Duck-Tape Chronicles
Rust/C++ Interop

Joint Rust & C++ Meetup

Prague

November 2024

@ciura_victor
@ciura_victor@hachyderm.io
@ciuravictor.bsky.social

Victor Ciura
Principal Engineer
Rust Tooling @ Microsoft

Abstract

C++/Rust is not a zero-sum game. We need to learn to play nice together... for a looong time! That applies equally to people, but also to code. Rust code everywhere is increasing at an accelerated rate, but so does C++ (and that's on top of gazillion lines already out there). So, hybrid codebases are quickly becoming the norm.

Having seamless interop between the C++ and Rust components is essential for the success of this symbiosis. There are many challenges in this process, but people found various ways to make things "work" - from dealing with ABI compatibility and platform/toolchain guarantees, to going down to C and FFI, to various techniques and tools for generating glue-code between the two languages.

Alas, general-purpose interoperability (not tied to a specific toolchain/IR) without loss of performance has yet to be achieved. Just "making things work" is not enough in the domain space of C++ and Rust; as such, many of the explored solutions so far by the community fail to deliver on all the needed requirements, swinging the wide range between performant and ergonomic.

This presentation aims to highlight all of the interop challenges, some of the explored solutions out there, and tease out the avenues at the forefront of this pursuit.

About me

Advanced Installer

Clang Power Tools

Oxidizer SDK

Visual C++

Rust Tooling

🐦 @ciura_victor

🐘 @ciura_victor@hachyderm.io

🦋 @ciuravictor.bsky.social

Rust ❤️ C++

Not a zero-sum game.

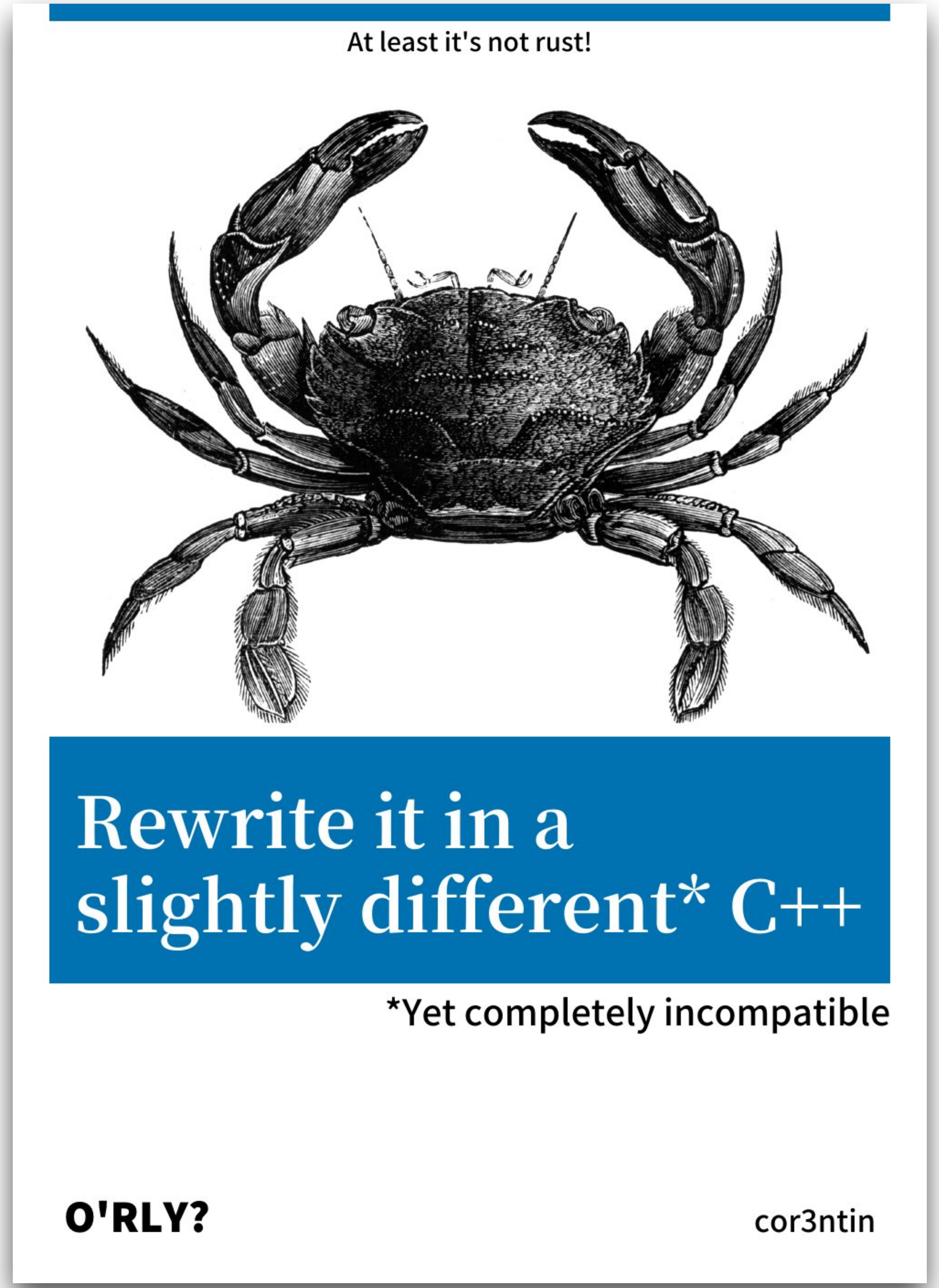
We need to learn to play nice together... for a looong time!

