TRUE LEGENDS

IT PROFESSIONALS

MEETUP: Rust meets C++

Dennis J. Alberts en Sander Geerts

TRUE LEGENDS

IT PROFESSIONALS

MEETUP: Rust meets C++

Dennis J. Alberts en Sander Geerts

Speakers

Dennis

- 7+ years experience as SE
- Did projects for different customers:
  - Alstom, Kverneland, Shell, JOZ, Somnox, Marine
- Used different languages
  - C, C++, C#, Fortran, Java, Python, Bash..
  - Only using Rust for hobby projects

Sander

- 12+ years of experience as SE
- Lead software engineer @ JOZ
- Embedded Linux
- Used different languages
  - **C/C++**, Bash, C#, *Lua, Python*, *Java*
  - New to Rust

TRUE LEGENDS
IT PROFESSIONALS

Introduction

- About Rust
- Why consider Rust?
- Transition strategies
  - Neutral Intermediary
  - Incremental replacement
- Conclusion
- Questions

Introduction

- Integrating Rust into a C++ codebase.
- Growing interest in Rust:
  - There has been a push to stop using languages that do not account for memory safety [1][2][3]. Start using Rust?
  - Rust is being used in the development of the Linux Kernel [4] since version 6.1. (11 December 2022)
- Cyber secur

Linus Torvalds: Rust will go into Linux 6.1

At the Kernel Maintainers Summit, the question wasn't, "Would Rust make it into Linux?" Instead, it was, "What to do about its compilers?"

TRUE LEGENDS

IT PROFESSIONALS

- Introduction
- **About Rust**
- Why consider Rust?
- Transition strategies
  - Neutral Intermediary
  - Incremental replacement
- Conclusion
- Questions

About Rust

- Rust was created as a personal project by Graydon Hoare in 2006 @Mozilla Research
- Promising replacement for C and C++, particularly for systems-level programming, infrastructure projects, and embedded software development.
- Version 1.0 was released in 2015
- Major upgrades were made in 2018 and 2021.

TRUE LEGENDS
IT PROFESSIONALS

- Introduction
- About Rust
- **Why consider Rust?**
- Transition strategies
  - Neutral Intermediary
  - Incremental replacement
- Conclusion
- Questions

Why consider Rust?

Rust is a system programming language that offers several key features making it a popular choice for developers:

- Guaranteed Memory Safety
- Threads without Data Races
- Zero Cost Abstraction
- Error Messages
- Pattern Matching
- Efficient C Bindings

TRUE LEGENDS

IT PROFESSIONALS

- Introduction
- About Rust
- Why consider Rust?
- **Transition strategies**
  - Neutral Intermediary
  - Incremental replacement
- Conclusion
- Questions

Transition Strategies

- As we all know there is only one thing constant in Software Engineering and that is that everything **CHANGES**.
- As organizations mature so do their requirements which puts increased pressure on Software Engineering to deliver a secure stable product.
- Improved tooling, of which Rust can be a part, is the solution.
- But let's be honest! Nobody is going to replace their entire codebase in one go!

Transition Strategies

- New Projects Only
- Neutral Intermediary
- Incremental Replacement of Existing Code

