Discussion: Userspace Libraries

TockWorld 7 | June 2024

State of Userspace Libraries

- C
  - Widest support for drivers
  - Fully relocatable (on ARM)
    - Limitation of non-ARM GCC PIC
  - Primary use cases:
    - “Legacy” application libraries (OpenThread, TPM2, LVGL …)
    - Compatibility
- Rust
  - Limited upstream driver support
  - Basically not relocatable
    - Limitation of LLVM PIC
  - Primary use cases
    - “Greenfield” applications

Exciting Developments

- `YieldWaitFor` system call can better support synchronous I/O
  - Significant quality-of-life improvement for “mostly” async applications
  - Simpler drivers for non-concurrent applications
  - (Modest) code-size improvements for synchronous code (more significant on RISC-V)
- Relocation is “easy” in integrated use-cases (with non-XIP code storage)
- Rust embedded ecosystem can fill gap in high-level code
  - E.g., see `embedded-graphics` support in libtock-rs

Challenges: Maintenance & Support: libtock-rs

Current maintainers:

- Johnathan Van Why
- Alistair Francis
- (implicitly also Core WG sometimes)

Porting new drivers is challenging

- Kernel-level documentation? Stability
- Complexity of Rust userland?

Building applications and libraries out-of-tree possible but not first-class

Challenges: Maintenance & Support: libtock-c

Current maintainers:

- Implicitly Core WG

Building applications and libraries out-of-tree is hard

- No standard C build system
- Distribute multi-arch object files

Duplication of sync and async interfaces

- Mostly reuse same code, but `YieldWaitFor` might change that

External stability

Thematic Challenge

Limited open examples that reflect real use cases

We have:

- Test applications
- Sensor network-y applications
- Toy root-of-trust

We don’t have

- “Automotive”
- “Complete” root-of-trust (maybe OpenSK?)

Discussion

- Beefing up maintenance and support for libtock-rs
- Story for libtock-c maintenance
- Automating userland driver generation
- Getting representative applications upstream