Test generator & managing resource dependencies

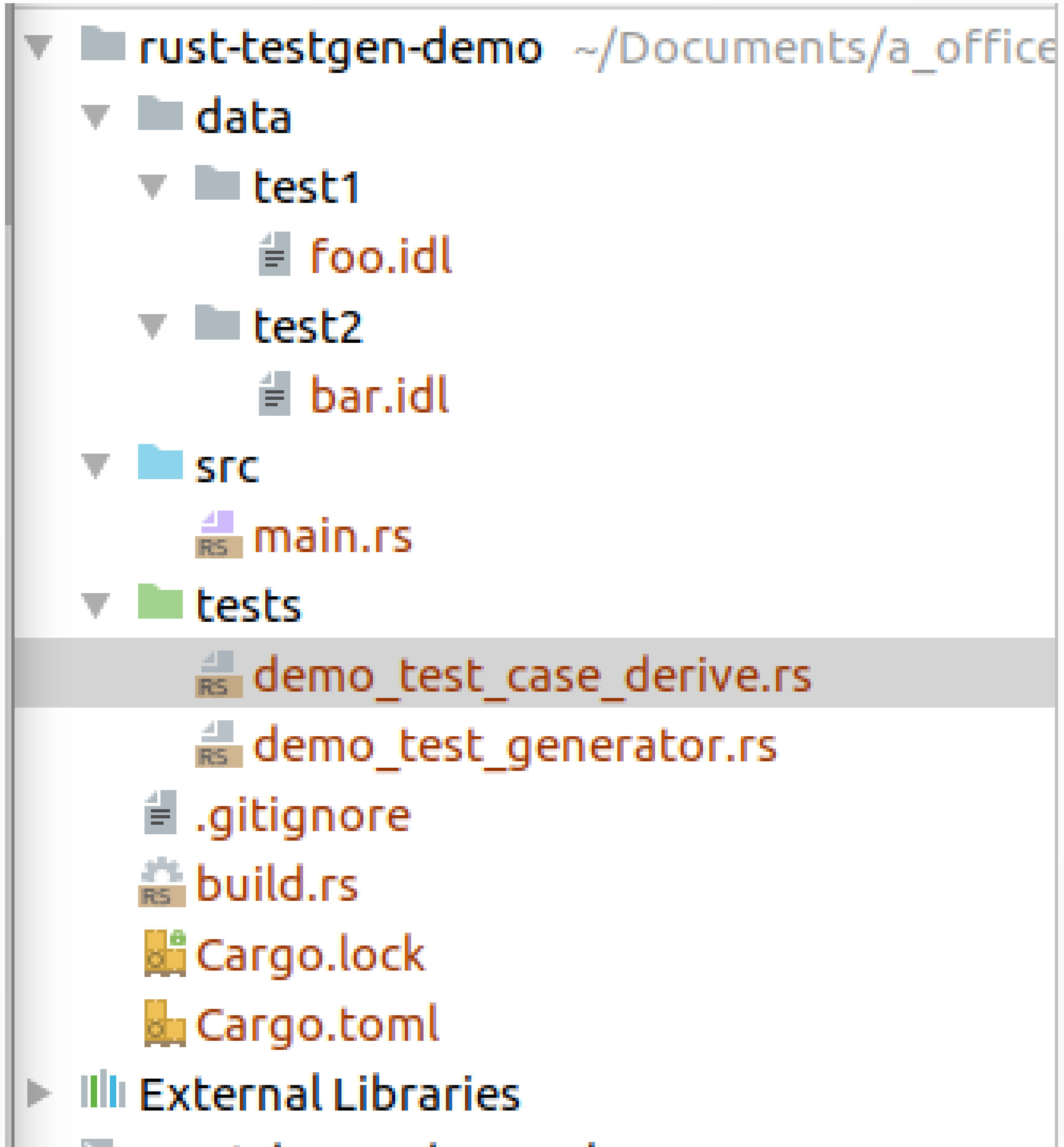
Frank Rehberger
fr@frehberg.com
Software/Security & Rust enthusiast

Use test-generators for...

