

Testing My Patience

In the beginning...

```
#[test]
fn some_case() {
    assert_eq!(1, 2);
}
```

```
awesomeness-rs/
Cargo.toml
src/           # whitebox tests go here
  lib.rs
  submodule.rs
  submodule/
    tests.rs
tests/         # blackbox tests go here
  is_awesome.rs
```

Pain points

Custom test harnesses

- `cargo-test-macro`
- `trybuild`
- `trycmd`
- `toml-test-rs`
- `criterion`

Holy envy

Stendahl's three rules of religious understanding:

1. When you are trying to understand another religion, you should ask the adherents of that religion and not its enemies.
2. Don't compare your best to their worst.
3. **Leave room for "holy envy."**

Holy envy

```
def pytest_addoption(parser):  
    parser.addoption(  
        "--can-in-interface", default="None",  
        action="store",  
        help="The CAN interface to use with the tests")  
  
@pytest.fixture  
def can_in_interface(request):  
    interface = request.config.getoption("--can-in-interface")  
    if interface.lower() == "none":  
        pytest.skip("Test requires a CAN board")  
    return interface  
  
def test_wait_for_intf_communicating(can_in_interface):  
    # ...
```

Pain point: conditional ignores

```
#[test]
fn simple_hg() {

    // ...

}
```

```
$ cargo test
  Compiling cargo v0.72.0
  Finished test [unoptimized + debuginfo] target(s) in 0.62s
  Running /home/epage/src/cargo/tests/testsuites/main.rs

running 1 test
test new::simple_hg ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Pain point: conditional ignores

```
#[test]
fn simple_hg() {
    if !has_command("hg") {
        return;
    }

    // ...
}
```

```
$ cargo test
Compiling cargo v0.72.0
Finished test [unoptimized + debuginfo] target(s) in 0.62s
Running /home/epage/src/cargo/tests/testsuites/main.rs

running 1 test
test new::simple_hg ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Pain point: conditional ignores

```
#[test]
fn simple_hg() {
    if !has_command("hg") {
        return;
    }

    // ...
}
```

```
@pytest.mark.skipif(!has_command("hg"), reason="requires `hg` CLI")
def test_simple_hg():
    pass
```


Pain point: lack of fixtures

```
fn cargo_add_lockfile_updated() {  
    let scratch = tempfile::tempdir::new().unwrap();  
  
    // ...  
  
}
```

Pain point: lack of fixtures

```
fn cargo_add_lockfile_updated() {  
    let scratch = tempfile::tempdir::new().unwrap();  
  
    // ...  
  
    scratch.close().unwrap();  
}
```

Pain point: lack of fixtures

```
fn cargo_add_lockfile_updated() {  
    let scratch = tempfile::tempdir::new().unwrap();  
  
    // ...  
  
    scratch.close().unwrap();  
}
```

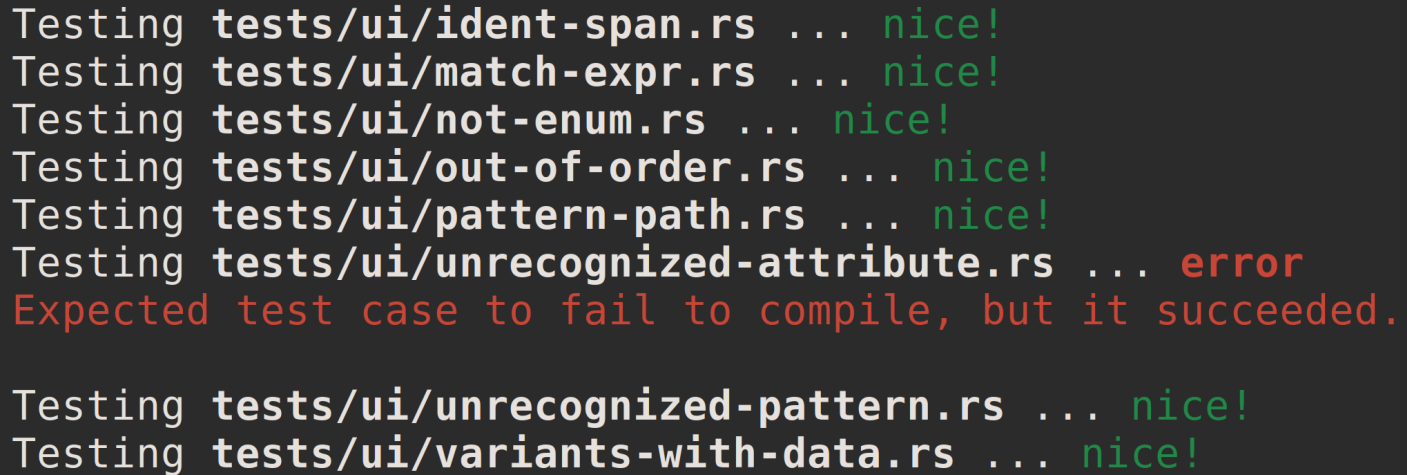
```
def cargo_add_lockfile_updated(tmpdir):  
    # ...
```

Pain point: lack of test generation

```
#[test]
fn integers() {
    let cases = [
        ("+99", 99),
        ("42", 42),
        ("0", 0),
        ("-17", -17),
        ("1_2_3_4_5", 1_2_3_4_5),
        ("0xF", 15),
        ("0o0_755", 493),
        ("0b1_0_1", 5),
        (&std::i64::MIN.to_string()[..], std::i64::MIN),
        (&std::i64::MAX.to_string()[..], std::i64::MAX),
    ];
    for &(input, expected) in &cases {
        let parsed = integer.parse(new_input(input));
        assert_eq!(parsed, Ok(expected));
    }
}
```

Pain point: lack of test generation (part 2)

```
#[test]
fn ui() {
    let t = trybuild::TestCases::new();
    t.compile_fail("tests/ui/*.rs");
}
```



```
Testing tests/ui/ident-span.rs ... nice!
Testing tests/ui/match-expr.rs ... nice!
Testing tests/ui/not-enum.rs ... nice!
Testing tests/ui/out-of-order.rs ... nice!
Testing tests/ui/pattern-path.rs ... nice!
Testing tests/ui/unrecognized-attribute.rs ... error
Expected test case to fail to compile, but it succeeded.

Testing tests/ui/unrecognized-pattern.rs ... nice!
Testing tests/ui/variants-with-data.rs ... nice!
```

Pain point: lack of test generation

```
#[test]
fn ui() {
  let t = trybuild::TestCases::new();
  t.compile_fail("tests/ui/*.rs");
}
```

```
@pytest.mark.parametrize("sample_rs", trybuild.find("tests/ui/*.rs"))
def ui(sample_rs):
    trybuild.verify(sample_rs)
```

Pain point: scaling up

```
#[cargo_test(requires_hg)]
fn simple_hg() {
    // ...
}
```

```
#[test]
fn ui() {
    let t = trybuild::TestCases::new();
    t.compile_fail("tests/ui/*.rs");
}
```

```
#[test]
fn cli_tests() {
    trycmd::TestCases::new()
        .case("tests/cmd/*.toml")
        .case("README.md");
}
```

Path forward

Milestones

1. libtest2-mimic ←we are here
2. libtest2
3. pytest
4. criterion, trybuild, etc
5. Merge libtest2 into libtest?

Help?

Questions

- What do you have "holy envy" for?
- What odd ball test scenarios have I not seen yet?

Get involved: github.com/epage/pytest-rs

Offload my work :)