TRUE LEGENDS
IT PROFESSIONALS

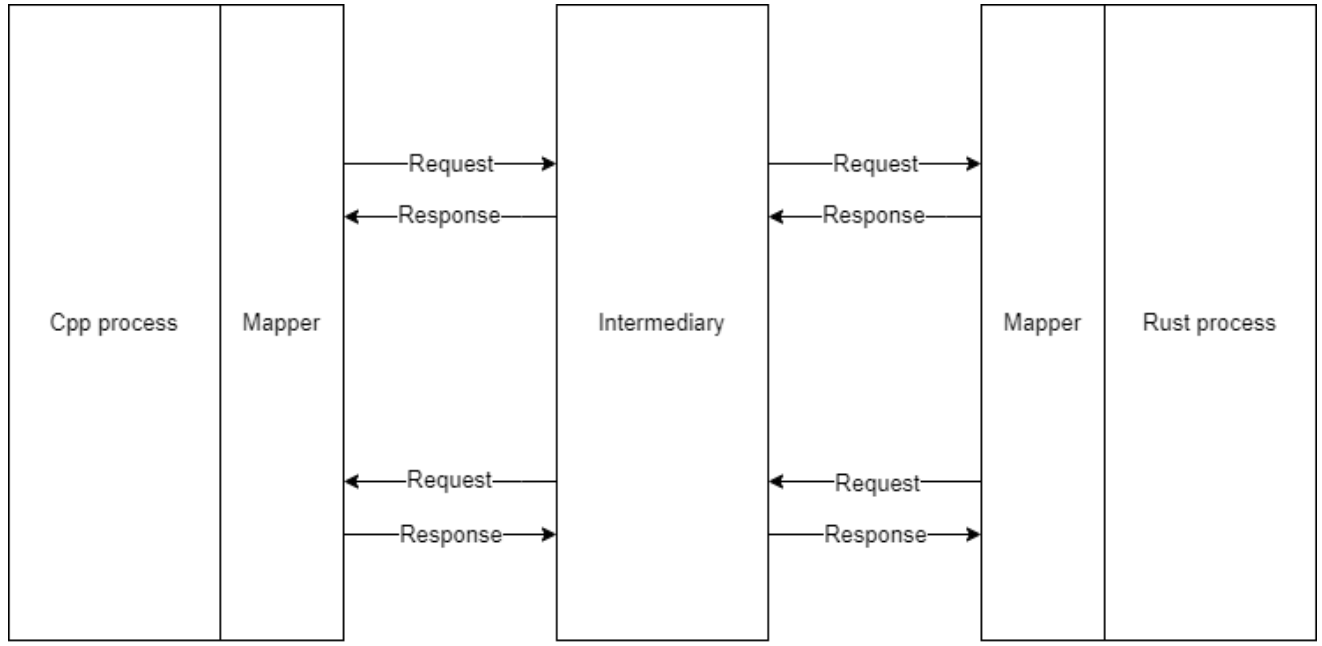
- Introduction
- About Rust
- Why consider Rust?
- Transition strategies
- **Neutral Intermediary**
  - Incremental replacement
- Conclusion
- Questions

Neutral Intermediary

- A layer is used between to processes.
  - IPC (inter process communication)
  - Third-party framework.

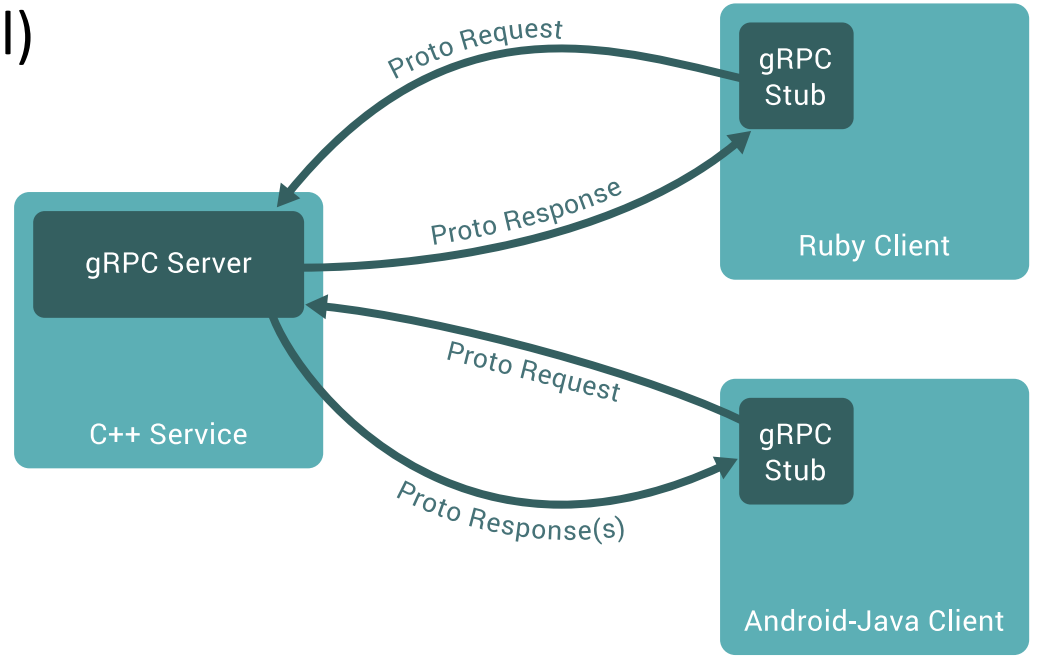
Neutral Intermediary

- Create a new Rust process that lives next to the C++ process.
- Establish a connection through a neutral intermediary.
- Map properties to message.
- Communicate with other process
- Receive response.
- Map response back to domain.



