How to make bindings in Rust

... and its modern toolchain, Drop traits, and Windows suffering.

Hiroshi Hatake

Technical information sharing seminar

Introduction

What is "Binding"?

Binding is a library which call or use other language's function and feature.

Introduction

Difficutlies of "Binding"

Binding is difficult. Because ...

Introduction

Difficutlies of "Binding"

Binding is difficult. Because ...

- Need to call Foreign Function Interface

Introduction

Difficutlies of "Binding"

Binding is difficult. Because ...

- Need to call Foreign Function Interface
- Absorb language differences (C and other languages)

Introduction

Difficutlies of "Binding"

Binding is difficult. Because ...

- Need to call Foreign Function Interface
- Absorb language differences (C and other languages)
- Interact two different environment

Introduction

Difficutlies of "Binding"

Binding is difficult. Because ...

- Need to call Foreign Function Interface
- Absorb language differences (C and other languages)
- Interact two different environment
- Control mangling rules

How to create bindings in Rust

Rust has generating binding tool from C header

It is **rust bindgen**[1].

[1] https://github.com/Yamakaky/rust-bindgen

How to create bindings in Rust

Rust has generating binding tool from C header

It is **rust bindgen**[1].

Example:

```
bindgen --link lua \
  --builtins /usr/include/lua.h --output lua.rs
```

[1] https://github.com/Yamakaky/rust-bindgen

Generate binding from C header

Actual example

Then, let's generate binding for Groonga!

Generate binding from C header

Actual example

Then, let's generate binding for Groonga!

```
bindgen --link groonga \
  --builtins /usr/include/groonga/groonga.h \
  --output groonga.rs
```

Create binding crate

Actual example

Second, put into src directory.

```
% tree -L 2 .
.
├── Cargo.toml
├── build.rs
└── src
    ├── groonga.rs # <- e.g.) Put groonga.rs here!
    └── lib.rs
```

Create binding crate

Actual example

Declare using groonga module in lib.rs which is the library entry point.

```
extern crate libc;

pub mod groonga;
```

Then, Complie and fix!

Complie and fix!!

Complie and fix!!!

Until errors are dismissed....

Create binding crate

Actual example

Confirm to succeed to be built.

```
% cargo build
     Updating registry `...`
   Compiling pkg-config v0.3.8
   Compiling libc v0.2.16
   Compiling groonga-sys v0.3.0 (file:///...)
```

Create binding crate

Actual example

Confirm to succeed to be built.

```
% cargo build
     Updating registry `...`
   Compiling pkg-config v0.3.8
   Compiling libc v0.2.16
   Compiling groonga-sys v0.3.0 (file:///...)
```

Yay!

Traits, manage resources, Rustish concept

Make more Rustish

rust-bindgen does **not** generate Rustish binding.
It is auto generated and just as a set of function signature declarations.

Traits, manage resources, Rustish concept

Make more Rustish

How to make more Rustish binding?

Traits, manage resources, Rustish concept

Make more Rustish

How to make more Rustish binding?
You should know about **Traits**. Especially, **Drop trait**.

Traits, manage resources, Rustish concept

Traits

Traits tells a funtionality which must be implemented in type.

Traits, manage resources, Rustish concept

Traits

Traits tells a funtionality which must be implemented in type. It sometimes are used in generics bound.

```
fn from_iter<T: Iterator<A>>(iterator: T)
  -> SomeCollection<A>
```

Traits, manage resources, Rustish concept

Drop trait[2]

Drop trait's drop method is called when a variable goes out of scope.[3]

[2] https://doc.rust-lang.org/std/ops/trait.Drop.html

[3] Perhaps, C++ users noticed by intuition that Drop trait is simiar to concept of destructor.

Traits, manage resources, Rustish concept

Drop trait[2]

Drop trait's drop method is called when a variable goes out of scope.[3]

```
/// rustc mydrop.rs
/// ./mydrop
struct MyDrop;

impl Drop for MyDrop {
    fn drop(&mut self) {
        println!("Dropping!");
    }
}

fn main() {
    let _x = MyDrop;
}
/// #=> Dropping!
```

[2] https://doc.rust-lang.org/std/ops/trait.Drop.html

[3] Perhaps, C++ users noticed by intuition that Drop trait is simiar to concept of destructor.

