rustfmt Rust code formatter

START HERE

`rustup component add rustfmt`
Install it

`rustfmt --print-config default rustfmt.toml`
Write a config

`cargo fmt`
Format the whole crate

`cargo +nightly fmt`
Use nightly options

`cargo fmt --check`
Verify only (CI)

`cargo fmt --all`
Whole workspace

`rustfmt <file.rs>`
Format one file

`rustfmt --edition 2021 <file.rs>`
Set parse edition

Source files *.rs → `parse (edition)` → **rustfmt** rustfmt.toml + style_edition → `emit` → **Result** files, stdout or diff

CONCEPTS

cargo fmt reads the edition from Cargo.toml; bare rustfmt assumes 2015 unless told otherwise.

edition controls parsing; style_edition controls the formatting rules. Set both in rustfmt.toml.

Unstable options need nightly plus unstable_features = true; stable rustfmt ignores them with a warning.

INSTALL AND RUN

`rustup component add rustfmt`
Install on the stable toolchain

`rustup component add rustfmt --toolchain nightly`
Install on nightly

`cargo fmt`
Format all targets of the crate

`cargo fmt --all`
Format every workspace member

`cargo fmt -p <crate>`
Format one package

`cargo fmt -- <rustfmt-args>`
Pass flags through to rustfmt

INSTALL AND RUN (cont.)

`cargo +nightly fmt`
Run the nightly formatter

`rustfmt <file.rs>`
Format a file and its out-of-line modules

`rustfmt <a.rs> <b.rs>`
Format several files in place

`echo 'fn main(){}' | rustfmt`
Format stdin to stdout

`rustfmt --version`
Show the rustfmt version

`rustfmt --help`
List all flags

`rustfmt --help=config`
List config options and defaults

CHECKING AND OUTPUT

`cargo fmt --check`
Exit 1 and print a diff if unformatted

`cargo fmt --all -- --check`
Check the whole workspace in CI

`rustfmt --check <file.rs>`
Check one file without writing

`rustfmt --emit stdout <file.rs>`
Print the result instead of writing

CHECKING AND OUTPUT (cont.)

`rustfmt --emit files <file.rs>`
Overwrite files (the default)

`rustfmt -l <file.rs>`
List files that would change

Exit codes: --check returns 1 on needed changes; other modes return 1 only on errors (parse failure).

CONFIGURATION

`rustfmt.toml / .rustfmt.toml`
Config file in the project or any parent dir

`rustfmt --print-config default rustfmt.toml`
Write all defaults to a file

`rustfmt --print-config current <file.rs>`
Show the effective config for a file

`rustfmt --print-config minimal rustfmt.toml`
Write only non-default values

`rustfmt --config-path <dir|file> <file.rs>`
Use a specific config

`rustfmt --config max_width=80 <file.rs>`
Override an option from the CLI

CONFIGURATION (cont.)

`cargo fmt -- --config max_width=80,tab_spaces=2`
Override options via cargo

`edition = "2021"`
Parse with this language edition

`style_edition = "2024"`
Pick the style guide edition

`rustfmt --edition 2021 <file.rs>`
Set the edition on the command line

`rustfmt --style-edition 2024 <file.rs>`
Set the style edition on the CLI

Set edition and style_edition in rustfmt.toml so cargo fmt and plain rustfmt agree.

COMMON OPTIONS (STABLE)

`max_width = 100`
Maximum line width

`tab_spaces = 4`
Spaces per indent level

`hard_tabs = false`
Indent with tabs instead of spaces

`newline_style = "Auto"`
Line endings: Auto, Native, Unix, Windows

COMMON OPTIONS (STABLE) (cont.)

`use_small_heuristics = "Default"`
Scale small-width limits: Off, Max, Default

`match_block_trailing_comma`
Comma after block match arms

`merge_derives = true`
Merge adjacent #[derive] attributes

`reorder_imports = true`
Sort use statements within a group

`reorder_modules = true`
Sort mod declarations

NIGHTLY-ONLY OPTIONS

`unstable_features = true`
Opt in to unstable options

`imports_granularity = "Crate"`
Merge imports: Preserve, Crate, Module, Item, One

`group_imports = "StdExternalCrate"`
Group imports: Preserve, StdExternalCrate, One

`wrap_comments = true`
Re-wrap long comments

`comment_width = 80`
Width used when wrapping comments

NIGHTLY-ONLY OPTIONS (cont.)

`format_code_in_doc_comments`
Format Rust code blocks in docs

`format_strings = true`
Break long string literals

`ignore = ["<path>"]`
Skip files/dirs (gitignore patterns)

Only some options are stable; the rest print a warning and are skipped on a stable toolchain.

SKIPPING AND ATTRIBUTES

`#[rustfmt::skip]`
Do not format the next item or statement

`#![rustfmt::skip]`
Skip the whole module or file (inner attr)

#[rustfmt::skip] is handy for hand-aligned tables and matrices that you want to keep.

RECIPES

CI format check

```
cargo fmt --all -- --check
cargo clippy --all-targets -- -D warnings
```