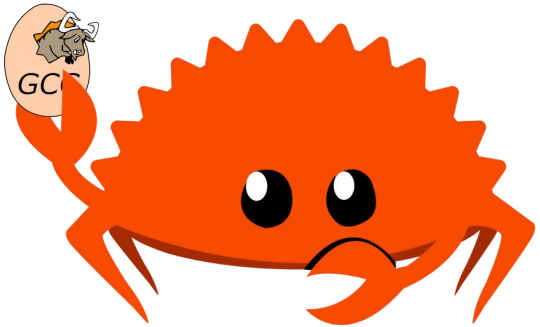
EMBECOSM®

OPEN SOURCE SECURITY

Rust-GCC



Philip Herron
Arthur Cohen

Summary

- Project recap
- Current status
- Compiling Rust-for-Linux using `gccrs`
  - Requirements
  - Adding it to our testing project

What is Rust GCC?

- Full Implementation of Rust on top of GNU Toolchain
  - Project originally started in 2014, revived in 2019
    - Progress stalled with the frequency of language changes
  - Two full time engineers
    - Receives contributions from many GCC and non GCC developers
  - Thanks to Open Source Security, inc and Embecosm

Motivations of Rust GCC

- Upstream with mainline GCC
- Reuses the GNU toolchain (`ld`, `as`, `gdb`)
- Reusing official Rust `libcore`, `libstd`, `libproc`
- Alternative implementation of Rust
- GCC Plugins support
  - LTO and CFI
- Drive adoption of Rust through backporting
- Backend support for more systems
- https://github.com/Rust-GCC/gccrs/wiki/Frequently-Asked-Questions

Current status

- Const generics
- Intrinsics
- Borrow-checking
- Working towards running the `rustc` test-suite
- Target an older version of `libcore`
- A first experimental release should be available in GCC 13 (next release)

Compiling RfL with gccrs

- Rust 1.62!
- Compiler flags
  - Improve our `cargo_gccrs` wrapper
  - Reuse `rustc`'s argument parsing library
  - Pull requests welcome!
- `libcore`
  - Currently targeting 1.49
  - Too old for Rust-for-Linux
- `liballoc`
  - Custom?
  - Depends on `libcore`

Rust version differences

- What does it mean?
- Few language differences
- Mostly library differences (additions to `core`, `std`)
- Lots of hidden additions
  - Nightly APIs
  - Unstable attributes, macros, intrinsics
  - That RfL probably relies on
  - ...right?

Rust version differences

```
export RUSTC_BOOTSTRAP := 1
```

Testing project

- Tries compiling various projects using `gccrs`
  - `blake3` cryptography library
  - `libcore` 1.49
  - All the valid cases from the `rustc` testsuite
    - in `#[no_std]` mode
    - in `#[no_core]` mode
- Eventually add RfL to it!

Community

Get Involved

- Goal is to make working on compilers fun
  - Lots of `good-first-pr` issues to work through
    - Refactoring work
    - Bugs
  - Lots of scope to make your mark on the compiler
- Google Summer of Code 2021 and 2022
- Status reporting
  - Weekly and Monthly
  - Shout out to contributors
  - Open and transparent
- Monthly Community Call
  - 1st Friday of the Month 09h00 UTC
  - Open to everyone who is interested
  - Hosted on Jitsi

Future Work

- Cross Compiler Testing Project
  - Compare Rustc vs Rust GCC
    - Error diagnostics
    - Code Size
    - Energy Efficiency
    - Benchmarking

- Language Standardization
  - Integration with the Rust community
  - `crater` runs
    - Automate testing Rust GCC against code from https://crates.io/

Links

- Github: https://rust-gcc.github.io/
- Reports: https://github.com/Rust-GCC/Reporting
- Email: philip.herron@embecosm.com
- Zulip: https://gcc-rust.zulipchat.com/
- IRC: irc.oftc.net #gccrust
- https://gcc.gnu.org/mailman/listinfo/gcc-rust

EMBECOSM®

Special Thanks

- Brad Spengler
  - https://opensrcsec.com/
- Jeremy Bennett
  - https://www.embecosm.com/
- David Edelsohn
  - https://gcc.gnu.org/steering.html

EMBECOSM®

Questions?

www.embecosm.com