Ruby master - Misc #16750

Change typedef of VALUE for better type checking

04/01/2020 06:22 PM - Dan0042 (Daniel DeLorme)

Status:	Open	
Priority:	Normal	
Assignee:		

Description

VALUE is currently defined as typedef unsigned long VALUE, but as we all know that only creates an alias, not an actual type. Since the compiler gives no warnings when comparing with other integer values, it's easy to have bugs such as v == 42 which should have been v == INT2FIX(42). Actually not so long ago I saw nobu fixing a bug of that kind where an ID had been mixed up with a VALUE.

So in order to prevent these kinds of bugs I propose changing VALUE to a non-scalar type such as:

```
//example for 64-bit system
typedef union VALUE {
    struct RBasic* as_basic; //easy access to obj.as_basic->klass and obj.as_basic->flags
    void* as_ptr;            //can assign ptr = obj.as_ptr without a cast
    unsigned long i;         //raw int value for bitwise operations

    int immediate : 3;
//obj.immediate != 0 if obj is immediate type such as fixnum, flonum, static symbol

    int is_fixnum : 1;   //obj.is_fixnum == 1 if obj is fixnum

    struct {
        int flag : 2;    //obj.flonum.flag == 2 if obj is flonum
        int bits : 62;
    } flonum;

    struct {
        int flag : 8;    //obj.symbol.flag == 0x0c if obj is a static symbol
        int id : 56;     //-> obj.symbol.id == STATIC_SYM2ID(obj)
    } symbol;

} VALUE;
```

This will allow proper type-checking via the compiler.

A little-known fact, structs and unions can be passed (and returned) by value to functions; this 64-bit union has the same performance as a 64-bit int. This approach also allows to simplify code like ((struct RBasic*)obj)->flags into the much more readable obj.as_basic->flags, and FIXNUM_P(obj) can be expressed directly as obj.is_fixnum. The only downside is that direct comparison of union variables is not possible, so you would need to use for example obj.i == Qfalse.i

To summarize, stricter type-checking would eliminate an entire class of bugs, and I estimate this change would require modifications to **no more than 14%** of the codebase (62,213 lines). Very much worth it I believe.

History

#1 - 04/01/2020 06:54 PM - jeremyevans0 (Jeremy Evans)

Affecting 62,213 lines is huge. That's at least 62,213 places where an error could be introduced by using the incorrect union member or something similar.

If we were going to consider this, it would be helpful to have a list of previous commits that fixed bugs caused by the current typedef of VALUE? That would make it easier to assess the practical benefits of this proposal.

I think this change would require breaking compatibility with almost all existing C extensions, and I'm sure the benefits of this proposal do not exceed the costs of dropping C extension compatibility.

#2 - 04/01/2020 10:54 PM - Eregon (Benoit Daloze)

I think the motivation to make C extensions safer is great here, but I see many problems with this approach.

TruffleRuby until recently typed VALUE (and ID) as void* (because TruffleRuby+Sulong store managed Java objects in there, and that used to require

to be a pointer type, but no longer).
This caused various incompatibilities, most notably VALUE var; switch (var) { ... failing to compile (the C compiler requires it to be an integer type to switch on it) and that's used in many gems.
Bitwise operations on VALUE are also done in multiple gems to get a "hash code" of the VALUE (e.g., openssl, pg).
So 2 weeks ago TruffleRuby changed to define VALUE as unsigned long just like MRI, which fixed all these incompatibilities:
https://twitter.com/TruffleRuby/status/1240688301276684288

So I'm negative on changing that for compatibility, and I can say from experience it would break many gems.
OTOH, I agree with your point that a better type such as void* or the union would find some incorrect usages.

There are actually multiple unexpected usages like that in MRI, including things like integer flags being typed as VALUE.
I think those would be nice to fix regardless (but not very big issues either).

We found several bugs by having VALUE as void*, and tried to report them to the respective gems (VALUE used as an int variable, malloc(VALUE), arithmetic done on VALUE, etc).

I think union is a very dangerous construct (in general, it's very unsafe in C) and in this case it would be so easy to misuse (e.g., value.as_basic->klass on a Fixnum, OTOH RBasic(value) can check if meaningful).
VALUE should remain an opaque pointer to let more freedom to the implementation.
CRuby doesn't expose struct members in include in recent Ruby versions, and that's a good thing (e.g. never a good idea to dig in struct RString or RArray in a C extension).

I think something that could be interesting is having a preprocessor flag to type VALUE as void*.
This might help C extension developers to find incorrect usages of VALUE (but at the cost of e.g. not allowing switch (VALUE)).
Such an approach doesn't seem feasible for the union, because it would require to change lots of code, while swapping unsigned long and void* is not much diff at least in include files:
https://github.com/oracle/truffleruby/commit/0552ac72e3396492dc42ea53c1ac5ee2eb24a3ba#diff-ceeab34971467098f88189501912c870
And also some changes in https://github.com/oracle/truffleruby/commit/4e127160f92fbccdcdd53981dc05d79800199b9d

#3 - 04/02/2020 12:35 AM - shyouhei (Shyouhei Urabe)

FWIW ruby internally has struct RVALUE inside of gc.c, which is something similar to what is proposed in this ticket. It is at lest intentional to avoid the approach. The intention itself is not necessarily clear to me, though.

#4 - 04/02/2020 04:01 AM - matz (Yukihiro Matsumoto)

The current opaque approach has several reasons:

- portability
  The bit layout of bit fields in struct is somewhat implementation dependant, especially mixed with endianness. The union definition in OP may not work on big-endian CPUs.

- performance
  The bit field operations may be far slower than simple bit operations on some compiler.

I don't think it's worth it, especially for portability's sake.

Matz.

#5 - 04/02/2020 12:03 PM - Dan0042 (Daniel DeLorme)

- Tracker changed from Feature to Misc

Looks like I'm not able to add tags so... April fools :-)

#6 - 04/02/2020 01:33 PM - nobu (Nobuyoshi Nakada)

- Assignee set to 10790

No tags in Redmine now.

#7 - 04/03/2020 02:34 AM - hsbt (Hiroshi SHIBATA)

- Tags set to joke