CS 3460

Introduction to ClangFormat

What is ClangFormat

- Utility that formats source code based upon customizable settings.
  - C++, C#, Java, JavaScript, and a few others
- Has several standard style options
  - llvm, Google, GNU, Mozilla, Microsoft, others
- Large number of style formatting configuration options
- Used as…
  - command line tool
  - editor integration

Style Customization

- Config file; use specific name
  - `.clang-format`
  - `_clang-format` (I prefer because it doesn’t get hidden)
- From command line, indicate config file use with

  ```
  clang-format -i -style=file main.cpp
  ```

  - `-i` : in-place modification of the file. Otherwise formatted code is sent to standard output (the console)
  - `-style=file` : This is the exact option, `file` is not a placeholder. Tells clang-format to look for the config file. Starts in current directory and keeps searching up parent directories until it finds one (or not)
  - `main.cpp` : The name of the source file to format. Can specifiy more than one file

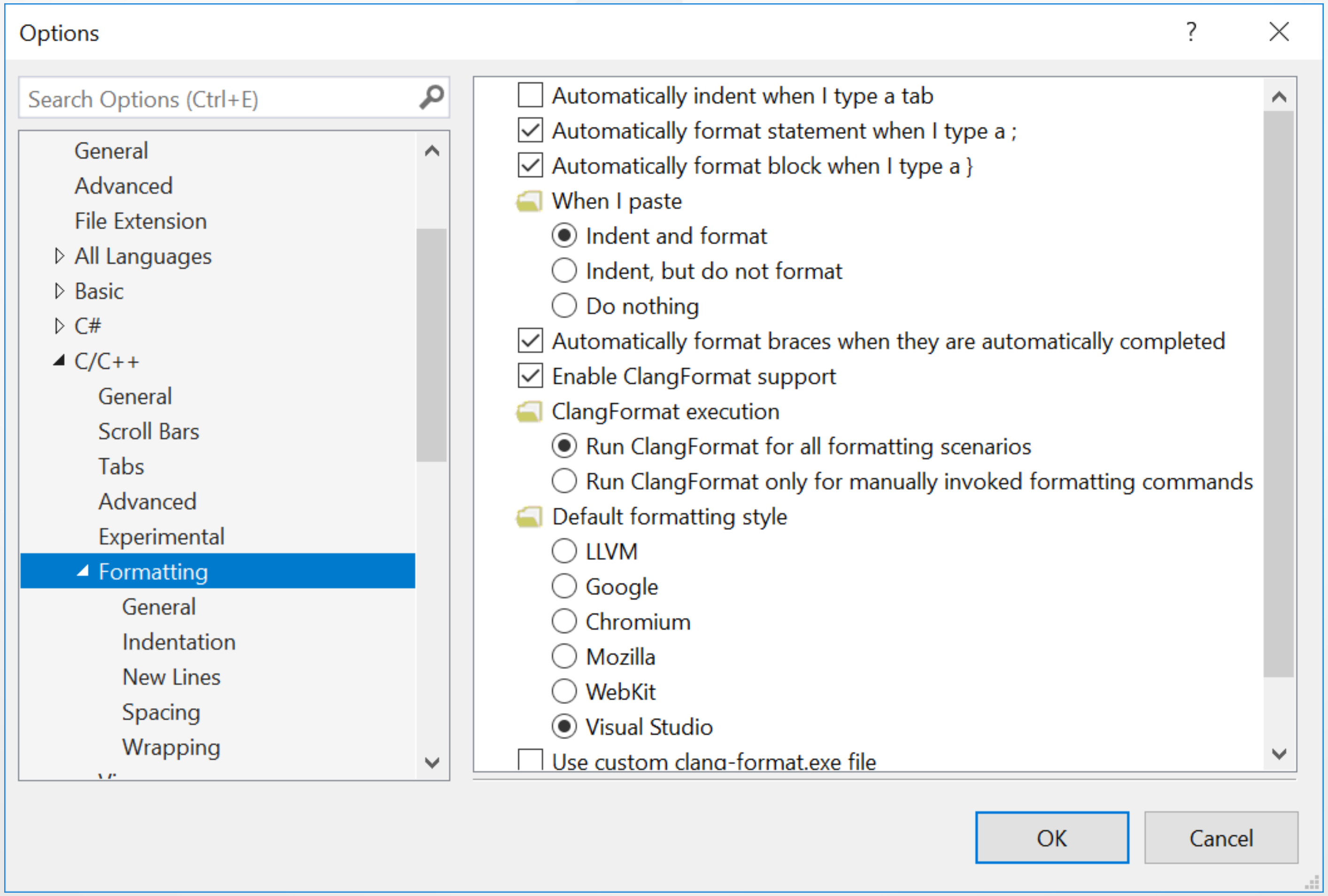
Style Customization

- Reference: https://clang.llvm.org/docs/ClangFormatStyleOptions.html
- Can find out the default settings with