- Creation of many and many different test functions
- Based on Proc-Macros generating multiple parameterized tests

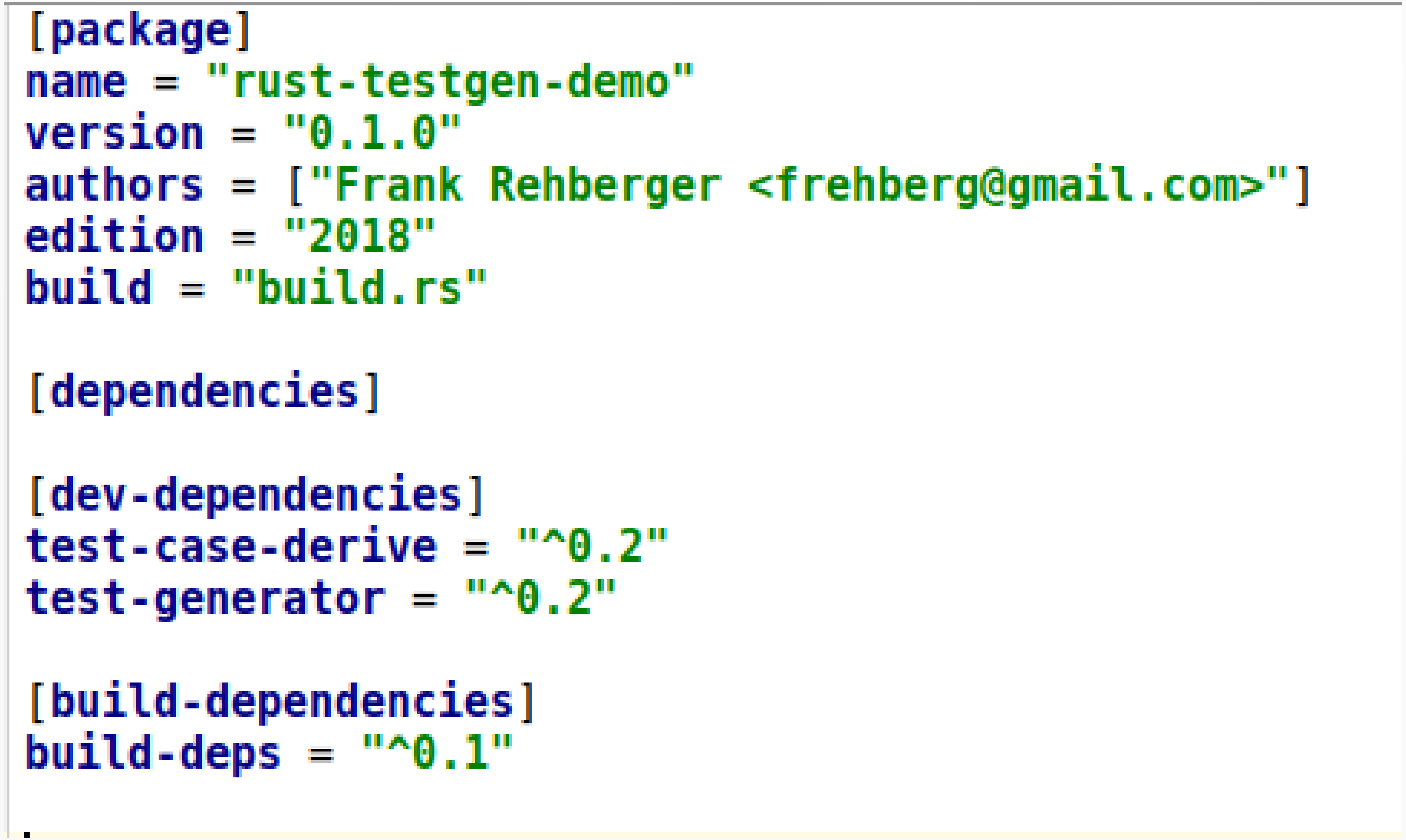
- **Crate test-case-derive**: one body with different input parameter values
- **Crate test-generator**: one body with different input files
- **Crate build-deps**: conditional rebuild 3party source-files

