Entwicklung von Kernelkomponenten in Rust am Beispiel eines Dateisystemtreibers

Luca Kleinschmidt, Jonas Dittrich, Clemens Tiedt, Andreas Grapentin, Andreas Polze

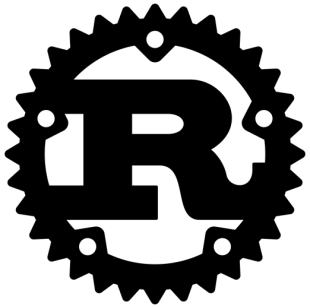
Warum nicht C?

> I did an analysis of all the commits in the null_blk driver (currently 256 exluding merge commits). 27% (68) of these commits are bug fixes. Out of these 27%, 41% (28) are fixes for memory safety issues. These are issues that would be avoided in a Rust based implementation.

- Windows 11 Schriftdarstellung neu in Rust geschrieben
  - Viele Speicherfehler und Überläufe
  - Nicht nur schneller sondern auch sicherer

Was ist Rust?

- „A language empowering everyone to build reliable and efficient software.”
- Seit 2006 in Entwicklung, stable seit 2014
- Speichersicherheit und sichere Nebenläufigkeit durch Compile Time Checks
- „Ownership“ und „Lifetimes“ unterstützen bei Speicherverwaltung
- Explizite Veränderlichekit vereinfacht Datenfluss



Struktur von Rust for Linux

- Primäre Aufgabe von "Rust for Linux":
  - Bereitstellen von Bindings und Abstraktionen für Kernel Bibliotheken
- Darauf aufbauend: Rust Kernel-Module
  - Apple M1 & M2 GPU Treiber
  - NVMe Treiber (Western Digital)
  - PuzzleFS

Geschichte

- Kommerzielles Interesse von Firmen
  - Samsung -> Treiber
  - Google ...
- Bindings seit Linux 6.1 im Mainline Kernel
- Ab 6.3 / 6.4 erste Rust Treiber zu erwarten
- aarch64 Support seit Anfang 2023

Rust for Linux am Hasso-Plattner-Institut

RamFS 2020

- Noch keine offizielle Unterstützung im Kernel
  - Nur im Emailverteiler angekündigt
- Projektstart bevor Rust for Linux funktionsfähig
- Selbstgeschriebene Bindings und C Kernel-Module zum Rust Code gelinkt
- Jedoch bereits lauffähig

RamFS 2021

- Rust for Linux nimmt Gestalt an
- Reimplementierung mit Rust for Linux Bindings
- Eigene Wrapper um Bindings für C-Funktionen
- Eigene Interfaces für Dateisystemoperationen
- Beginn FAT Implementierung in Rust
  - Teilen sich Abstraktionen
  - Aktueller Stand: Partitionen können gemountet werden
  - Linux 5.x


Universität
Potsdam


HPI
Hasso
Plattner
Institut
Digital Engineering · Universität Potsdam

RamFS 2023

Projekt im Rahmen der Betriebssysteme 2 Vorlesung

- Rust lernen
- Kontakt mit der Rust for Linux Community
- Migration zu Linux 6.x
- Migration zu veränderten Rust for Linux Abstraktionen

Verwenden der R4L Abstraktionen für Dateisysteme

- Cisco PuzzleFS Weiterarbeit an Abstraktionen
- Geänderte Struktur der Abstraktionen im Kernel
  - Reimplementierung der Funktionen von 2021
  - Basis Linux 6 und aktualisierte Bindings
  - Rust for Linux Bindings noch unvollständig
  - Erweiterung nach Funktionsbedarf

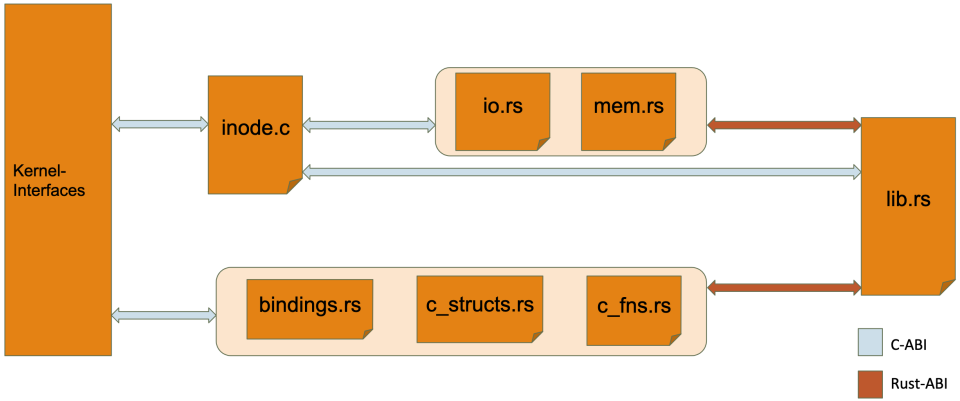
Erfahrung und Evolution am Beispiel der Registrierung eines Dateisystems

Vor Rust for Linux Unterstützung

- Registrierung via C Macros

```c
static int __init init_ramfs_module(void)
{
    printk(KERN_INFO "Starting module...\n");
    return register_filesystem(&ramfs_type);
}

static void __exit exit_ramfs_module(void)
{
    unregister_filesystem(&ramfs_type);
    printk(KERN_INFO "Stopping module\n");
}
```



Rust for Linux Abstractions

```
module! {
    type: BS2Ramfs,
    name: b"bs2ramfs",
    author: b"Rust for Linux Contributors",
    description: b"RAMFS",
    license: b"GPL v2",
}

struct BS2Ramfs;
```

Rust for Linux Projektorganisation

- Projektstruktur Parallel zu Linux Mailing List
- Zustimmungen zuerst von Rust for Linux
- Zustimmung auch von Linux Maintainer der Komponente erforderlich
- Rust for Linux Kommunikation via Online Chatportal
  - Geringere Einstiegshürde
  - Kurze Feedbackschleife
  - Offener und freundlicher Umgangston

Learnings

- Der Linux Kernel entwickelt sich stetig weiter
- Große Unternehmen experimentieren, genauso wie Universitäten und Studenten
- Viele Parteien haben mitzureden

=> Rust for Linux ist ein junges Projekt, das schnell wächst

- Linux-Kernel ist komplex, aber Teile sind innerhalb eines Semesters lernbar
- Rust for Linux-Community ist sehr offen und hilfsbereit
- Viele Probleme (z.B. VM-Konfiguration zum Entwickeln) tauchen jedes Semester auf => gute Dokumentation hilft