Neutral Intermediary

- gRPC (google Remote Procedure Call)
  - Security
  - Error handling
- Define proto files
  - Messages
  - Service
- Protoc compiles to C++/Rust code
  - Tonic crate for Rust
- Use classes to do remote calls



Demo: Neutral Intermediary – gRPC protobuf

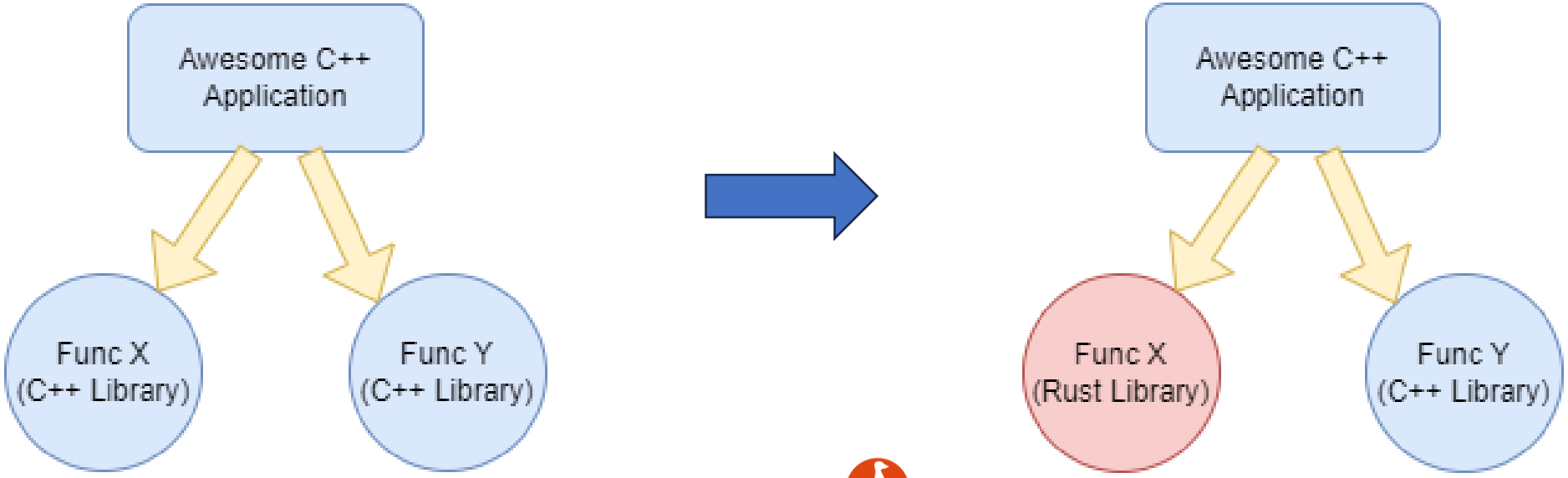
TRUE LEGENDS

IT PROFESSIONALS

- Introduction
- About Rust
- Why consider Rust?
- Transition strategies
  - Neutral Intermediary
  - Incremental replacement
- Conclusion
- Questions

