Rust – eine Einführung

Dr. Christoph Zimmermann

Kieler Open Source und Linux Tage 2022

17. 9. 2022

stat /proc/self

- Promotion im Bereich reflektive Betriebssystemarchitekturen
- Start mit Linux: Kernel 0.95
- Tech Support @ FraLUG
- Hobbies:
  - SDLC
  - IT-Sicherheit und andere Formen schwarzer Kunst
  - Arch Package Maintainer

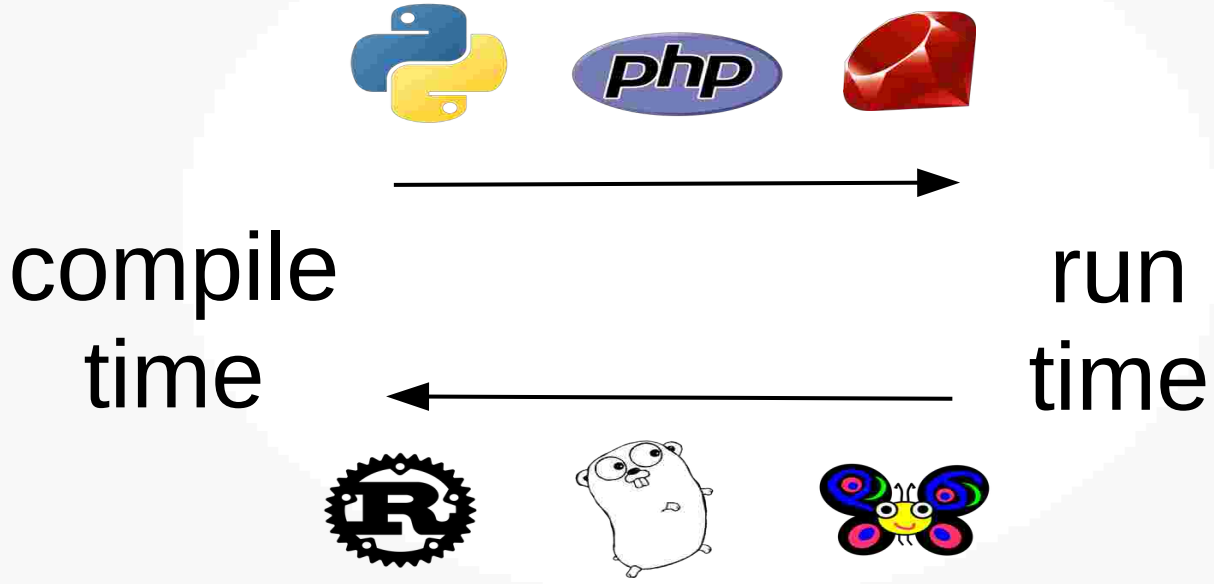
linuxinlaws.eu

cat /etc/motd

- Einführung
- Rust – die Sprache
- Pakete & cargo
- Compiler & mehr
- Case Study
- Ausblick

info rust

- Philosophie:



info rust (ff.)

- Mozilla Research, 7/7/10
- Schnelle, sichere und parallele Programmiersprache
- Compiler-basierend mit strikter statischer Typisierung
- Umfangreiches Modul-Ökosystem unterstützt durch Paket-Manager
- Fokus auf Standardisierung

man rustc

- Memory management:
  - Keine GC
  - Optimiert für Geschwindigkeit
- Regeln:
  - Jede Variable hat einen Besitzer
  - Nur *ein* Besitzer während der Ausführung
  - Besitzer verläßt Scope => Variable wird freigegeben

=> Referenzen / Borrowing / Lifetimes

man rustc (ff.)

- Rust: keine “wirkliche” Objekt-Orientierung
- Grundsätzliche Abstraktionen:
  - Enums: mächtiger als in C/C++
  - Structs: Abstrakte Datentypen
  - Generics: Instanziierte Typen ähnlich wie in Java / C++ Templates
  - Traits: vergleichbar zu Mixins in OOPLs

=> Grundlage für strikte Typprüfung bereits während der Übersetzung

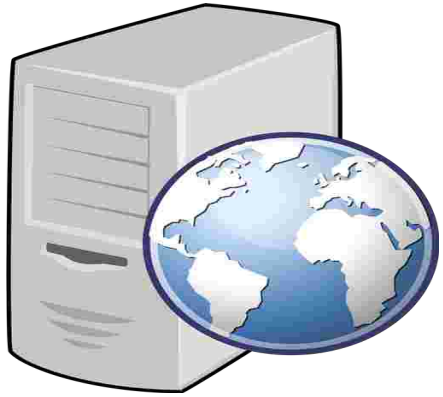
man rustc (ff.)

