ext/-test-/bignum/bigzero.c: Hide a Bignum definition.
  [ruby-core:42891] [Feature #6083]

#18 - 02/25/2014 12:01 AM - tenderlovemaking (Aaron Patterson)

Hi,

r44957 breaks the sqlite3 gem. I guess it uses the RBIGNUM_LEN macro:

https://github.com/sparklemotion/sqlite3-ruby/blob/2070182f461f1e2d76b9d40aa45fed2d04e9a4d6/ext/sqlite3/statement.c#L271-274

What should we use as a replacement? @naruse (Yui NARUSE), I think you added this check to the sqlite3 gem:

https://github.com/sparklemotion/sqlite3-ruby/commit/7dbd82d3

Thanks!

#19 - 02/25/2014 12:42 AM - akr (Akira Tanaka)

nobu wrote a patch: https://github.com/nobu/sqlite3-ruby/compare/bignum-conversion?expand=1

I'm not sure the status to merge it to the upstream.
(I think the patch should be similified by removing code for nails != 0.)

Note that the original condition, RBIGNUM_LEN(result) * SIZEOF_BDIGITS <= 8, is wrong.
It tests -2**64 < x < 2**64 but it should test -2**63 <= x < 2**63 which accepts NUM2LL.

#20 - 02/25/2014 01:40 AM - tenderlovemaking (Aaron Patterson)

Ah, thanks! I think I can get it merged upstream.

#21 - 02/25/2014 09:59 AM - nobu (Nobuyoshi Nakada)

I've forgotten to push the latest commit, which removes unused code for nails != 0.

Files

hide-bignum-internal.patch	4.66 KB	02/12/2014	akr (Akira Tanaka)