Traits, manage resources, Rustish concept

Actual example

In Ruroonga, Drop trait is often used to manage allocated resources.

Traits, manage resources, Rustish concept

Actual example

In Ruroonga, Drop trait is often used to manage allocated resources.

```
pub struct LibGroonga {/* omitted */}

impl LibGroonga {
    pub fn new() -> Result<LibGroonga, String> {
        // initialize libgroonga
    }

    fn close(&mut self) -> Result<(), String> {
        // finalize libgroonga
    }
}

impl Drop for LibGroonga {
    // Called when a variable goes out of scope.
    fn drop(&mut self) {
        self.close().unwrap();
    }
}
```

Cargo's build script mechanism

About Cargo[4]

Cargo, which is the package manager for Rust, has build script feature[5].
This feature is used to customize building phase.
Some crates need to link non-Rust code. This kind of linking task sometimes should be customizable.[6]

[4] http://doc.crates.io/index.html

[5] http://doc.crates.io/build-script.html

[6] Some crate should compile C libraries before linking. And some crate should distinguish platforms whether it is Windows or not.

pkg-config

pkg-config

pkg-config is the one of great tool on UNIX like environment.

pkg-config

pkg-config

pkg-config is the one of great tool on UNIX like environment. It is a helper tool used when compiling applications and libraries.

pkg-config in Rust

pkg-config crate

Rust community already has pkg-config crate! Amazing!!

To use pkg-config, specify the following in Cargo.toml:

```
[package]
  ...
build = "build.rs"

[build-dependencies]
pkg-config = "~0.3"
```

pkg-config in Rust

pkg-config in build script

pkg-config is used in build script.

```
/// build.rs
extern crate pkg_config;

use std::env;

fn main() {
    let target = env::var("TARGET").unwrap();

    if !target.contains("windows") {
        if let Ok(info) = pkg_config::probe_library("groonga") {
            if info.include_paths.len() > 0 {
                let paths = env::join_paths(info.include_paths).unwrap();
                println!("cargo:include={}", paths.to_str().unwrap());
            }
            return;
        }
    }
}
```

Yay, it's so easy!

For Windows?

Windows support in build script

Why not support Windows?

For Windows?

Windows support in build script

Why not support Windows?
It's a nightmare for *nix developers.

For Windows?

Windows support in build script

Why not support Windows?
It's a nightmare for *nix developers.
But Rust community encourages to support Windows.

Windows support

Windows support in build script

If you use Windows, how to set environment information? Use pkg-config? No, Windows platform often lacks of it.

Windows support

Windows support in build script

If you use Windows, how to set environment information? Use pkg-config? No, Windows platform often lacks of it.
Instead, we can always use **environment variables**.

Windows support

Actual build script example

See the next page.

Windows support in build script

```
// fn main() ..
let target = env::var("TARGET").unwrap();

let lib_dir = env::var("GROONGA_LIB_DIR").ok();
let bin_dir = env::var("GROONGA_BIN_DIR").ok();
let include_dir = env::var("GROONGA_INCLUDE_DIR").ok();

if lib_dir.is_none() && include_dir.is_none() {
    if !target.contains("windows") {
        // same as before
    }
}

let lib = "groonga";
let mode = if env::var_os("GROONGA_STATIC").is_some() {
    "static"
} else {
    "dylib"
};

if let Some(lib_dir) = lib_dir {
    println!("cargo:rustc-link-search=native={}", lib_dir);
}
if let Some(bin_dir) = bin_dir {
    println!("cargo:rustc-link-search=native={}", bin_dir);
}

println!("cargo:rustc-link-lib={}={}", mode, lib);

if let Some(include_dir) = include_dir {
    println!("cargo:include={}", include_dir);
}
```

Demo

Conclusion

- rust bindgen makes easy to create binding.
- Drop trait is useful to manage allocated resources.
- cargo package manager can handle custom build script.
- cargo's build script can handle environment variables which is often used for Windows platform.
- Rust bindings sometimes works well on Windows.

Any questions?