Binding Generators

From C to Rust and Back

Florian “Florob” Zeitz

2025-05-07

Outline

1. bindgen
2. cbindgen

bindgen

bindgen

- generates low-level bindings to C libraries
- uses `clang` to parse C or C++ code
- either as separate tool, or as library (recommended)
- library can generate correct types for current target
- User Guide

Binding to a Library

- by convention low-level bindings in crates called `libxyz-sys`
- exposes C-API functions
- takes care of linking

Example: Bindings to libdrm

- create a `wrapper.h` including files we want bindings for
- in `build.rs`
  - create bindings with bindgen
  - tell `cargo` to link to libdrm
- in `lib.rs`
  - include generated bindings
  - disable unwanted warnings (e.g. non camel-case type names)

Example: wrapper.h

```
#include <xf86drm.h>
```

Example: First stab at builds.rs

```
> cargo add --build bindgen
```

```
// build.rs:
fn main() {
  let bindings = bindgen::Builder::default()
      .header("wrapper.h")
      .generate()
      .expect("Unable to generate libdrm bindings");

  let out_path = PathBuf::from(env::var("OUT_DIR").unwrap());
  bindings
      .write_to_file(out_path.join("bindings.rs"))
      .expect("Couldn't write libdrm bindings!");
}
```

Example: Include search path

```
> cargo build
Unable to generate libdrm bindings:
ClangDiagnostic("/usr/include/xf86drm.h:40:10: fatal error:
'drm.h' file not found\n")
```

- our include search path is incomplete
- we should check what to add with `pkg-config`

Example: pkg-config

```
cargo add --build pkg-config
```

```
// build.rs
let libdrm = pkg_config::Config::new().probe("libdrm")
    .expect("Unable to find libdrm");

let include_flags = libdrm.include_paths.into_iter()
    .map(|path| format!("-I{}", path.display()));

let bindings = bindgen::Builder::default()
    .header("wrapper.h")
    .clang_args(include_flags)
    .generate()
    .expect("Unable to generate libdrm bindings");
```

Example: lib.rs

```
#![allow(non_upper_case_globals)]
#![allow(non_camel_case_types)]
#![allow(non_snake_case)]

include!(concat!(env!("OUT_DIR"), "/bindings.rs"));
```

Example: Linking

```rust
// build.rs
for path in libdrm.link_paths {
    println!("cargo:rustc-link-search={}", path.display());
}

for lib in libdrm.libs {
    println!("cargo:rustc-link-lib={lib}");
}
```

Example: Restricting Symbols

- documentation shows unwanted symbols:
  - `pthread_barrier_t`
  - `uint_least32_t`
- we can restrict to useful once with allow and block lists
  - e.g. `.allowlist_item("drm_.+")`

cbindgen

cbindgen

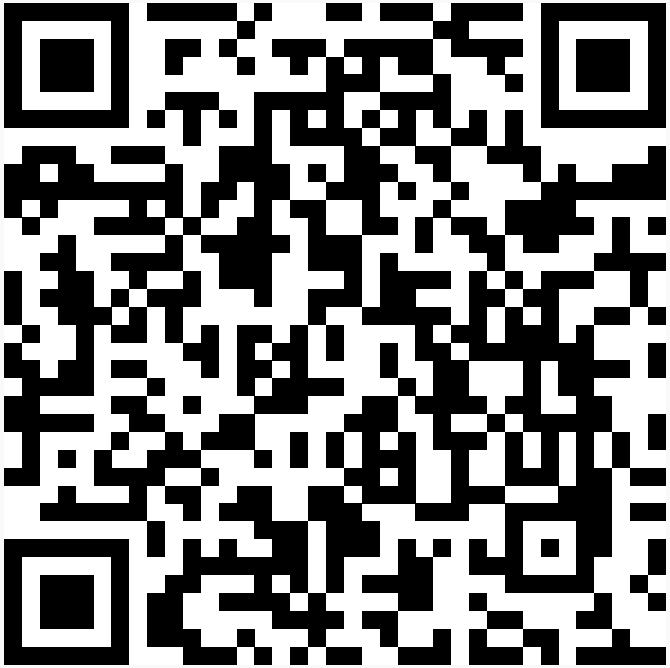
- generates C headers for Rust libraries
- looks for:
  - `#[no_mangle] pub extern fn` (functions)
  - `#[no_mangle] pub static` (globals)
  - `pub const` (constants)
- declares those items and all types used by them recursively
- User Guide

Example: build.rs

```
fn main() {
    let crate_dir = env::var("CARGO_MANIFEST_DIR").unwrap();

    cbindgen::Builder::new()
      .with_crate(crate_dir)
      .generate()
      .expect("Unable to generate bindings")
      .write_to_file("bindings.h");
}
```

Questions?



https://babelmonkeys.de/~florob/talks/RC-2025-05-07-bindings.pdf

Thank you for your attention. Any questions?