ClangFormat

ClangFormat describes a set of tools that are built on top of **LibFormat**. It can support your workflow in a variety of ways including a standalone tool and editor integrations.

Standalone Tool

clang-format is located in *clang/tools/clang-format* and can be used to format C/C++/Java/JavaScript/Objective-C/Protobuf/C# code.

```
$ clang-format -help
OVERVIEW: A tool to format C/C++/Java/JavaScript/Objective-C/Protobuf/C# code.

If no arguments are specified, it formats the code from standard input
and writes the result to the standard output.
If <file>s are given, it reformats the files. If -i is specified
together with <file>s, the files are edited in-place. Otherwise, the
result is written to the standard output.

USAGE: clang-format [options] [<file> ...]

OPTIONS:

Clang-format options:

  --Werror                    - If set, changes formatting warnings to errors
  --assume-filename=<string> - Override filename used to determine the language.
                               When reading from stdin, clang-format assumes this
                               filename to determine the language.
  --cursor=<uint>             - The position of the cursor when invoking
                               clang-format from an editor integration
  --dry-run                   - If set, do not actually make the formatting changes
  --dump-config               - Dump configuration options to stdout and exit.
                               Can be used with -style option.
  --fallback-style=<string>  - The name of the predefined style used as a
                               fallback in case clang-format is invoked with
                               -style=file, but can not find the .clang-format
                               file to use.
                               Use -fallback-style=none to skip formatting.
  --ferror-limit=<uint>       - Set the maximum number of clang-format errors to
                               emit before stopping (0 = no limit). Used only
                               with --dry-run or -n
  -i                          - Inplace edit <file>s, if specified.
  --length=<uint>             - Format a range of this length (in bytes).
                               Multiple ranges can be formatted by specifying
                               several -offset and -length pairs.
                               When only a single -offset is specified without
                               -length, clang-format will format up to the end
                               of the file.
                               Can only be used with one input file.
  --lines=<string>            - <start line>:<end line> - format a range of
                               lines (both 1-based).
                               Multiple ranges can be formatted by specifying
                               several -lines arguments.
                               Can't be used with -offset and -length.
                               Can only be used with one input file.
  -n                          - Alias for --dry-run
  --offset=<uint>             - Format a range starting at this byte offset.
                               Multiple ranges can be formatted by specifying
                               several -offset and -length pairs.
                               Can only be used with one input file.
  --output-replacements-xml  - Output replacements as XML.
  --sort-includes             - If set, overrides the include sorting behavior
                               determined by the SortIncludes style flag
  --style=<string>            - Coding style, currently supports:
                                 LLVM, Google, Chromium, Mozilla, WebKit.
                               Use -style=file to load style configuration from
                               .clang-format file located in one of the parent
```

```
                               directories of the source file (or current
                               directory for stdin).
                               Use -style="{key: value, ...}" to set specific
                               parameters, e.g.:
                                 -style="{BasedOnStyle: llvm, IndentWidth: 8}"
  --verbose                  - If set, shows the list of processed files

Generic Options:

  --help                     - Display available options (--help-hidden for more)
  --help-list                - Display list of available options (--help-list-hidden for more)
  --version                  - Display the version of this program
```

When the desired code formatting style is different from the available options, the style can be customized using the `-style="{key: value, ...}"` option or by putting your style configuration in the `.clang-format` or `_clang-format` file in your project's directory and using `clang-format -style=file`.

An easy way to create the `.clang-format` file is:

```
clang-format -style=llvm -dump-config > .clang-format
```

Available style options are described in **Clang-Format Style Options**.

Vim Integration

There is an integration for **vim** which lets you run the **clang-format** standalone tool on your current buffer, optionally selecting regions to reformat. The integration has the form of a *python*-file which can be found under *clang/tools/clang-format/clang-format.py*.

This can be integrated by adding the following to your *.vimrc*:

```
map <C-K> :pyf <path-to-this-file>/clang-format.py<cr>
imap <C-K> <c-o>:pyf <path-to-this-file>/clang-format.py<cr>
```

The first line enables **clang-format** for NORMAL and VISUAL mode, the second line adds support for INSERT mode. Change "C-K" to another binding if you need **clang-format** on a different key (C-K stands for Ctrl+k).

With this integration you can press the bound key and clang-format will format the current line in NORMAL and INSERT mode or the selected region in VISUAL mode. The line or region is extended to the next bigger syntactic entity.

It operates on the current, potentially unsaved buffer and does not create or save any files. To revert a formatting, just undo.

An alternative option is to format changes when saving a file and thus to have a zero-effort integration into the coding workflow. To do this, add this to your *.vimrc*:

```
function! Formatonsave()
  let l:formatdiff = 1
  pyf ~/llvm/tools/clang/tools/clang-format/clang-format.py
endfunction
autocmd BufWritePre *.h,*.cc,*.cpp call Formatonsave()
```

Emacs Integration

Similar to the integration for **vim**, there is an integration for **emacs**. It can be found at *clang/tools/clang-format/clang-format.el* and used by adding this to your *.emacs*:

```
(load "<path-to-clang>/tools/clang-format/clang-format.el")
(global-set-key [C-M-tab] 'clang-format-region)
```

This binds the function *clang-format-region* to C-M-tab, which then formats the current line or selected region.

BBEdit Integration

clang-format cannot be used as a text filter with BBEdit, but works well via a script. The AppleScript to do this integration can be found at *clang/tools/clang-format/clang-format-bbedit.applescript*; place a copy in *~/Library/Application Support/BBEdit/Scripts*, and edit the path within it to point to your local copy of **clang-format**.

With this integration you can select the script from the Script menu and **clang-format** will format the selection. Note that you can rename the menu item by renaming the script, and can assign the menu item a keyboard shortcut in the BBEdit preferences, under Menus & Shortcuts.

CLion Integration

clang-format is integrated into **CLion** as an alternative code formatter. It is disabled by default and can be turned on in Settings/Preferences | Editor | Code Style.

If **clang-format** support is enabled, CLion detects config files when opening a project and suggests overriding the current IDE settings. Code style rules from the `.clang-format` files are then applied automatically to all editor actions, including auto-completion, code generation, and refactorings.

Visual Studio Integration

Download the latest Visual Studio extension from the **alpha build site**. The default key-binding is Ctrl-R,Ctrl-F.

Script for patch reformatting

The python script *clang/tools/clang-format/clang-format-diff.py* parses the output of a unified diff and reformats all contained lines with **clang-format**.

```
usage: clang-format-diff.py [-h] [-i] [-p NUM] [-regex PATTERN] [-style STYLE]

Reformat changed lines in diff. Without -i option just output the diff that
would be introduced.

optional arguments:
  -h, --help      show this help message and exit
  -i              apply edits to files instead of displaying a diff
  -p NUM          strip the smallest prefix containing P slashes
  -regex PATTERN  custom pattern selecting file paths to reformat
  -style STYLE    formatting style to apply (LLVM, Google, Chromium, Mozilla,
                  WebKit)
```

So to reformat all the lines in the latest **git** commit, just do:

```
git diff -U0 --no-color HEAD^ | clang-format-diff.py -i -p1
```

With Mercurial/**hg**:

```
hg diff -U0 --color=never | clang-format-diff.py -i -p1
```

In an SVN client, you can do:

```
svn diff --diff-cmd=diff -x -U0 | clang-format-diff.py -i
```

The option *-U0* will create a diff without context lines (the script would format those as well).