FFI Workshop

By Jonathan Louie

Agenda

- What is FFI and why do we care?
- FFI Basics
- Writing Safe Code
- Hands-on Example
- Additional Resources

What is FFI and why do we care?

Definition

- Stands for “Foreign Function Interface”
- Allows Rust code to call code written in other languages
- Allows code written in other languages to call Rust code
- Usually done through C ABI

Why FFI?

- **Leveraging Rust's performance**
  - **Scripting languages, such as Python, JavaScript, etc.**
- **Leveraging existing libraries in other languages**

FFI Basics

The unsafe Keyword

- Allows you to use 5 superpowers
  - Dereference a raw pointer
  - Call an unsafe function or method
  - Access or modify a mutable static variable
  - Implement an unsafe trait
  - Access fields of a `union`
- Does NOT mean your code might trigger Undefined Behaviour

unsafe fn and unsafe block

https://play.rust-lang.org/?version=stable&mode=debug&edition=2024&gist=24b26dd3ccc02aef5496f8267bb3ebcc

The extern Keyword

- `extern` block is used to declare a foreign function that Rust can call
- `extern fn` is used to declare a Rust function that can be called by foreign code
- See https://doc.rust-lang.org/std/keyword.extern.html
  - `extern crate` is not relevant and is old syntax

Calling Conventions

- Specifies how a function is called
  - How to pass parameters
  - How to return values
- Common ABIs (Application Binary Interface)
  - “C”
  - “C-unwind”
  - “system”
- Full list: https://doc.rust-lang.org/reference/items/external-blocks.html#abi

Name Mangling

- What is name mangling?
  - Gives unique names to generated symbols
  - https://doc.rust-lang.org/stable/rustc/symbol-mangling/index.html
- Why mangle?
  - https://stephencoakley.com/2019/04/24/how-rust-solved-dependency-hell
- `extern` functions must disable name mangling
  - https://doc.rust-lang.org/stable/reference/abi.html#the-no_mangle-attribute

repr(C)

- **Rust's default data layout is not the same as C's**
  - **https://doc.rust-lang.org/nomicon/repr-rust.html**
- **`repr(C)` forces types to have same order, size and alignment of fields as C types**
  - **https://doc.rust-lang.org/nomicon/other-reprs.html**

extern fn and extern block

https://play.rust-lang.org/?version=stable&mode=debug&edition=2024&gist=4483d47c0cdb3f56424539902c5d439d

Writing Safe Code

Undefined Behaviour

- **What is Undefined Behaviour?**
  - **Cases where the compiler is free to assume your code never triggers that behaviour**
  - **Example: https://godbolt.org/z/nzhdb3dd3**
- **C Standard: https://www.csagroup.org/store/product/CSA%20ISO%25100IEC%2014882%3A21/?format=PDF**
  - **Yes, you have to PAY to see what's UB in C**
- **UB in Rust: https://doc.rust-lang.org/reference/behavior-considered-undefined.html**

Safe Abstraction

https://doc.rust-lang.org/src/core/iter/traits/iterator.rs.html#307

Tips

- **Memory allocated by one language's allocator should be freed by the same allocator**
- **Pay attention to who owns data when passing pointers across FFI boundaries**

Helpful Tools

- rust-bindgen (https://github.com/rust-lang/rust-bindgen)
  - Use it to generate Rust bindings for C/C++ libraries
  - In other words, use this when calling C/C++ from Rust
- cbindgen (https://github.com/mozilla/cbindgen)
  - Use it to generate C/C++ bindings for Rust libraries
  - In other words, use this when calling Rust from C/C++
- Sanitizers (valgrind, ASAN, MIRI)

Hands-on Example

Activity

Clone the following repository:

https://github.com/jonathanrlouie/ffi-playground

Follow the steps in the `README.md` to run both sample apps.

If you are not a Docker or Nix user, run the following commands to install cbindgen and rust-bindgen (must have Rust installed):

```
cargo install --force cbindgen
cargo install bindgen-cli
```

When you're able to run both sample apps, `git checkout workshop` and read `ACTIVITY.md`

Additional Resources

Additional Resources

- **Note: A lot of documentation (including some of these pages) have outdated examples that do not follow best practices established in Edition 2024**

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

http://jakegoulding.com/rust-ffi-omnibus/

https://github.com/Quin-Darcy/rust-c-ffi-guide