COMP6991 23T1

Foreign Function Interface

How to talk to external code

that (*gasp*) *may not* have been written in Rust!?

> Spawn a process
  > Communicate e.g. stdin/stdout
> Be spawned!
  > Communicate e.g. stdin/stdout

> File system
> Raw socket

> HTTP?
> Carrier pigeon?

What about performance?

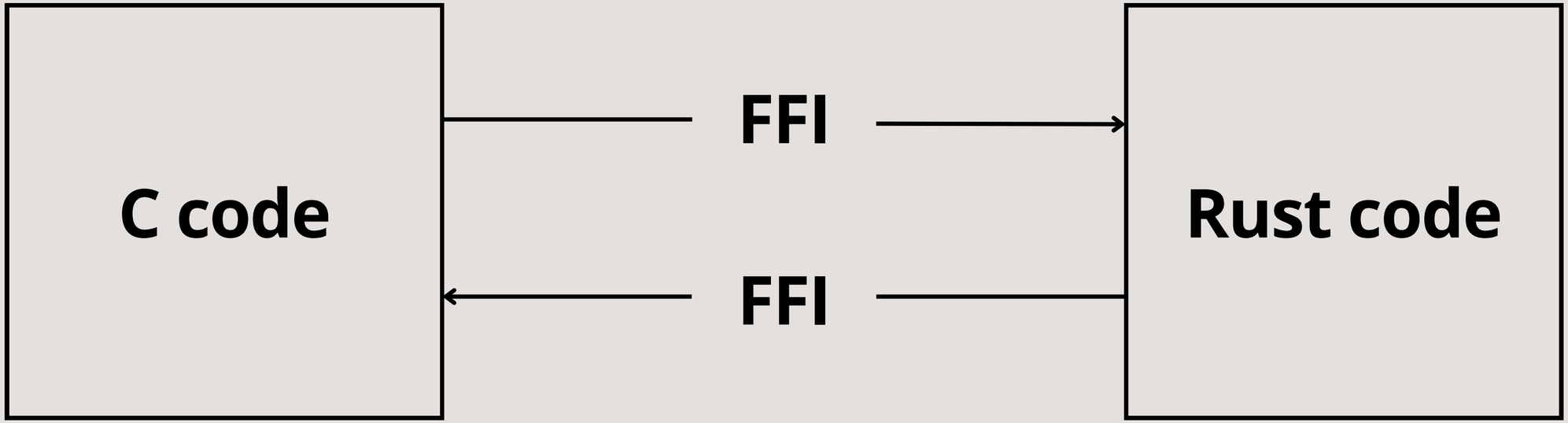
(spoiler: it's probably slow)

What about platform availability?

(spoiler: it's probably a pain on embedded)


FFI

Overall linked program



Useful resources

> Nomicon FFI
  > https://doc.rust-lang.org/nomicon/ffi.html

> std::ffi

> Unofficial FFI guide
  > https://michael-f-bryan.github.io/rust-ffi-guide/
  > Somewhat incomplete

> Libc Rust bindings
  > https://crates.io/crates/libc

Calling libcurl from Rust

(insanity edition)

We're gonna need these...

https://curl.se/libcurl/c/

https://curl.se/libcurl/c/libcurl-easy.html

https://curl.se/libcurl/c/simple.html

https://github.com/curl/curl/blob/master/include/curl/curl.h

What sucked?

(about that whole libcurl thing)

I don't like generating `extern "C" { … }` manually!

(me neither!)

> Rust bindgen

  > Automatically generates Rust FFI bindings to C (and some C++) libraries.

  > https://github.com/rust-lang/rust-bindgen

> For the other direction... C bindgen!

  > cbindgen creates C/C++11 headers for Rust libraries which expose a public C API.

  > https://github.com/eqrion/cbindgen

But I don't want to use [c]bindgen!

(me neither!)

> Someone probably already has!

> Rust bindings

> e.g. https://docs.rs/curl-sys/latest/curl_sys/

> e.g. https://docs.rs/sqlite3-sys/latest/sqlite3_sys/

But I don't want to use C types everywhere!

(me neither!)

> Build an abstraction on top!

> Make it safe! (Safe abstraction over an unsafe implementation)

> e.g. https://docs.rs/curl/latest/curl/

> e.g. https://docs.rs/sqlite/latest/sqlite/