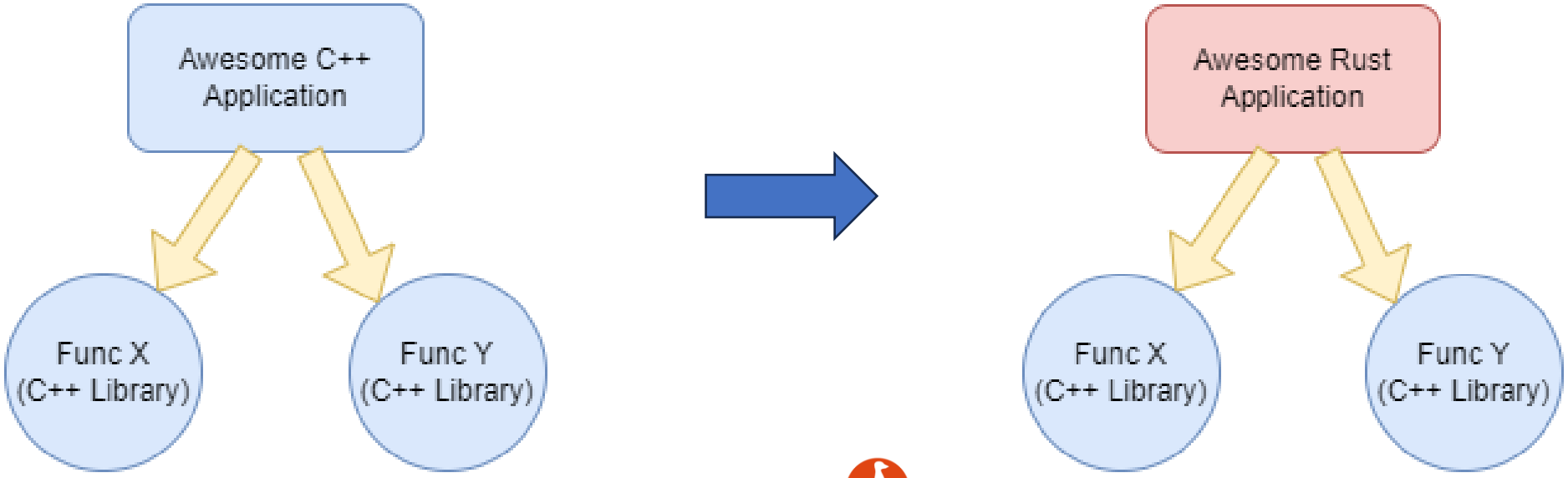
Incremental Replacement

- Bottom to Top
  - C/C++ as controlling layer using Rust components



Incremental Replacement

- Top to Bottom
  - Rust as controlling layer using C/C++ components



Incremental Replacement – FFI

- Foreign Function Interface, also known as FFI
- Cumbersome to create/maintain by hand
- Generators to the rescue

