Clang tools Format, lint, analyze C/C++

CONCEPTS

> Each tool reads its own YAML file from the nearest parent directory: .clang-format, .clang-tidy, .clangd.

CMake, Meson
↓ writes
compile_commands.json
↓ -p build
clang-tidy, clangd
↓ report
warnings + fixes

One compile database feeds the analysis tools

CLANG-FORMAT

clang-format *<file>*
Print the formatted file to stdout

clang-format -i *<a.cpp> <b.h>*
Format files in place

clang-format --style=google -i *<file>*
Use a preset style

clang-format --dry-run --Werror *<file>*
Fail if a file is not formatted

STYLE FILES

clang-format --style=llvm --dump-config
Print every option of a preset

.clang-format

```
BasedOnStyle: LLVM
IndentWidth: 4
ColumnLimit: 100
---
Language: Proto
DisableFormat: true
```

// clang-format off
Stop formatting until … on

FORMAT ONLY CHANGES

git clang-format
Format lines changed vs HEAD

FORMAT ONLY CHANGES (cont.)

git clang-format *<branch>*
Format all changes since <branch>

CLANG-TIDY

clang-tidy -p *<build> <file>*
Lint with flags from compile_commands

clang-tidy *<file>* **--** *<compiler flags>*
Lint without a compile database

clang-tidy --checks='-*,bugprone-*' *<file>*
Pick checks by glob

clang-tidy --fix -p *<build> <file>*
Apply the fixes

CONFIGURE AND SILENCE TIDY

// NOLINT
Silence findings on this line

// NOLINT(*<check>***)**
Silence one check on this line

CLANG-ONLY WARNINGS

clang -fsyntax-only *<file>*
Check code, produce no output

STATIC ANALYZER

scan-build make
Analyze a whole build

COMPILE-TIME INSIGHT

clang -ftime-trace -c *<file>*
Write a Chrome trace of the compile

TOOLCHAIN CHOICES

CC=clang CXX=clang++ cmake -B build
Build a CMake project with clang