Package Layout



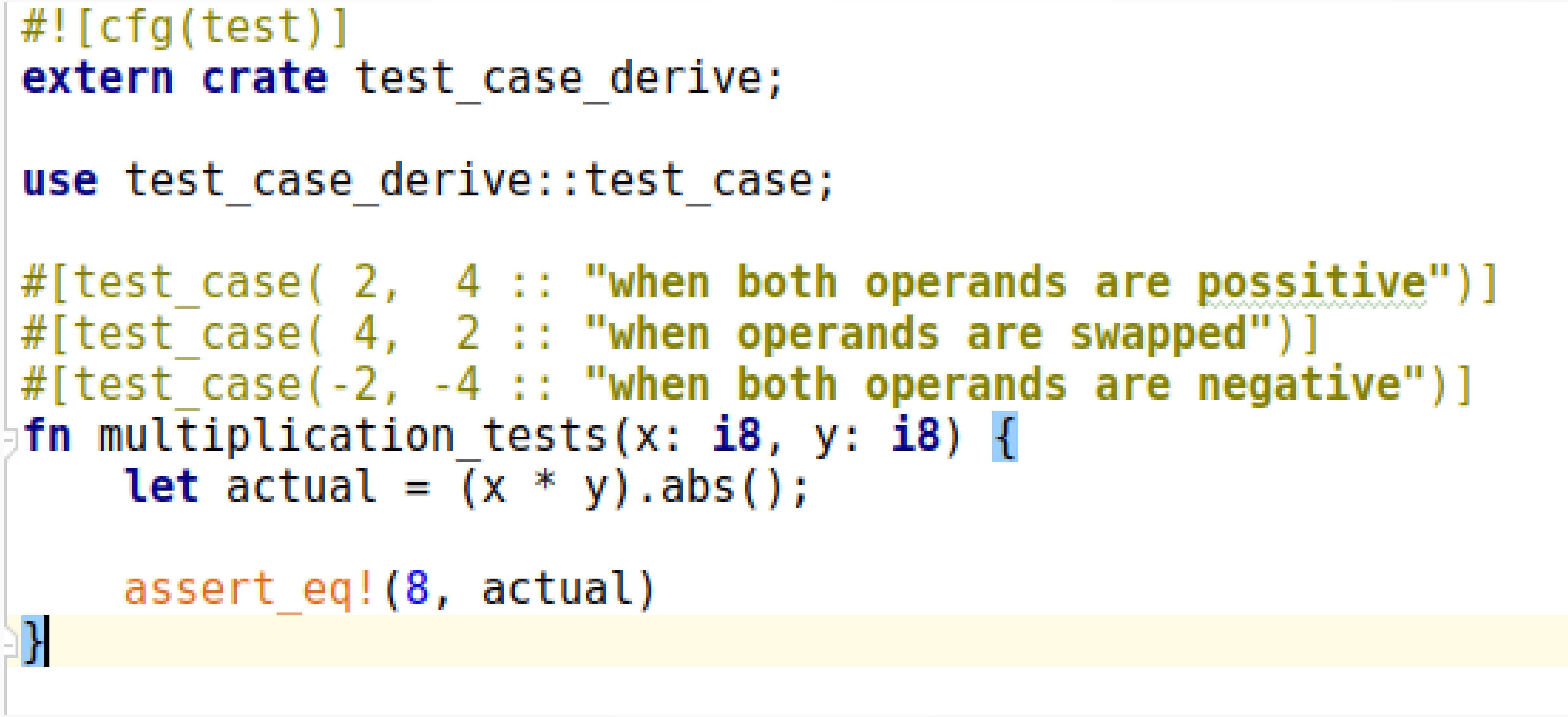
```
rust-testgen-demo  ~/Documents/a_office
    data
        test1
            foo.idl
        test2
            bar.idl
    src
        main.rs
    tests
        demo_test_case_derive.rs
        demo_test_generator.rs
    .gitignore
    build.rs
    Cargo.lock
    Cargo.toml
External Libraries
```