- Paketierung:
  - Module: structs / enums / impl(mentierungen) => Namespaces
  - Crates: Sammlung von Modulen (vergleichbar mit Packages)
  - crates.io: *die* Rust Community Crate Registry
- Vor Neuerfindung des Rads => crates.io
- cargo: Umfangreiches Paketierungs- / Build-System (=> make on speed :-)

man rustc (ff.)

- Andere Spracheigenschaften:
  - Closures: Anonyme Funktionen (eg. Lambdas)
  - Macros (streng typisiert, auch für AST-Verwaltung benutzt)
  - Unterstützung mehrere Multiprocessing / Threading Modelle bereits durch Standard-Bibliothek
  - Umfangreiche Standard-Bibliothek (vergleichbar mit Python, etc.)

webserver &



rustc -v

- Ökosystem:
  - IDEs: VS Code, Intellij IDEA, Eclipse (volle RLS-Unterstützung)
  - Standard-Compiler: pro Benutzer, Betriebssystem-Pakete
  - Debugging: gdb, lldb
  - Plattformen: Linux, OSX, Windows
  - H/W: i686*, amd64*, ARM (32/64*), MIPS, PPC

* Volle Q/A

sum

- Vorteile:
  - Stark typisierte, schnelle Programmiersprache
  - Umfangreiches Ökosystem
  - Ideal für sichere Systemprogrammierung
- Nachteile:
  - Steile Lernkurve (Typisierung)
  - Keine richtige OOPL (noch nicht)
  - Nicht geeignet für Anfänger

=> Meine Meinung, andere können dies anders sehen :-)

shutdown -H +5

- Foundation (Feb. 2021):
  - AWS, Microsoft, Google, Mozilla, etc.
- Nur einige Beispiele:
  - Mozilla: Servo, Quantum
  - Google: Android, Fuchsia, ChromeOS
  - Microsoft: Azure IoT Edge
  - Redis Labs: redis modules
  - Linux Kernel Module Framework

shutdown -H +5 (ff.)

```
Date  Mon, 19 Jan 2004 22:46:23 -0800 (PST)
From  Linus Torvalds <>
Subject  Re: Compiling C++ kernel module + Makefile
On Tue, 20 Jan 2004, Robin Rosenberg wrote:
[...]
In fact, in Linux we did try C++ once already, back in 1992.
It sucks. Trust me - writing kernel code in C++ is a BLOODY STUPID IDEA.
The fact is, C++ compilers are not trustworthy. They were even worse in
1992, but some fundamental facts haven't changed:
 - the whole C++ exception handling thing is fundamentally broken. It's
   _especially_ broken for kernels.
 - any compiler or language that likes to hide things like memory
   allocations behind your back just isn't a good choice for a kernel.
 - you can write object-oriented code (useful for filesystems etc) in C,
   _without_ the crap that is C++. [...]
```

shutdown -H +5 (ff.)

- Werkzeug-Unterstützung für automatische C-Bindings Generierung (via libffi)
- Crate-Implementierung für Kernel I/F
- Momentan amd64, aarch464, PPC und RISC-V Unterstützung (andere Architekturen in Arbeit / geplant)
- Voraussetzung: Rust nightly + clang
- `github.com/fishinabarrel/linux-kernel-module-rust`

apropos rust

- rust-lang.org: Offizelle Webseite + Dokumentation
- crates.io: Crate Registry
- github.com/rust-lang/rust: Source Code Github Repo
- this-week-in-rust.org: News und wöchentliche Updates
- newrustacean.com: Podcast (+1, leider keine neuen Episoden)
- rustacean-station.org: Community-Podcast
- request-for-explanation.github.io/podcast: Diskussionen von Rust RFCs
- gist.github.com/mjohnsullivan/e5182707caf0a9dbdf2d: web server

Fragen?

Danke!

Dr. Christoph Zimmermann

monochromec at <ignore>space</ignore>gmail<dot></dot>com