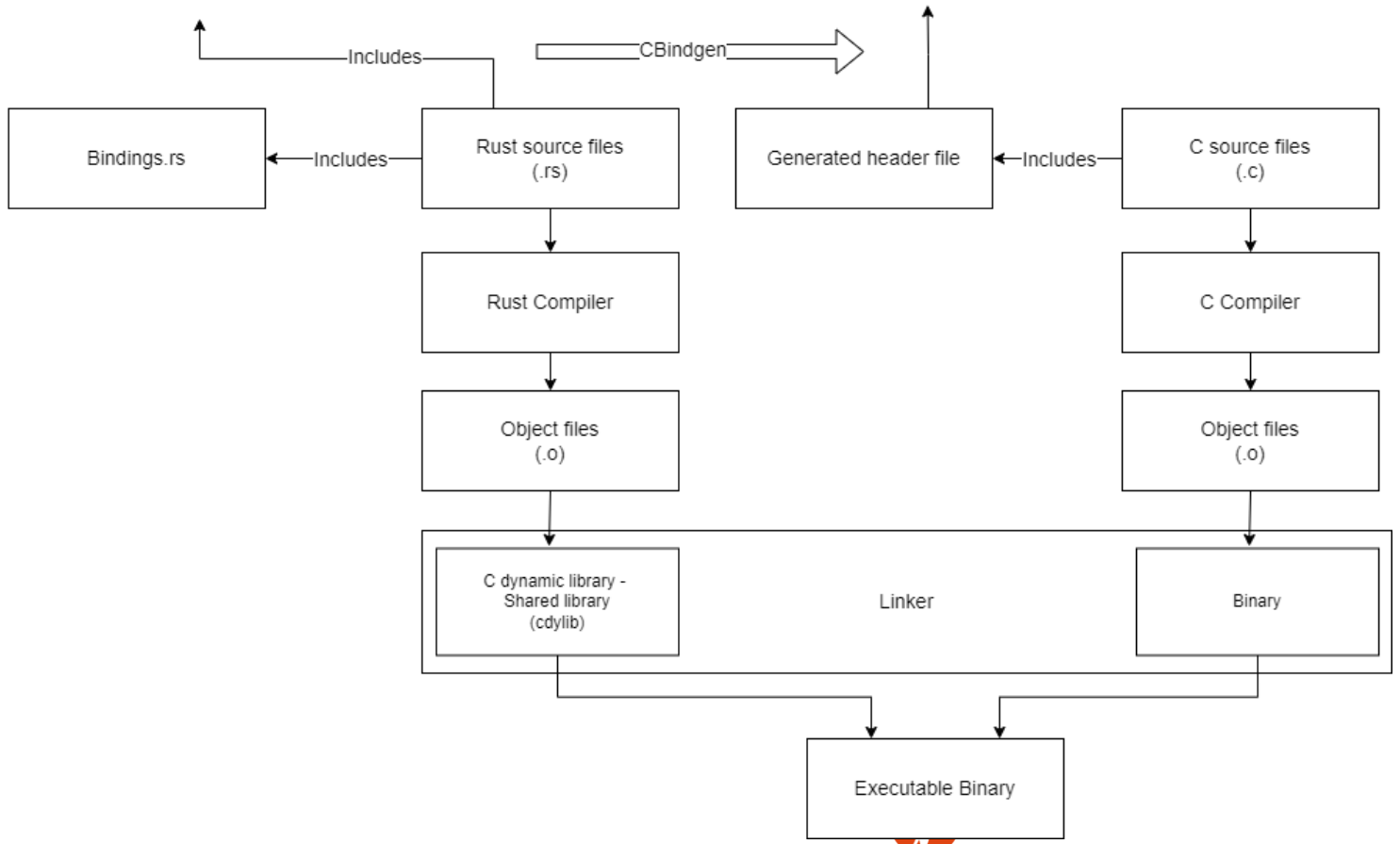
C++ -> Rust with CBindgen

- C++ controller layer
- Rust functionality
- CBindgen automatically generates C header that can be used by C++ to call Rust functions.

C++ -> Rust with CBindgen

```rust
#[no_mangle]
pub extern "C" fn add(left: i32, right: i32) -> i32 {
    left + right
}
```

```c
int32_t add(int32_t left, int32_t right);
```



Demo: C++ -> Rust with CBindgen

Rust -> C++ with Bindgen

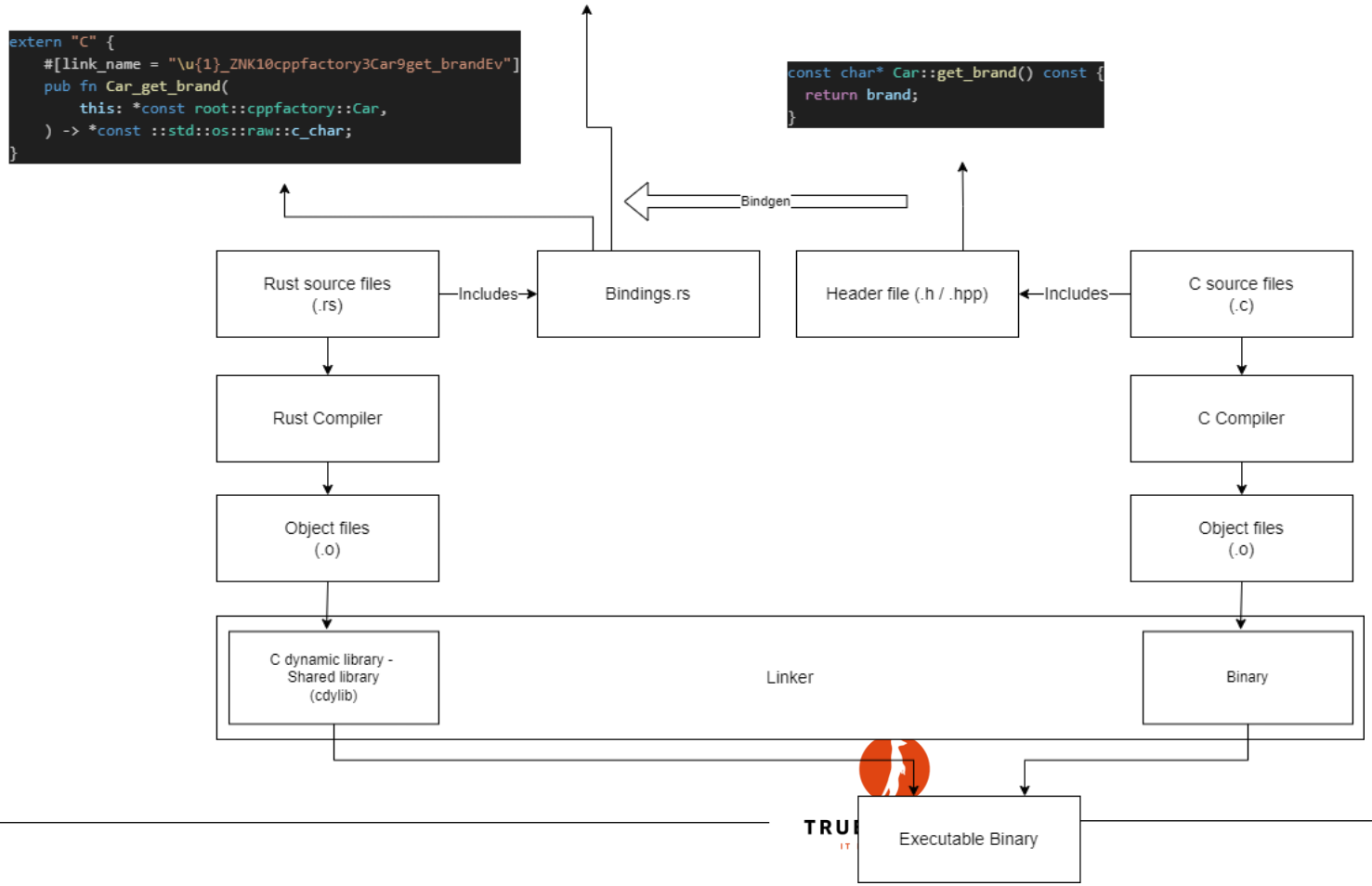
- Rust controller layer
- C++ library
- Bindgen automatically generates Rust file that can Rust can use