Package Manifest Cargo.toml



```toml
[package]
name = "rust-testgen-demo"
version = "0.1.0"
authors = ["Frank Rehberger <frehberg@gmail.com>"]
edition = "2018"
build = "build.rs"

[dependencies]

[dev-dependencies]
test-case-derive = "^0.2"
test-generator = "^0.2"

[build-dependencies]
build-deps = "^0.1"
```

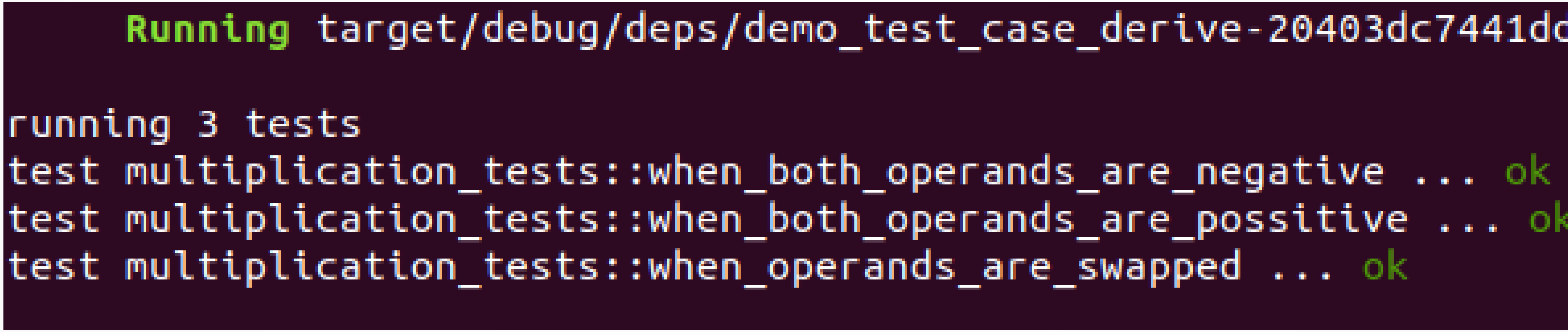
demo_test_case_derive.rs



```
#![cfg(test)]
extern crate test_case_derive;

use test_case_derive::test_case;

#[test_case( 2,  4 :: "when both operands are possitive")]
#[test_case( 4,  2 :: "when operands are swapped")]
#[test_case(-2, -4 :: "when both operands are negative")]
fn multiplication_tests(x: i8, y: i8) {
    let actual = (x * y).abs();

    assert_eq!(8, actual)
}
```

