rust-lang / **rust**

Tracking issue for 128-bit integer support (RFC 1504) #35118

New issue

Open **nikomatsakis** opened this issue on 29 Jul 2016 · 119 comments

nikomatsakis commented on 29 Jul 2016 • edited by nagisa Contributor

Tracking issue for rust-lang/rfcs#1504.

cc **@Amanieu**

Blocking stabilization:

- [x] #41799 (Casting u128::MAX to f32 is undefined)
- [x] ~~Interaction with FFI? (#35118 (comment))~~ - #44261
- [x] ~~separately feature gate `repr(i128)`~~ - #44262
- [x] #45676 (u/i)128 lowering for backends without native support
  - lowering still does not work for emscripten even with the work around
- [ ] Enums with 128-bit discriminant: `repr128` feature

👍 23

nikomatsakis added **T-lang** **B-unstable** **B-RFC-approved** labels on 29 Jul 2016

nikomatsakis referenced this issue in **rust-lang/rfcs** on 29 Jul 2016

Add support for 128-bit integers #1504 Merged

durka commented on 2 Aug 2016 • edited Contributor

Is `#[repr(u128)] enum SuchWideVeryDiscriminantWow { ... }` allowed?

👍 12 😄 13

Amanieu commented on 2 Aug 2016 Contributor

That's a good point, I don't see any reason why it shouldn't be allowed.

durka commented on 2 Aug 2016 Contributor

The return type of the `discriminant_value` intrinsic needs to be updated, then :)

nagisa commented on 2 Aug 2016 Contributor

Intrinsics are stuff internal to the compiler, therefore changes to them doesn't need discussion in the RFCs.

durka commented on 2 Aug 2016 Contributor

I know, I just wanted to mention it here since I didn't see enums discussed in the RFC.

japaric referenced this issue on 14 Aug 2016

Provide support for arbitrary width atomics from 16bit to 2x word size (u16, u32, u64, u128) #24564 Closed

Assignees

No one assigned

Labels

B-RFC-approved
B-unstable
C-tracking-issue
T-lang
T-libs
disposition-merge
finished-final-comment-period

Projects

None yet

Milestone

No milestone

Notifications

34 participants

and others

nikomatsakis referenced this issue in **rust-lang/rfcs** on 23 Aug 2016

Add i128 and u128 types #521 — Closed

nagisa referenced this issue on 24 Aug 2016

Initial implementation of the 128-bit integers #35954 — Closed

est31 referenced this issue on 20 Nov 2016

i128 and u128 support #37900 — Closed

2 of 2 tasks complete

bors added a commit that referenced this issue on 4 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `e618fe0`

bors added a commit that referenced this issue on 4 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `229be52`

bors added a commit that referenced this issue on 4 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `cfba287`

bors added a commit that referenced this issue on 4 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `854bf23`

bors added a commit that referenced this issue on 8 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `d9738bb`

bors added a commit that referenced this issue on 8 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `03fc80e`

bors added a commit that referenced this issue on 8 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `ab66ecc`

bors added a commit that referenced this issue on 14 Dec 2016

`Auto merge of #37900 - est31:i128, r=eddyb` … ✗ `255dba5`

est31 referenced this issue on 20 Dec 2016

i128 and u128 support #38482 — Merged

bors added a commit that referenced this issue on 20 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `764d2f6`

bors added a commit that referenced this issue on 21 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `1602354`

cmr commented on 21 Dec 2016 — Contributor

How should FFI with this type be handled? Is there any standard ABI support for these types? AFAICT, the answer is "no", which means this type should be FFI-unsafe, and the FFI unsafety lint should be updated to reject `Ty{I,Ui}nt` with 128-bit sizes.

cmr commented on 21 Dec 2016 — Contributor cc **@est31**

retep998 commented on 21 Dec 2016 • edited Member

On Windows there is most definitely no standard i128 yet because the standard compiler (msvc) does not support `__int128` yet on x86. There are some really good guesses though based on the MSDN documentation.

est31 commented on 21 Dec 2016 Contributor

> Is there any standard ABI support for these types?

It depends on the architecture. SysV defines the ABI for 64 bit architectures. For x86, its most likely its not defined. Same goes for windows: it should in theory be defined for 64 bit (just sadly nobody adheres to it, its a gigantic mess), but not on x86.