The Playbook to Safety

Rewrite it in... ~~Rust~~ SafeC++? 🤦

- A cakewalk and eating it too
- But we care about safety, right?
- Dogma... galore
- Down with Safety!
- To UB or not to UB? 💀
- Correct by confusion
- ++(C++) successor languages
- Borrowing the borrow checker

⚙️ Circle Compiler
safecpp.org/P3390


At least it's not rust!

Rewrite it in a slightly different* C++

*Yet completely incompatible

O'RLY?

cor3ntin

cor3ntin.github.io/posts/safety/

The Playbook to Safety

Rewrite it in... ~~Rust~~ SafeC++? 🤦

- A cakewalk and eating it too
- But we care about safety, right?
- Dogma... galore
- Down with Safety!
- To UB or not to UB? 💀
- Correct by confusion
- ++(C++) successor languages
- Borrowing the borrow checker

I'm not selling any RiiR or SafeC++ snake oil

⚙️ Circle Compiler
safecpp.org/P3390

At least it's not rust!

Rewrite it in a slightly different* C++

*Yet completely incompatible

O'RLY?

cor3ntin

cor3ntin.github.io/posts/safety/

Google

“Based on historical vulnerability density statistics, Rust has proactively prevented hundreds of vulnerabilities from impacting the Android ecosystem. This investment aims to expand the adoption of Rust across various components of the platform.”
– Dave Kleidermacher, Google Vice President of Engineering, Android Security & Privacy

“While Rust may not be suitable for all product applications, prioritizing seamless interoperability with C++ will accelerate wider community adoption, thereby aligning with the industry goals of improving memory safety.”
– Royal Hansen, Google Vice President of Safety & Security

- https://security.googleblog.com/2024/09/eliminating-memory-safety-vulnerabilities-Android.html
- https://security.googleblog.com/2024/10/safer-with-google-advancing-memory.html

Microsoft

Ongoing efforts:

- Making step-changes in our **SDL** operations and making additional investments to meet the evolving needs of cloud and emerging technologies
- Completing our deployment of **CodeQL**, integrated with GitHub Copilot learnings
- Continue to invest in **hardening** C & C++ code
  (see my EuroRust 2024 presentation & Amanda's CppCon 2024 Keynote)
- Standardizing on **Rust** and other memory safe languages (MSLs)
- Contribute 💰 to support the work of the Rust Foundation
- Assist developers making the transition from C, C++, C# to Rust
  - we will continue to invest 💰 in Rust developer tooling

Rust and C++ interoperability

It's important for Rust to be able to call C++ functions (and vice-versa) in a way that meets the following criteria:

Choose... some?

- No need for excessive unsafe keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad type support - with safety
- Ergonomics - with safety
- Works with dynamic libraries (including the weirdness* of Windows DLLs, CRT)
- Plays well with C++ ABI
- Easily automated
- Hybrid build systems (CMake, cargo, bazel, MSBuild, ...)

Rust and C++ interoperability

There's (some) progress in Rust community in solving some of these problems

⚙️ autocxx, bindgen, cbindgen, diplomat, cxx, crubit, etc.

But way more work is needed here!

🚧 💰

cxx / crubit

- cxx optimizes for simplicity, at the expense of requiring manual duplication of the C++ API in Rust and accepting some (small) performance overhead
  - given the simplicity advantage of cxx, if one can use cxx instead of Crubit for their application today, they probably should do so
- Crubit is Google's C++/Rust bidirectional interop tool
  - attempts to provide a bridge for every API, such that we merge the C++ and Rust ecosystems
  - aims to address the case where this level of duplication has unreasonable maintenance costs for large codebases and even a tiny performance overhead is unacceptable

Open Discussion

What does interop mean for you?

What are the interop requirements of your project?
(constraints, limitations, priorities, goals)

Duck-Tape Chronicles
Rust/C++ Interop

Joint Rust & C++ Meetup

Prague

November 2024

@ciura_victor
@ciura_victor@hachyderm.io
@ciuravictor.bsky.social

Victor Ciura
Principal Engineer
Rust Tooling @ Microsoft