`profile`

The tracking issue for this feature is: [#42524](#).

This feature allows the generation of code coverage reports.

Set the `-Zprofile` compiler flag in order to enable gcov profiling.

For example:

```
cargo new testgcov --bin
cd testgcov
export RUSTFLAGS="-Zprofile -Ccodegen-units=1 -Copt-level=0 -Clink-dead-code -
Coverflow-checks=off -Zpanic_abort_tests -Cpanic=abort"
export CARGO_INCREMENTAL=0
cargo build
cargo run
```

Once you've built and run your program, files with the `gcno` (after build) and `gcda` (after execution) extensions will be created. You can parse them with [llvm-cov gcov](#) or [grcov](#).

Please note that `RUSTFLAGS` by default applies to everything that cargo builds and runs during a build! When the `--target` flag is explicitly passed to cargo, the `RUSTFLAGS` no longer apply to build scripts and procedural macros. For more fine-grained control consider passing a `RUSTC_WRAPPER` program to cargo that only adds the profiling flags to rustc for the specific crates you want to profile.