

## PSA: Movement of rust-lang crates

alexcrichton

Sep '15

Good evening rustlers! Today the libs subteam has implemented **RFC 1242** which changes where a number of the rust-lang crates are located, and to ensure that the word gets out I'd like to summarize what happened here.

**tl;dr;** Crates are now found in the new **rust-lang-nursery** and **rust-lang-deprecated** organizations, and those in the nursery may eventually move to rust-lang while those in deprecated would appreciate an active maintainer!

### RFC 1242 Recap

The main purpose of **RFC 1242** was to detail an explicit plan for grown a larger set of 'official' libraries outside of std as well as set forth a path to move large sets of functionality into the standard library itself. The large number of crates that are currently in rust-lang are a bit of a mishmash of high to low quality in varying degrees of upkeep. In order to organize this, a vision was spelled out for how crates become an "official" crate of the rust-lang organization. The life cycle looks like:

- Crates first enter the rust-lang-nursery organization as "0.X.Y" crates.
  - This represents no major commitment on behalf of rust-lang but rather simply a "trial period".
  - The original author of the crate maintains control, approving PRs, editing code, etc.
  - The broader community is encouraged to participate in development and crates in the nursery are more broadly advertised.
- At some point, crates in the nursery either move to rust-lang via an accepted RFC or to rust-lang-deprecated/another owner at a point decided by the library subteam.
  - An RFC for movement to rust-lang is required as it signifies that ownership is being transferred to the community, and buy in is important. Additionally, diving into API design and rationale is always useful!
  - Once in rust-lang, the crate is then owned by the Rust community at large, and major changes are governed by the library subteam in consultation with the original maintainers.
  - Future significant changes to the library require an RFC.
  - Crates in rust-lang are advertised as "core Rust" packages.
- Some crates may eventually move into std itself, at which point the library subteam will call an FCP after which the library will be folded in if accepted.
- If a rust-lang crate becomes stale over time, an RFC is written to move it to the rust-lang-deprecated organization.

If you've got any questions, please feel free to consult the **RFC itself**, or just ask questions here!

### Existing Crates

Given that background, here's a summary of where all current rust-lang crates are now located:

#### Nursery Crates

- bitflags
- getopts
- glob
- libc
- log
- rand
- regex
- rustc-serialize
- tempdir
- uuid

#### Deprecated Crates

These crates will continue to live in rust-lang-deprecated for a few reasons such as:

- Niche functionality that doesn't necessarily justify being an "official rust-lang crate".
- The crate is already deprecated in favor of another implementation (e.g. url in favor of rust-url).
- The API of the crate is in such need of an overhaul that it needs to radically change to be considered to

Sep 2015

1 / 20

Sep 2015

Nov 2015

move back into the nursery. This isn't an indication that the functionality itself is deprecated, simply that it's being messaged that it needs to be overhauled.

If you'd like to become the maintainer the library subteam is more than willing to transfer ownership to you!

- fourcc
- hexfloat
- num
- rlibc
- semver
- term
- threadpool
- time
- url

### What does this mean for me?

In short, this doesn't actually mean much for the consumers of these crates. All crates will retain the same name on **crates.io**, and the crates aren't tagged as "nursery" or "deprecated", so the compiler won't be issuing warnings or anything about usage of any of these crates.

This change is largely intended to message clearly what crates are "getting ready for stabilization" and which are "in need of dire overhaul". It's highly likely for crates like `time` to move from the deprecated area to the nursery after receiving some API scrutinization, for example.

---

### % New owners for some rust-lang crates

**steveklabnik** 

✉ Sep '15

I've already said previously that I'd adopt semver, if that's still cool.

**stebalien**

Sep '15

I'm willing to maintain (but am unlikely to improve) the `term` crate if nobody else wants it (although you'd probably be a better maintainer as you wrote almost all of the code).

**hauleth**

Sep '15

I could take care of `num` crate as I often find myself in need to use that (and if there is no one who want to do so).

**cuviper**

Sep '15

I also use `num`, and am willing to co-maintain if you like.

**alexcrichton** 

Sep '15

Thanks for volunteering! I'm also willing to take over `rlibc`, and we'll talk more about transferring the crates in the `libs` triage meeting this week if we get the chance, so I'll get back to you!

**hauleth**

Sep '15

That would be great.

**photino**







Sep '15

The numeric traits in `num` are very useful for playing with generic mathematics, such as `Float` and `Signed`. I hope that they could be added in `std::num`. What is the current status of **num-reform** RFC?

**steveklabnik** 

Sep '15

The `num-reform` RFC was what led to the creation of the `num` crate in the first place. It's still not exactly clear what we want to eventually stabilize.

<b>aturon</b> 	<b>Sep '15</b>
<p>Right. In particular, the hope was that we could explore various “numeric hierarchies” using traits <i>outside</i> of <code>std</code> for a while. It’s worth remembering that there was a long pre-1.0 history of trying different traits, and we still weren’t in a mature place.</p> <p>I would really love for some domain experts to lead the charge, here!</p>	
<b>photino</b>	<b>Sep '15</b>
<p>From my personal experience, we only need six primitive traits (<code>Int</code>, <code>UnsignedInt</code>, <code>Signed</code>, <code>Float</code>, <code>Zero</code>, <code>One</code>) in <code>std::num</code> to cover most use cases. They provide very basic functionalities for generic mathematics. Other numeric hierarchies should left to external crates.</p>	
<b>alexcrichton</b> 	<b>Sep '15</b>
<p>As an update, the libs team didn’t have time to talk about this topic this week in triage, but we’ll be sure to get to it next week!</p>	
<b>alexcrichton</b> 	<b>Oct '15</b>
<p>Alright, we discussed this at the last triage meeting and we’re all good with the transfers, if you want to sync up with me on IRC <code>acrichto</code> I’ll get the crate transferred to you!</p>	
<b>frewsxcv</b>	<b>Oct '15</b>
<p>I have some interest in maintaining the <code>threadpool</code> crate. We use it for a few projects internally at my job and am fairly familiar with internals.</p>	
<b>alexcrichton</b> 	<b>Oct '15</b>
<p>Thanks for volunteering <code>@frewsxcv</code> ! The threadpool crate has <b>now been transferred to you</b> .</p>	
<b>hauleth</b>	<b>Oct '15</b>
<p><code>@alexcrichton</code> would you create GH organization for me and <code>@cuviper</code> and move <code>num</code> into it?</p>	
<b>alexcrichton</b> 	<b>Oct '15</b>
<p><code>@hauleth</code> I’ve replied in a PM, we can discussion off-thread from here 😊</p>	
<b>briansmith</b>	<b>Nov '15</b>
<p>What’s the status of the <code>num</code> crate? I want to contribute code to it. The <code>crates.io</code> metadata indicates that the transfer hasn’t happened yet.</p>	
<b>huon</b> 	<b>Nov '15</b>
<p>It’s <b>been transfered</b> (FWIW, github will auto-redirect transferred projects, so the <code>github.com/rust-lang/num</code> “Repository” link reported on <code>crates.io</code> will actually get you there).</p>	
<b>cuviper</b>	<b>Nov '15</b>
<p>We haven’t published any new versions of <code>num</code> yet, but probably should soon. We’re definitely taking contributions though!</p>	