# 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

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

**git clang-format --diff**
Print the diff, change nothing

## 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

**clang-tidy --list-checks --checks='*'**
List every available check

**clang-tidy --dump-config**
Print the effective config

**clang-tidy --warnings-as-errors='*'** *<file>*
Fail CI on any finding

**run-clang-tidy -p** *<build>* **-j 8**
Lint the whole database in parallel

## CHECK GROUPS

| Prefix | Looks for |
|---|---|
| `bugprone-` | Likely bugs |
| `modernize-` | Newer idioms |
| `performance-` | Wasted copies |
| `readability-` | Clarity |
| `cppcoreguidelines-` | Guidelines |
| `clang-analyzer-` | Analyzer |

## CONFIGURE AND SILENCE TIDY

**.clang-tidy**

```
Checks: '-*,bugprone-*,performance-*'
WarningsAsErrors: 'bugprone-*'
HeaderFilterRegex: 'src/.*'
CheckOptions:
  <check>.<Option>: <value>
```

**// NOLINT**
Silence findings on this line

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

**// NOLINTNEXTLINE(***<check>***)**
Silence the next line

## CLANGD

clangd is the C/C++ language server most editors use. It finds compile_commands.json in a parent dir or its build/ subdir.

**.clangd**

```
CompileFlags:
  CompilationDatabase: build
  Add: [-Wall]
  Remove: [-fno-*]
Diagnostics:
  UnusedIncludes: Strict
  ClangTidy:
    Add: [bugprone-*]
```

## CLANG-ONLY WARNINGS

**clang -Weverything** *<file>*
Every warning clang knows

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

## SANITIZERS AND COVERAGE

**clang -fsanitize=memory -g** *<file>*
MemorySanitizer: uninitialized reads

**clang -fprofile-instr-generate -fcoverage-mapping** *<file>*
Source-based coverage build

## STATIC ANALYZER

**clang --analyze** *<file>*
Analyze one file

**scan-build make**
Analyze a whole build

**scan-build -o** *<dir>* **-V make**
HTML report, open in browser

## 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