Huon Wilson – Rust Contributions

I have been contributing to Rust as a volunteer for almost 3 years, with a 3 month stint in the middle of 2015 as an intern. In this time, I've landed more than 800 commits into the main project, opened nearly 500 issues, and performed code-review on & approved nearly 600 pull requests by others. I have been on the Rust team (attending weekly design meetings) since early 2014, on the core team (also attending weekly meetings) since mid 2014, and on the language design and library design subteams since they were introduced in early 2015 (with weekly design meetings for both). I have thus been engaged in the decision-making process, and especially RFCs (and even their introduction), and hence essentially every decision in the project for almost 2 years.

Some of the specific contributions I have made include:

- ◇ Designing and implementing a revamp of Rust's SIMD support, along with a support library that exposes this functionality behind a safe (and partially cross-platform) interface.
- ◇ Designing and implementing most of the `rand` crate for handling random numbers.
- ◇ Writing widely-read pieces about and explanations of Rust features, both on my blog and elsewhere, such as Stack Overflow. Others have decided to merge some of this into the official documentation.
- ◇ Optimising the standard library, both asymptotically and streamlining to reduce constant factors and generate better machine code (along with many other examples).
- ◇ Implementing a high performance `sort` for the standard library, using low level `unsafe` code, while still being careful to justify the possible failures, and even test the most dangerous situations exhaustively.
- ◇ Being involved in and even resolving many design decisions, from tiny-but-ubiquitous things like the name of the `parse` method on strings, to large items like `rand` and SIMD.
- ◇ Doing a large rewrite of the compiler's code for `#[derive]` to share code, reduce bugs and make it trivial to introduce new modes.
- ◇ Making many developer quality-of-life improvements to the compiler and ecosystem, such as writing documentation, adding more detailed warnings via static analysis, and implementing core parts of the stability system that allows Rust to consider stability a deliverable. Outside the compiler, I maintain the widely-used `travis-cargo` script that makes it easy for people to automatically test their Rust code.
- ◇ Introducing guard-based RAII for unlocking `Mutex`, a pattern that is now key to Rust enforcing "lock data, not code" in concurrent code.
- ◇ Implementing the improvements to Rust's concurrency traits that allow for safely borrowing data between threads, so that one thread can have pointers directly into the stack of another, with no risk of memory unsafety.
- ◇ Publishing, maintaining and contributing to libraries outside the compiler and standard library. The most popular of mine is `simple_parallel`, which builds on the previous point to make it extremely easy to parallelise operations over iterators.
- ◇ Organising and speaking at Rust events in Sydney.