Why Rust?

Bart Massey 2023

- Hundreds people have made the case for Rust, so this document needs to be special
- Rust is usually said to be about memory safety without GC, which is its big deal
- I will argue that even without this it has big advantages for large projects, especially low-level ones

Modularity

- Rust is a truly modular language. That is, Rust has support for namespace-based modules as part of the language: no preprocessor (C/C++) or simple inclusion (Python)
- Rust has built-in support for compiling standalone packages ("lib crates") that integrate smoothly and are source-portable

Third-Party Crates

- Rust has a package-manager-style ecosystem using Cargo ("`crates.io`"). This is the default way to use the language
- `crates.io` quality tends to be decent and well-managed
- `crates.io` is not *required* for external crates
  - Crates can be accessed directly with git by Cargo
  - Crates can be vendored as part of a project
  - Bespoke `crates.io` instances can be used
- Rust's crate interfaces tend to be reasonably narrow
- Rustdoc is really good, enabling easy use

Errors Are Caught Early

- Errors that can reasonably be detected at compile time mostly are
- Errors that slip through and can be detected at runtime almost always are. In particular, no UB in safe Rust, just panics

Rust's Standard Library Is Sweet

- Good choices about what belongs in
- Extremely well built and and documented

- "Favored crates" style allows `std` stability while making good things available

Rust Has A Good Standard Style

- Rustdoc is fairly mandatory
- Compiler lints plus Clippy enforce style guidelines
- Rustfmt enforces code formatting
- Community ethos is to use all this

Rust Encourages Disciplined Data Design

- Some anti-patterns in data design, such as weird interlinking of data structures through pointers ("spaghetti data") are very difficult to implement in Rust
- Rust's module interface style encourages narrow, well-defined data interactions

Rust Integrates Well with C/C++

- No need to "rewrite the world" to work with Rust in existing projects
- Targeting Rust to pain points is nice

Rust's People Are Amazing

- Organized effort to make a friendly, people-safe ecosystem
- Sincere desire to bring people in and help them
- Ability to directly influence the language and ecosystem

The Counter-Argument For C/C++

- Rust is a complicated language compared to C
- Rust lacks some C++ "power features", notably around template typing. Some substitutes are available...
- C/C++ has a ginormous ecosystem — albeit hard to access and integrate
- Learning a new language is a pain. Rust's learning difficulty is overrated, but still very real
- Rust can make you feel dumb. Humility is required

Oh Yeah, The Memory Safety Thing

- It really is a big deal