

Fixing bugs once and for all



Preparations

- rustup update nightly
- git checkout https://github.com/oli-obk/rustfest2018_workshop.git
- (oli-obk/rustfest2018_workshop)
- cd rustfest2018_workshop
- rustup override set nightly
- cargo test
- Wifi: Ionis Portal
- user: event-epitech-prs
- pw: Pr\$3p1t3ch

\$whoami

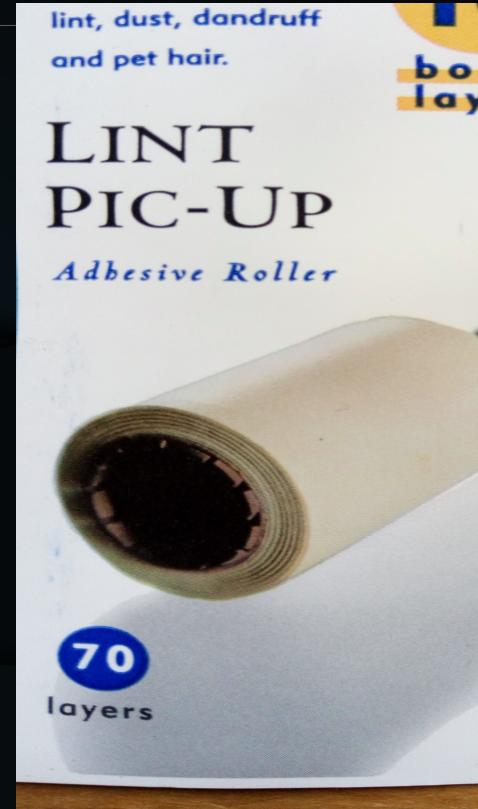
- Clippy
- Miri
- Serde

Ftp

- Wifi: Ionis Portal
- user: event-epitech-prs
- pw: Pr\$3p1t3ch

What is clippy?

- A linter
- A static analysis tool
- Very annoying
- Very helpful



Static analysis

The process of obtaining information about the runtime behaviour of your program without actually running your program

- Aka error messages

That's great...
everything already exists...

So you may be asking yourself:
...why am I here?

Project specific static analyses

- Servo
 - Garbage collection is hard
 - Encoding it in the type system produces e.g.

```
error: no method named `then` found for type `futures::AndThen<std::boxed::Box<futures::Future<Error=std::io::Error, Item=std::boxed::Box<futures::Stream<Error=std::io::Error, Item=()> + std::marker::Send>> + std::marker::Send>, std::boxed::Box<futures::Stream<Error=std::io::Error, Item=()> + std::marker::Send>, [closure@src/main.rs:99:19: 99:66]>` in the current scope
--> src/main.rs:100:10
100 |         .then(|_| ());
    |         ~~~~~
= note: the method `then` exists but the following trait bounds were not satisfied: `std::boxed::Box<futures::Stream<Error=std::io::Error, Item=()> + std::marker::Send> : futures::IntoFuture`, `futures::AndThen<std::boxed::Box<futures::Future<Error=std::io::Error, Item=std::boxed::Box<futures::Stream<Error=std::io::Error, Item=()> + std::marker::Send>> + std::marker::Send>, std::boxed::Box<futures::Stream<Error=std::io::Error, Item=()> + std::marker::Send>, [closure@src/main.rs:99:19: 99:66]> : futures::Future`
```


How does it all work?

- Step 1
 - Create a new rustc binary that has more lints than the vanilla one
- Step 2
 - Use that rustc binary instead of the vanilla one by setting `RUSTC_WRAPPER=my_awesome_rustc`



Fin

Just kidding – Let's do this

- There are 150 lines of boilerplate
 - (without code for even a single lint)
 - I abstracted them away for you
- Now it's three lines of code:

```
rustfest2018_workshop::run_lints(|ls| {  
    ls.register_early_pass(None, false, box NoTransmute);  
});
```

early_pass? What's going on?

Two kinds of lints (actually 3, but not today)

- EarlyLintPass
 - Syntactic lints, no types available
- LateLintPass
 - Lowered Datastructures (no for/while, just loop+if)
 - Types!

Declaring a lint

- declareLint! macro
- 3 arguments:
 - pub LINT_NAME
 - Lint level (Allow, Warn, Deny, Forbid)
 - A short description

Lint structure

- `pub struct HelperType;`
- `impl LintPass for HelperType { ... }`
- choose one:
 - `impl EarlyLintPass for HelperType { ... }`
 - `impl LateLintPass for HelperType { ... }`

LintPass Boilerplate

```
fn get_lints(&self) -> LintArray {  
    lint_array!(LINT_NAME)  
}
```

Documentation

- forge.rust-lang.org
- „The rustc API docs are hosted here“
- Search for „EarlyLintPass“
- Big list of methods!
 - Implement only those you want to run checks for