Rust's cargo

Special Topic

Lukas Wais

CODERS.BAY

Version: 30. November 2023

Inhaltsverzeichnis I

Intro to Rust

Dependency Trees

Application Binary Interface

.lock Files

build.rs

Auditing and Inspection

Appendix

Intro to Rust

Hello Rust World

```rust
fn main() {
    println!("{}", convert_hex("Supply Chain Security"));
}

fn convert_hex(input: &str) -> String {
    return const_hex::encode_upper(input.as_bytes());
}
```

Crates

A crate is a compilation unit. A crate can be compiled into a binary or into a library. By default, rustc will produce a binary from a crate.

Cargo.toml

Tom's Obvious Minimal Language
Cargo is the build system of Rust. For configuration the Cargo.toml file is used.

Cargo.toml

Tom's Obvious Minimal Language

Cargo is the build system of Rust. For configuration the Cargo.toml file is used.

```
[package]
name = "hex_converter"
version = "0.1.0"
edition = "2021"


[dependencies]
const-hex = "1.10.0"
```

Dependency Trees

Dependency Trees

Can be displayed with the cargo tree command.

Dependency Trees

Can be displayed with the cargo tree command.

- Rust has a small standard library.
- A lot of functionality is in external crates.
- Popular crates depend only on micro-crates and sub-crates, thus the illusion of a huge dependency tree.
  - In Tokio for example a lot of dependencies are optional and barred behind additive compilation features.
- Smaller ones with less community support forget to turn off features.

Dependency Trees

Can be displayed with the cargo tree command.

- Rust has a small standard library.
- A lot of functionality is in external crates.
- Popular crates depend only on micro-crates and sub-crates, thus the illusion of a huge dependency tree.
  - In Tokio for example a lot of dependencies are optional and barred behind additive compilation features.
- Smaller ones with less community support forget to turn off features.
- Use lightweight sub-crate

cargo tree output

```
hex_converter v0.1.0 (/home/.../cargo/example/hex_converter)
   const-hex v1.10.0
       cfg-if v1.0.0
       cpufeatures v0.2.11
```

Application Binary Interface

Application Binary Interface (ABI) I

- A stable ABI enables dynamic linking between Rust crates, which would allow for Rust programs to support
  - Support for dynamically loaded plugins (a feature common in C/C++).
  - Because of dynamic linking, short compile times.
  - Lower disk-space use for projects, as multiple projects could link to the same dynamic library.
- abi-stable-crate allows linking.

- It would allow Rust libraries to be loaded by other languages, such as Swift, which has an ABI.
- Interop with libraries defined in other programming languages.
- Non-Rust crates could be integrated with Rust toolchains.
- Cross-language compatibility would increase the diversity of Rust's package ecosystem.

- It enables swapping out binaries, without recompilation.
  - Security updates without recompiling.

.lock Files

Example

```
# This file is automatically @generated by Cargo.
# It is not intended for manual editing.
version = 3

[[package]]
name = "autocfg"
version = "1.1.0"
source = "registry+https://github.com/rust-lang/crates.io-index"
checksum = "d468802bab17cbc0cc575e9b053f41e72aa36bfa6b7f55e...."

....
```

Changing It

With the command cargo update the .lock file gets regenerated.

build.rs

build.rs

- Build script written in Rust lang.
- Used to build a native library before the Rust crate itself.
- Manifest in cargo.toml

Example of a build.rs

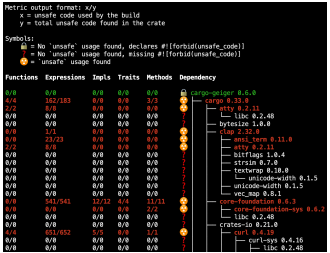
```rust
fn main() {
    // if the given file changes rerun this build script.
    println!("cargo:rerun-if-changed=src/hello.c");

    // cc crate to build a C file and statically link it.
    cc::Build::new()
        .file("src/hello.c")
        .compile("hello");
}
```

Auditing and Inspection

Some Usefull Commands

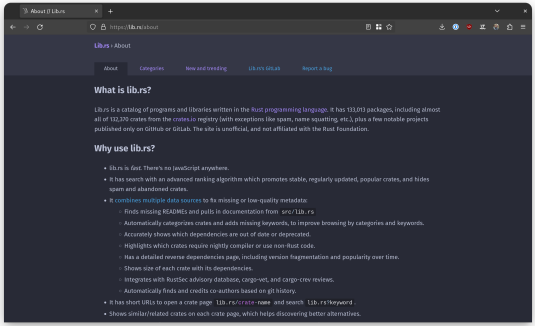
- cargo-audit for checking against rustsec advisories.
- cargo-geiger lists statistics related to the usage of unsafe Rust code in a crate and all its dependencies.



Example output of cargo-geiger. Source: github.com.

lib.rs

libr.rs replaces crates.io. But not crates.io.



Key Takeaways

- Use smaller sub-crates.
- ABI is nice to have.
- Be very cautious with build.rs files.
- Use provided tools and integrate them.
- Check out lib.rs.

Appendix

More Commands to try out yourself

- cargo-outdated; compare .lock file to what is available
- cargo deny; fetching crates only from trusted sources and blacklisting crates.

Supply Chain Security Features from GitHub

- GitHub Advisory Database
- Dependabot
- GitHub blog entry