This directly maps the support of the i128 type in C, and I think generally it makes little sense to have FFI for types that don't exist in C.

est31 commented on 21 Dec 2016 Contributor

bug report in llvm about the x86_64 ABI problems: https://llvm.org/bugs/show_bug.cgi?id=31362

nagisa commented on 21 Dec 2016 Contributor

Also one in GCC: https://gcc.gnu.org/bugzilla/show_bug.cgi?id=78799

bors added a commit that referenced this issue on 21 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `517d2bd`

bors added a commit that referenced this issue on 30 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `7bfa313`

bors added a commit that referenced this issue on 30 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `04d4b1b`

bors added a commit that referenced this issue on 31 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `ab2adfd`

bors added a commit that referenced this issue on 31 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✗ `88f2ea6`

bors added a commit that referenced this issue on 31 Dec 2016

`Auto merge of #38482 - est31:i128, r=eddyb` … ✓ `38bd207`

This was referenced on 1 Jan 2017

[i128] C FFI mismatch on GNU/Linux with gcc #38762 Open

[i128] ICE when calling function with #[repr(C)]'d tuple struct #38763 Closed

robinst referenced this issue in **rust-lang/rfcs** on 4 Jan 2017

Add as_millis function to std::time::Duration #1545 Open

est31 commented on 5 Jan 2017 Contributor cc #38824

116 hidden items
Load more...

withoutboats commented on 2 Feb Contributor

Alex says that 128s work fine on `wasm-unknown-unknown` & that there is a test for this, the only problem was with emscripten.

To be clear, the other two platforms are nvidia GPUs (NVPTX) and a 16bit chip (atmel), both tier 3 platforms. Both of these platforms are extremely unusual platforms, and I suspect there are platform compatibility problems with many libraries *already* (e.g. assuming that usize is at least 32 bits). I think it would be great to support 128bit integers on these platforms, but I do not think we should block stabilizing the feature on this.

👍 7

plietar referenced this issue in **librespot-org/librespot** on 8 Feb

API review for librespot-core #130 Open

23 of 24 tasks complete

rfcbot commented on 16 Feb

🔔 **This is now entering its final comment period**, as per the review above. 🔔

rfcbot commented on 16 Feb

🔔 **This is now entering its final comment period**, as per the review above. 🔔

rfcbot added **final-comment-period** and removed **proposed-final-comment-period** labels on 16 Feb

cramertj commented on 16 Feb Member

@rfcbot Are you having a bad day? **@sfackler** hasn't checked their box yet.

cramertj commented on 16 Feb Member

@rfcbot fcp cancel

rfcbot commented on 16 Feb

@cramertj proposal cancelled.

rfcbot removed the **final-comment-period** label on 16 Feb

cuviper commented on 16 Feb Member

That's probably the new FCP process in action:
https://internals.rust-lang.org/t/psa-tweaks-to-fcp-process/6775

cramertj commented on 16 Feb Member

@cuviper oh gosh darn it! I didn't see that-- time to undo my cancels :) Thanks for the heads-up.

cramertj commented on 16 Feb

Member

@rfcbot fcp merge

rfcbot commented on 16 Feb • edited by cramertj

Team member **@cramertj** has proposed to merge this. The next step is review by the rest of the tagged teams:

- [x] **@alexcrichton**
- [x] **@aturon**
- [x] **@cramertj**
- [x] **@dtolnay**
- [x] **@eddyb**
- [x] **@nikomatsakis**
- [x] **@nrc**
- [x] **@pnkfelix**
- [ ] **@sfackler**
- [x] **@withoutboats**

No concerns currently listed.

Once a majority of reviewers approve (and none object), this will enter its final comment period. If you spot a major issue that hasn't been raised at any point in this process, please speak up!

See this document for info about what commands tagged team members can give me.

rfcbot added the **proposed-final-comment-period** label on 16 Feb

rfcbot commented on 16 Feb

🔔 **This is now entering its final comment period**, as per the review above. 🔔

rfcbot added the **final-comment-period** label on 16 Feb

rfcbot removed the **proposed-final-comment-period** label on 16 Feb

rfcbot commented on 26 Feb

The final comment period is now complete.

🎉 10

PlasmaPower commented on 16 Mar

Contributor

This should be stabilized, right?

👍 2

mark-i-m commented on 17 Mar

Contributor

I would like to give stabilizing a feature a try :)

Does anyone know what the difference between the `i128` feature and the `i128_type` feature is?

PlasmaPower commented on 17 Mar • edited

Contributor

I've always used `i128_type` , but all the docs refer to `i128` . I think they might be aliases.

SimonSapin commented on 17 Mar Contributor

They're likely the language features (the primitive types) vs the library features (various APIs that involve those types).

PlasmaPower commented on 17 Mar Contributor

I thought so too, but both the i128 primitive and the std::i128 module pages both have `i128` listed as the feature gate in the documentation.

cuviper commented on 17 Mar **Member**

Yes, `i128_type` is the language feature, and `i128` is for the library implementations. The latter includes all of the inherent methods you find on the primitive page.

PlasmaPower commented on 17 Mar Contributor

Oh, that's a bit confusing. Is there a reason for their separation? I'm assuming we want to stabilize both?

durka commented on 17 Mar • edited Contributor

@PlasmaPower lang and lib features are always separate -- lang features are listed in `src/libsyntax/feature_gate.rs` and checked at various places in the compiler, while lib features are simply scraped from `#[unstable]` attributes in the code.

