ELISA
Enabling Linux in Safety Applications

WORKSHOP

Role of Rust in Safety Critical Applications

and potential Implications for ELISA

Paul Albertella, Codethink
Daniel Krippner, ETAS/Bosch

Rust for Linux

https://rust-for-linux.com/

- Greg K-H on Rust in the kernel:
  - https://www.phoronix.com/news/Greg-KH-On-New-Rust-Code
- Process of opening up kernel for Rust has already uncovered bugs
- Potential for engaging new generation of developers

Rust Toolchain

Certified compilers from Ferrous Systems, AdaCore

- Language specification used to certify Ferrocene donated to Rust Foundation
  - https://blog.rust-lang.org/2025/03/26/adopting-the-fls/

Rust vs MISRA

An investigation of Rust using the exida ctools suite in 2022 :

- Set of tests for C to compare effectiveness of code analysis tools
  - Considered around 100 of the test cases for Rust
- Rust, by construction, prevents two thirds of the test cases
  - i.e. the test cases simply would not compile
- At runtime, Rust's controlled panics would catch a further 25% of the tests.
  - Using `no-panic` crate, most would be convertible to compile-time failures, bringing the "not possible by construction" percentage up as high as 90%

Rust and security

Tight correlation between cybersecurity and safety

- Rust was designed with cybersecurity in mind
- Some classes of issues that are relevant to safety are already known and managed as a a result of this
- Process aspects of cybersecurity are different, but many of the software aspects are the same

Reasons to choose over C for new development

Codethink decision to write the Safety Monitor in Rust based

- Considered using C, but complexity of some features made Rust more appropriate
    - Enabled developers without specialised skill set to contribute with confidence
    - Reduced need for supervision by safety experts, because most classes of error were already addressed by the language
- Motivated a younger cohort of engineers to engage with the challenge
- Longer-term maintenance is easier, because the rules don't need to be policed
    - Reduces cognitive load of programmer, because they don't have to think about these rules

Debugging Rust

- First class integrated tooling in IDE
- Can still use valgrind and tools designed to work on binaries, not C code
  - These tools using debug symbols and ld-preload
  - e.g. cachegrind, memcheck, hellgrind

Licensing of Workshop Results

All work created during the workshop is licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) [https://creativecommons.org/licenses/by/4.0/] by default, or under another suitable open-source license, e.g., GPL-2.0 for kernel code contributions.

You are free to:

- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Questions?