arm

Rust and the Trustedfirmware.org Project Ecosystem

Community Discussions

Joanna Farley/Dan Handley

May 2023

Objective

- Seek Trustedfirmware.org Member Feedback
  - Initially with Steering Committee discussion
    - Agree high-level Trustedfirmware.org position
  - Follow up when/if required with individual project discussions
    - Using high-level Trustedfirmware.org position as a framework

- So why talk about Rust?
  - Rust is rapidly increasing in popularity
  - Rust has a particular focus on safety and robustness
  - Rust was designed for and is being adopted by Firmware/Embedded software
  - Could be attractive to Trustedfirmware.org community
  - Should not ignore it or risk future community disruption

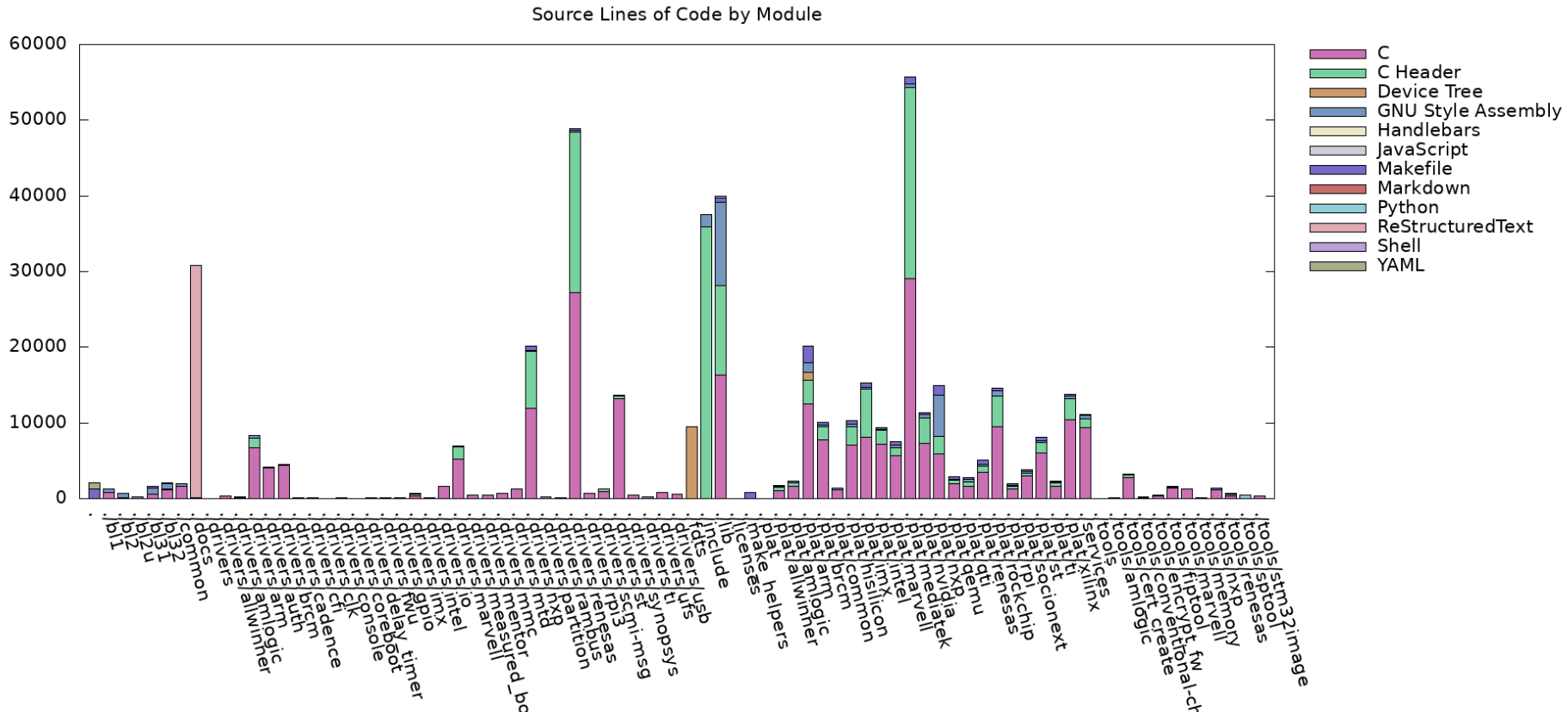
Overview

- Three broad usage of Rust
  - New project
    - Primarily Rust but possibly some C linkage (e.g. Parsec, rtic, embassy, opentitan, hubris, …)
  - Alternative project
    - Originally C and reimplemented in Rust (e.g. Coreboot -> Oreboot)
  - Contributions Project
    - Primarily C project that can take Rust Contributions (e.g. Linux Kernel that can take drivers written in Rust)
      - Additional examples: Zephyr, riot-os, vxworks, chromium, …
- Provide guidance for existing TrustedFirmware.org projects is the initial priority
  - Details are likely to be very Project specific
  - Project community need to be engaged
  - Likely through a Techforum Discussions
- Quick Example
  - Of possibilities to give discussion a real context using TF-A

TF-A

https://ci-builds.trustedfirmware.org/static-files/Nz0yWDDcjRUtFTxFT52d04od3FGmGbnfDKQECE_-5oMxNjgxMzg5Mjk2NjA5OjE2OmpvYW5uYWZhcmxleS1hcm06am9iL3RmLWEtc2xvYy12aXN1YWxpemF0aW9uLzY1L2FydGlmYWN0/output.png

TF-A Codebase and Contributions



Discussion Points (assuming language advantages are a given)

- Training / Resourcing
  - Increasingly popular, especially for younger engineers, so easy to attract engineers in short term
  - May be more difficult to retain expertise for long-term maintenance
  - Steep learning curve
    - In exchange, potentially spend less time on code reviews, debugging and maintenance. More robust product.
  - Possibly more resources needed for projects with mixed language support or during migration phase
- Cargo – de-facto build system and package manager; Crates.io – default package repo
  - Pros: Enables standardized project layout and development tools, extensive library support
  - Cons: Can result in many dependencies – higher risk of supply chain attacks, license compliance issues
  - High update frequency also increases these risks (but has benefit of getting latest features, bug fixes)
  - Could use managed package repo or (temporarily) lock dependencies or pass on risk to deployments
- License – most of ecosystem uses "MIT OR Apache-2.0"
  - Not standard for TF.org (would require board approval) but good for compatibility
  - To aid compliance, Cargo.toml files have "package.license" fields (typically SPDX tags), which Cargo can parse for dependency checking – risk of not matching source!
  - No equivalent copyright field, only (original) author

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה