clang-format C/C++ code formatter

START HERE

`clang-format -i <file>`
Format a file in place

`clang-format --style=file <file>`
Print with project style

`clang-format --dry-run -Werror <file>`
Check only, fail if unformatted

`clang-format --style=Google -i <file>`
Use a preset style

`clang-format --dump-config`
Show the active settings

`clang-format -style=llvm --dump-config > .clang-format`
Start a config file

`git clang-format`
Format staged changes only

`// clang-format off`
Protect a region

Source file or stdin → **`find config`** → **.clang-format** nearest parent dir → **`apply`** → **Formatted code** stdout or in place

CONCEPTS

Put a .clang-format file in the repo root and use --style=file: every tool and editor then agrees on one style.

Without a .clang-format file, --style=file falls back to LLVM style; change that with --fallback-style.

Formatting only rewrites whitespace and line breaks (plus optional include sorting), not what the code means.

RUN

`clang-format <file>`
Print formatted code to stdout

`clang-format -i <file>...`
Rewrite files in place

`clang-format --style=<style> <file>`
Pick a preset or file

`clang-format --version`
Show the version

`clang-format -i src/*.cpp include/*.h`
Format many files

`clang-format --lines=<start>:<end> -i <file>`
Format only a line range

RUN (cont.)

`clang-format --lines=10:20 --lines=40:50 <file>`
Several ranges at once

`clang-format --verbose -i <file>...`
List files as they are processed

`clang-format --files=<list> -i`
Read file names from a list

`<cmd> | clang-format --assume-filename=<name>`
Format stdin as that file type

`clang-format --offset=<n> --length=<n> <file>`
Format a byte range

CHECK (CI)

`clang-format --dry-run <file>`
Report violations, change nothing

`clang-format -n -Werror <file>`
Exit non-zero if unformatted

`clang-format --dry-run -Werror --ferror-limit=1 <file>`
Stop at the first problem

`clang-format --fail-on-incomplete-format <file>`
Exit 1 if code could not be fully formatted

CHECK (CI) (cont.)

`git ls-files '*.c' '*.h' | xargs clang-format -n -Werror`
Check all tracked sources

STYLES & CONFIG

`--style=LLVM`
Preset: LLVM (default)

`--style=Google`
Preset: Google

`--style=Chromium / Mozilla / WebKit`
Other browser presets

`--style=file`
Use nearest .clang-format

`--style=file:<path>`
Use a specific config file

`--style="{BasedOnStyle: Google, IndentWidth: 4}"`
Inline settings

`--fallback-style=<style>`
Style when no config is found

`--fallback-style=none`
Skip formatting without a config

`clang-format -style=<style> --dump-config > .clang-format`
Write a full config file

`clang-format --dump-config`
Show config for the cwd

STYLES & CONFIG (cont.)

`clang-format --dump-config --assume-filename=<f>`
Show config used for a file

.CLANG-FORMAT KEYS

`BasedOnStyle: Google`
Start from a preset

`BasedOnStyle: InheritParentConfig`
Inherit parent directory's file

`IndentWidth: 4`
Spaces per indent level

`ColumnLimit: 100`
Maximum line width (0 = none)

`UseTab: Never`
Tabs: Never, ForIndentation, Always

`BreakBeforeBraces: Allman`
Brace style: Attach, Linux, Allman

`PointerAlignment: Left`
Align * and & left, right or middle

`SortIncludes: CaseSensitive`
Sort #include lines (or Never)

`IncludeBlocks: Regroup`
Merge, Preserve or Regroup blocks

`IndentCaseLabels: true`
Indent case inside switch

.CLANG-FORMAT KEYS (cont.)

`NamespaceIndentation: None`
Indent inside namespaces (None/Inner/All)

`BinPackArguments: false`
One argument per line when wrapping

`Language: Cpp`
Limit a section to one language

GIT INTEGRATION

`git clang-format`
Format staged changes

`git clang-format <commit>`
Format lines changed since a commit

`git clang-format main`
Format all changes on a branch

`git clang-format --diff`
Show the diff, apply nothing

IN CODE & EDITORS

`// clang-format off`
Stop formatting from here

`// clang-format on`
Resume formatting

`Alt+Shift+F`
VS Code: format document
Needs the C/C++ or clang-format extension; macOS uses Option+Shift+F.

IN CODE & EDITORS (cont.)

`clang-format.py`
Vim: source and map a key

CLion picks up a .clang-format file in the project root automatically.

Use `// clang-format off` sparingly, for hand-aligned tables or macros; fix the style first.

RECIPES

Start a project config

```
clang-format -style=Google --
dump-config > .clang-format
# edit IndentWidth,
ColumnLimit ...
clang-format -i src/*.cpp
```

Minimal .clang-format

```
# .clang-format
BasedOnStyle: Google
IndentWidth: 4
ColumnLimit: 100
PointerAlignment: Left
```

CI format check

```
git ls-files '*.c' '*.cpp'
'*.h' \
  | xargs clang-format --dry-
run -Werror
```

Format only what you changed

```
git add -p
git clang-format
git add -u
git commit
```