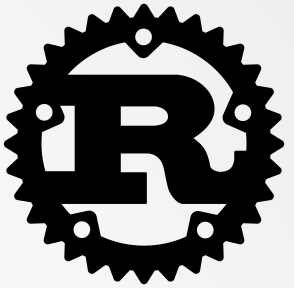


Rust
The programming language and its ecosystem

Dr. Christoph Zimmermann

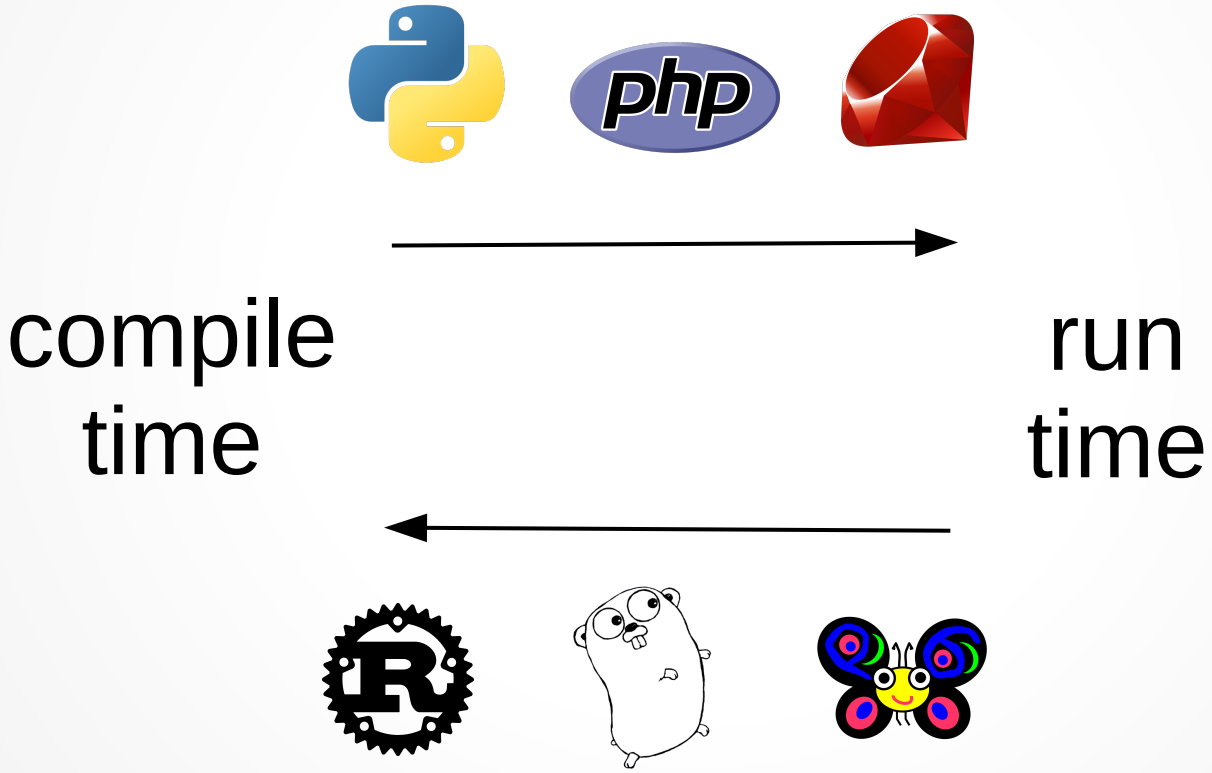
22. 10. 2019

cat /etc/motd

- Introduction
- Rust – the language
- Packages & cargo
- Tool chain support
- Case study
- Outlook

info rust

- Main philosophy:



info rust (ctd.)

- Mozilla Research, 7/7/10
- High performance, memory-safe concurrent language
- Compiler-based with strong static type system
- Comprehensive module ecosystem powered by standard package manager
- Focus on standardization

man rustc (ctd.)

- Memory management:
  - No GC
  - Optimized for speed
- Rules:
  - Each variable has an owner
  - Only *one* owner at any time
  - Owner leaves scope => value dropped

=> References / borrowing / lifetimes

man rustc (ctd.)

- Rust: not a real OOPL
- Main abstractions:
  - Enums: more powerful than in C/C++
  - Structs: abstract data types
  - Generics: instantiated types similar to Java / C++ templates
  - Traits: comparable to mix-ins in OOPLs

=> Foundation for strong type checking at compile time

man rustc (ctd.)

- Packaging:
  - Modules: structs / enums / impl(mentations) => principal namespaces
  - Crates: collection of modules, comparable to packages
  - crates.io: *the* Rust community crate registry
- Before reinventing the wheel, check crates.io
- cargo: comprehensive package / build management system

man rustc (ctd.)

- Other language features:
  - Closures: anonymous functions (eg. lambdas)
  - Macros (strongly typed, also used for AST handling)
  - Various multiprocessing / threading models already supported by standard library
  - Comprehensive standard library (similar to Python)

webserver &

rustc -v

- Toolchain support:
  - IDEs: VS code, Intellij IDEA, Eclipse (full RLS support)
  - Standard compiler: per-user tool chain, OS packages
  - Debugging: gdb, lldb
  - Platforms: Linux, OSX, Windows
  - H/W: i686*, amd64*, ARM (32/64), MIPS, PPC

* Full Q/A

cat /etc/motd

- Pros:
  - Strongly typed, high-performance language
  - Comprehensive ecosystem
  - Ideal for secure system programming
- Cons:
  - Steep learning curve (esp. type system)
  - Not a real OOPL
  - Unsuitable for learning programming

=> My $.02, your mileage may vary

shutdown -H +5

- Rapidly gaining community momentum:
  - Servo, Quantum
  - Tor
  - Azure IoT Edge
  - Nu shell
  - boringtun: user space WireGuard implementation
- Redis modules
- Kernel module framework

shutdown -H +5 (ctd.)

```
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 (ctd.)

- Tool chain support for automatic C bindings generation (via libffi)
- Crate implementation for kernel i/f
- Only amd64 support at the moment (other architectures are WiP)
- Rust nightly + clang
- `github.com/fishinabarrel/linux-kernel-module-rust`

apropos rust

- rust-lang.org: official language website including documentation
- crates.io: crate registry
- github.com/rust-lang/rust: source code Github repo
- this-week-in-rust.org: weekly news & updates
- newrustacean.com: weekly podcast (+1)
- rusty-spike.blubrry.net: weekly podcast (dead?)
- request-for-explanation.github.io/podcast: weekly discussion of Rust RFCs
- gist.github.com/mjohnsullivan/e5182707caf0a9dbdf2d: web server foundation

Questions?

Thank you!

Dr. Christoph Zimmermann

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