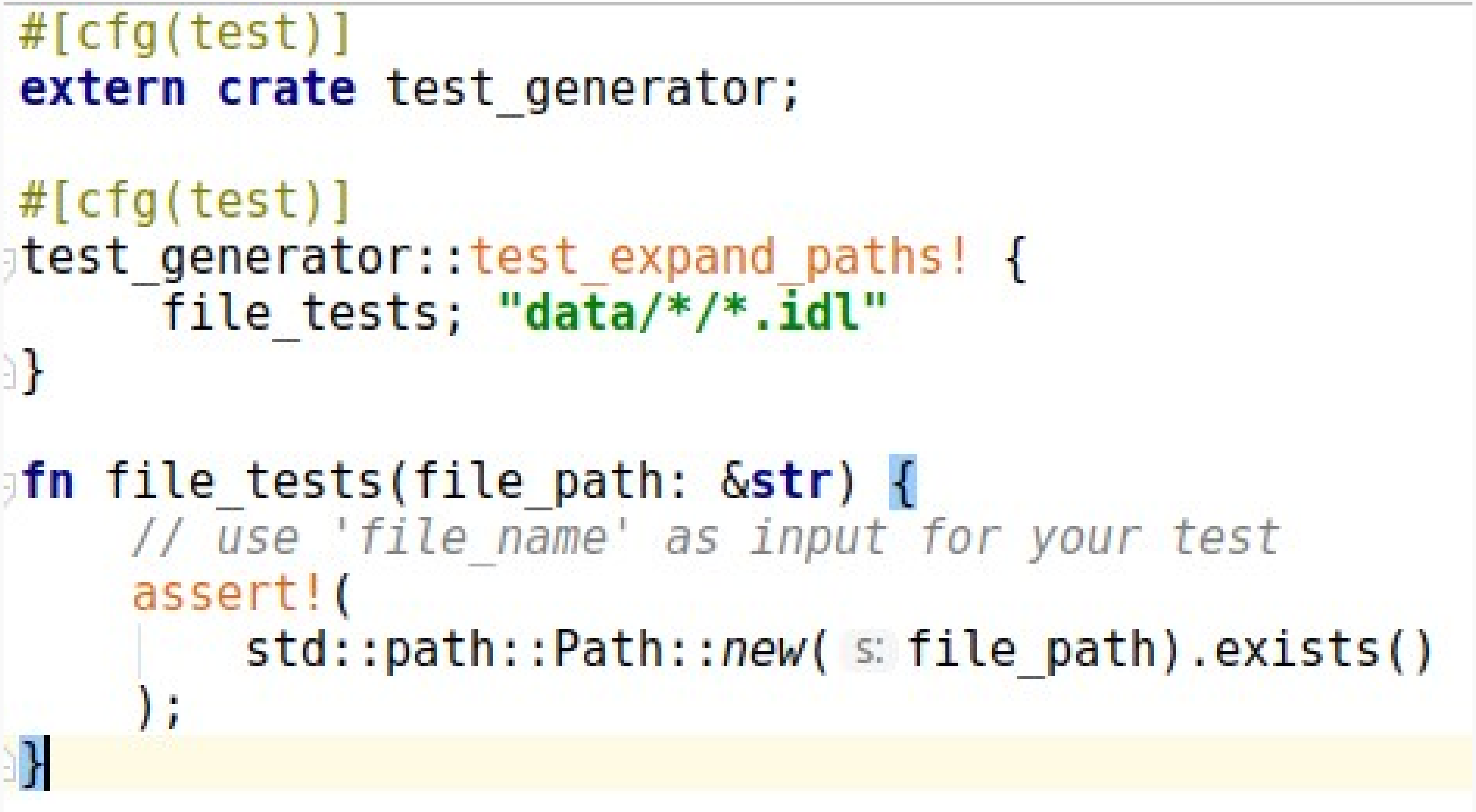
Out: demo_test_case_derive.rs

$ cargo test multiplication_tests



```
     Running target/debug/deps/demo_test_case_derive-20403dc7441dc

running 3 tests
test multiplication_tests::when_both_operands_are_negative ... ok
test multiplication_tests::when_both_operands_are_possitive ... ok
test multiplication_tests::when_operands_are_swapped ... ok
```