There's a third feature, `repr128` , that also points to this tracking issue. It covers `#[repr(u128)]` enums, which cause problems if you look too closely at their discriminants. This is just a note to whomever does the stabilization PR (**@mark-i-m**), to keep this issue open or make a new one for `repr128` .

mark-i-m commented on 17 Mar Contributor

@durka Thanks! I will stabilize `i128` and `i128_type` and *not* `repr128` .

👍 3

mark-i-m commented on 17 Mar Contributor

So do I make a PR for the book and reference before I make PR for the feature gate?

mark-i-m referenced this issue on 17 Mar

Stabilize 128-bit integers :tada: #49101 Merged

2 of 2 tasks complete

est31 commented on 17 Mar Contributor

@durka in fact, this separation seems to have been caused by tidy behaviour. It is relaxed since some time already: #43247 So the separation is more of a historic artifact.

SimonSapin commented on 17 Mar Contributor

Docs for the primitive type are at https://github.com/rust-lang/rust/blob/1.24.1/src/libstd/primitive_docs.rs#L766-L776. They also have a "library" `#[unstable]` attributes. As far as I know language feature flags don't show up in rustdoc at all.

This was referenced on 18 Mar

Add 128-bit to types rust-lang-nursery/reference#273 Merged

Add 128-bit ints to ch3 rust-lang/book#1230 Merged

i128 is being stabilized rust-lang-nursery/compiler-builtins#235 Merged

stable_features allowed temporarily rust-lang-nursery/compiler-builtins#236 Merged

alexcrichton added a commit to alexcrichton/rust that referenced this issue on 23 Mar

`Rollup merge of #49101 - mark-i-m:stabilize_i128, r=nagisa` … Verified ✗ `20b5bf6`

kennytm added a commit to kennytm/rust that referenced this issue on 24 Mar

`Rollup merge of #49101 - mark-i-m:stabilize_i128, r=nagisa` … Verified `ef563a6`

bors added a commit that referenced this issue on 24 Mar

`Auto merge of #49101 - mark-i-m:stabilize_i128, r=nagisa` … ✗ `8d41c9f`

bors added a commit that referenced this issue on 25 Mar

`Auto merge of #49101 - mark-i-m:stabilize_i128, r=nagisa` … ✗ `97f3424`

bors added a commit that referenced this issue on 26 Mar

`Auto merge of #49101 - mark-i-m:stabilize_i128, r=nagisa` … ✓ `188e693`

mark-i-m commented on 27 Mar Contributor

I think this is done 🎉

🎉 1 ❤️ 1

est31 commented on 27 Mar Contributor

@mark-i-m congrats!

Just don't close this issue yet as like **@durka** pointed out, the `repr128` feature is still pointing to this tracking issue.

👍 1 ❤️ 1

mark-i-m commented on 27 Mar Contributor

Could someone update the OP too create more check boxes for repr128?

ebfull commented on 29 Mar Contributor

The checkbox for #45676 is checked in this issue but #45676 is not actually closed. What's the status of lowering on other platforms?

nagisa commented on 29 Mar Contributor

I *think* that is fixed, although not in the nicest way. I closed the issue.

est31 commented on 29 Mar Contributor

@ebfull see my comment here: #35118 (comment)

i128 works on the entire x86 family as well as the ARM family. This covers all of the tier 1 platforms. The platforms where we lack i128 support are tier 2 or 3.

What you are doing in ebfull/pairing#80 is exactly what I wanted to prevent by blocking stabilisation until all platforms support i128 lol, but people thought otherwise.

ebfull commented on 30 Mar Contributor

@est31 I'm not doing ebfull/pairing#80 *until* it works on all platforms, which I mention in that issue. In the mean time all I will do is remove the `i128_type` feature flag, but still require users to opt into usage of `u128` .

est31 commented on 31 Mar Contributor

@ebfull not blaming you. I'm partially blaming myself, thought that lowering worked already. Partially I'm blaming those people who decided to stabilize before all platforms support it. And the backend vendors who refuse to take {u,i}128 seriously.

Sadly, this is more of a "stable beta" release of i128 than an arrival of a feature that can be relied upon.

👍 1

hdevalence commented on 2 Apr

It would be nice if there was a (documented) way to set a cargo feature as the default on a given architecture. From what I can tell this isn't quite possible, since the architecture-selection code happens with `#[cfg(...)]` s while the crate is being compiled, by which point the crate's features are already selected. Am I missing something?

The motivation is that even if `u128` s are available, it may not be desirable to use them, unless they actually correspond to instructions available on that platform.

durka commented on 2 Apr Contributor

You can set cfg flags from the build script.
...

hdevalence referenced this issue in **dalek-cryptography/curve25519-dalek** on 6 Apr
Use the u64 backend by default on x86_64 #126 Open
0 of 3 tasks complete

dtolnay referenced this issue in **serde-rs/serde** on 17 May
i128 and u128 integers missing Deserialize impls #1136 Closed

Centril added **disposition-merge** **finished-final-comment-period** and removed **final-comment-period** labels 26 days ago

kngwyu referenced this issue in **PyO3/pyo3** 17 days ago
Add 128bit integer support #173 Merged