Rust -> C++ with Bindgen

```rust
impl Car {
    #[inline]
    pub unsafe fn get_brand(&self) -> *const ::std::os::raw::c_char {
        Car_get_brand(self)
    }
}
```

```rust
extern "C" {
    #[link_name = "\u{1}_ZNK10cppfactory3Car9get_brandEv"]
    pub fn Car_get_brand(
        this: *const root::cppfactory::Car,
    ) -> *const ::std::os::raw::c_char;
}
```

```cpp
const char* Car::get_brand() const {
  return brand;
}
```



Demo: Rust -> C++ with Bindgen

TRUE LEGENDS

IT PROFESSIONALS

- Introduction
- About Rust
- Why consider Rust?
- Transition strategies
  - Neutral Intermediary
  - Incremental replacement
- **Conclusion**
- Questions

Conclusion

- Integrating Rust into C/C++ is a powerful approach to gradually harness the advantages of Rust
- Tooling is continually improving

Try it out yourself!

https://gitlab.com/djalberts/cpp-and-rust

TRUE LEGENDS

IT PROFESSIONALS

- Introduction
- About Rust
- Why consider Rust?
- Transition strategies
  - Neutral Intermediary
  - Incremental replacement
- Conclusion
- Questions

Questions?

- `[no_std]` means that the standard lib is not included. Therefore, a reduced set of functionality is available when using Rust. Mostly applied in bare metal programming.
  - Note: No stack overflow protection
- For bare metal embedded development, crates, such as, mynewt and embassy can be used. Both are RTOS variants.

Bibliography

[1] M. Russinovich, “Speaking of languages, it's time to halt starting any new projects in C/C++ and use rust for those scenarios where a non-GC language is required. for the sake of security and reliability. the industry should declare those languages as deprecated.,” *Twitter*, 19-Sep-2022. [Online]. Available: https://twitter.com/markrussinovich/status/1571995117233504257?lang=en. [Accessed: 14-Feb-2023].

[2] “Rust in the linux kernel,” *Google Online Security Blog*, 14-Apr-2021. [Online]. Available: https://security.googleblog.com/2021/04/rust-in-linux-kernel.html. [Accessed: 14-Mar-2023].

[3] “White House warns against using memory-unsafe languages,” The New Stack, Available: https://thenewstack.io/white-house-warns-against-using-memory-unsafe-languages/ [Accessed: Mar. 26, 2024].

[4] “Linus Torvalds: Rust will go into linux 6.1,” *ZDNET*. [Online]. Available: https://www.zdnet.com/article/linus-torvalds-rust-will-go-into-linux-6-1/. [Accessed: 16-Mar-2023].

[5] “MSRC-security-research/2019_01 - bluehatil - trends, challenge ... - github.” [Online]. Available: https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2019_02_BlueHatIL/2019_01%20-%20BlueHatIL%20-%20Trends%2C%20challenge%2C%20and%20shifts%20in%20software%20vulnerability%20mitigation.pdf. [Accessed: 13-Mar-2023].

[6] “Chrome: 70% of all security bugs are memory safety issues,” *ZDNET*. [Online]. Available: https://www.zdnet.com/article/chrome-70-of-all-security-bugs-are-memory-safety-issues/. [Accessed: 13-Mar-2023].