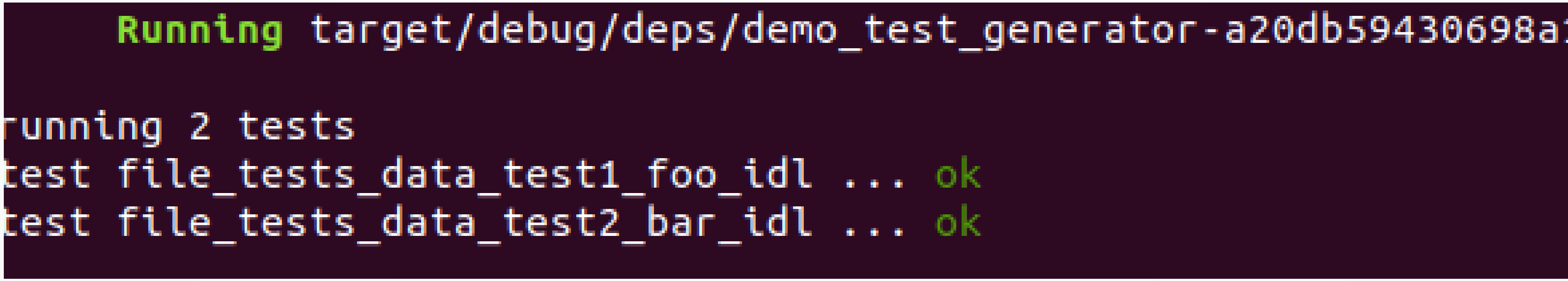
demo_test_generator.rs



```
#[cfg(test)]
extern crate test_generator;

#[cfg(test)]
test_generator::test_expand_paths! {
    file_tests; "data/*/*.idl"
}

fn file_tests(file_path: &str) {
    // use 'file_name' as input for your test
    assert!(
        std::path::Path::new( s: file_path).exists()
    );
}
```

Out: demo_test_generator.rs

```
$ cargo   test   file_tests
```



```
     Running target/debug/deps/demo_test_generator-a20db59430698a

running 2 tests
test file_tests_data_test1_foo_idl ... ok
test file_tests_data_test2_bar_idl ... ok
```

- Enumeration defined at compilation-time
- Adding a new file, does not trigger re-compilation

  ==> Need for dependency management

Cargo/Rustc Dependency Management

Compilation rerun if changed:

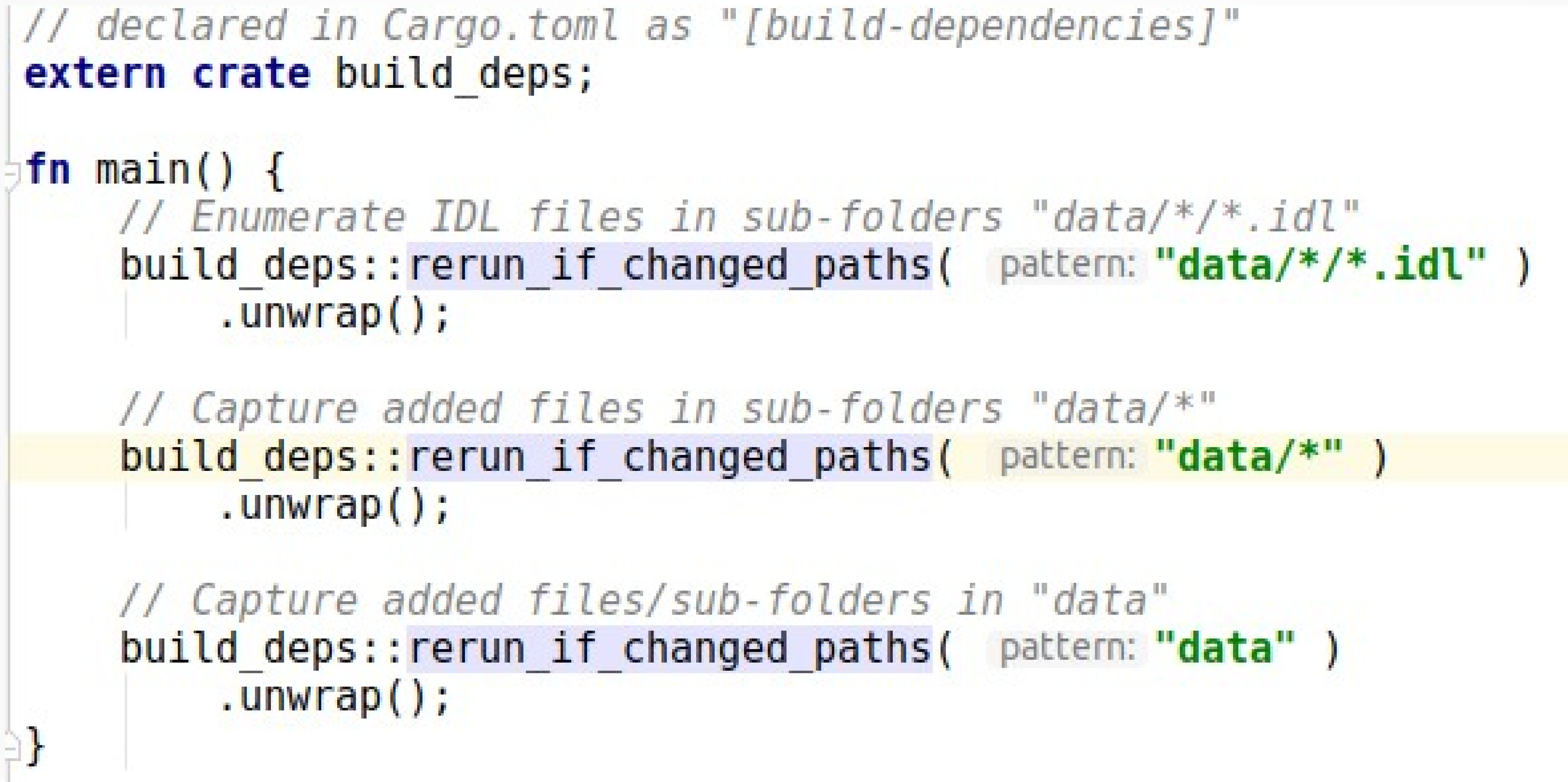
- Rs-files
- External crates
- Include! macro sourced files
- **Build-Script dependencies**
  - **Source/Header-files, etc.**
  - **Other files**

Build-Script

- Commands printed to stdout are interpreted by cargo
  - cargo:rustc-link-lib=static=foo
  - cargo:rustc-link-search=native=/path/to/foo
  - cargo:rustc-env=FOO=bar
  - ...
  - **cargo:rerun-if-changed=PATH**

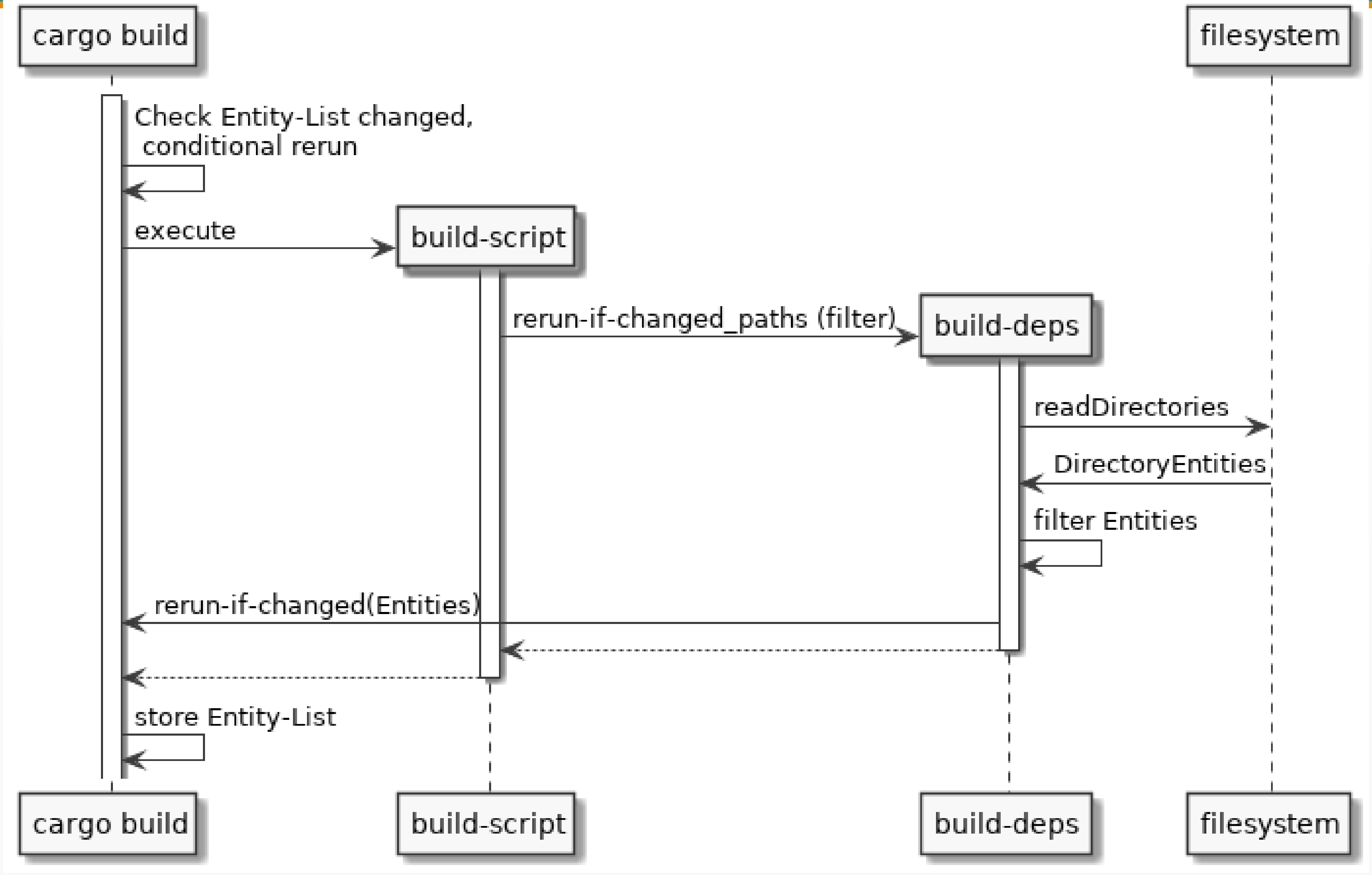
==> Combine with GLOB in crate build-deps

build.rs



```rust
// declared in Cargo.toml as "[build-dependencies]"
extern crate build_deps;

fn main() {
    // Enumerate IDL files in sub-folders "data/*/*.idl"
    build_deps::rerun_if_changed_paths( pattern: "data/*/*.idl" )
        .unwrap();

    // Capture added files in sub-folders "data/*"
    build_deps::rerun_if_changed_paths( pattern: "data/*" )
        .unwrap();

    // Capture added files/sub-folders in "data"
    build_deps::rerun_if_changed_paths( pattern: "data" )
        .unwrap();
}
```

build-deps


cargo build
filesystem
Check Entity-List changed,
conditional rerun
execute
build-script
rerun-if-changed_paths (filter)
build-deps
readDirectories
DirectoryEntities
filter Entities
rerun-if-changed(Entities)
store Entity-List
cargo build
build-script
build-deps
filesystem

Achievement

- Proc-Macros generating parameterized tests over
  - VALUES
  - FILES-ENTITIES
- Enforce Rerun compilation depending on
  - Adding FILE-ENTITIES
  - Modifying FILE-ENTITIES
  - Deleting FILE-ENTITIES

References

- https://crates.io/crates/test-case-derive
- https://crates.io/crates/test-generator
- https://crates.io/crates/build-deps
- https://github.com/frehberg/rust-testgen-demo

- THANK YOU!