  - `clang-format -dump-config > config.txt`

```
Language: Cpp
BreakBeforeBraces : Allman
IndentCaseLabels: true
NamespaceIndentation: Inner
IndentWidth: 4
TabWidth: 4
UseTab: Never
DerivePointerAlignment: false
PointerAlignment: Left
```

Visual Studio Integration

- Visual Studio 2017 and later have automatic detection and use of a ClangFormat config file.



Options

Search Options (Ctrl+E)

- General
- Advanced
- File Extension
- All Languages
- Basic
- C#
- C/C++
  - General
  - Scroll Bars
  - Tabs
  - Advanced
  - Experimental
  - Formatting
    - General
    - Indentation
    - New Lines
    - Spacing
    - Wrapping

☐ Automatically indent when I type a tab
☑ Automatically format statement when I type a ;
☑ Automatically format block when I type a }

When I paste
- [✓ Indent and format]
- Indent, but do not format
- Do nothing

☑ Automatically format braces when they are automatically completed
☑ Enable ClangFormat support

ClangFormat execution
- [✓ Run ClangFormat for all formatting scenarios]
- Run ClangFormat only for manually invoked formatting commands

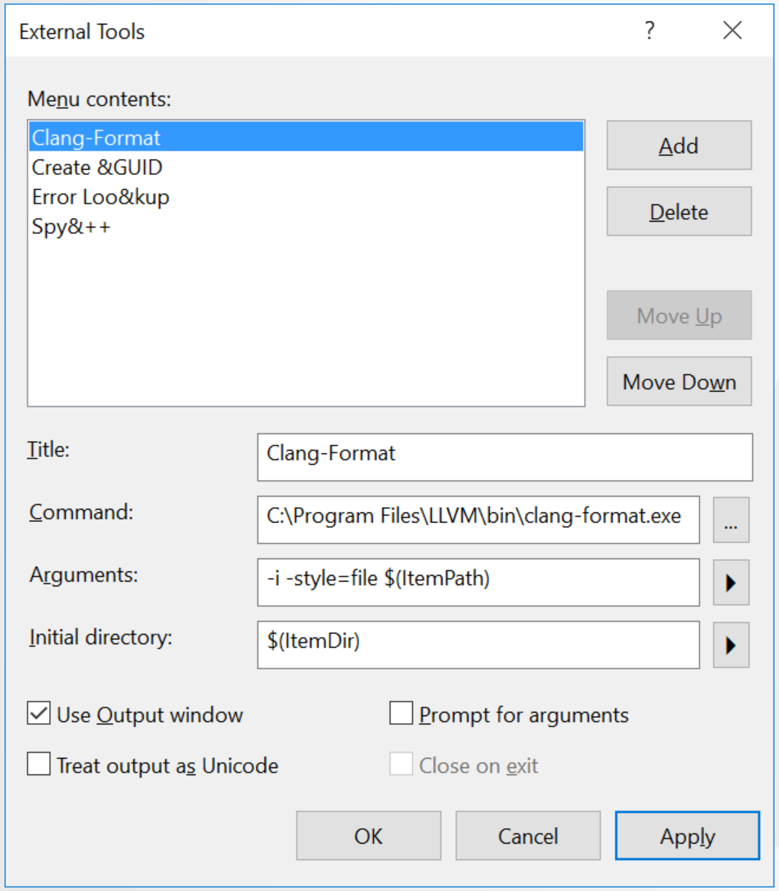
Default formatting style
- LLVM
- Google
- Chromium
- Mozilla
- WebKit
- [✓ Visual Studio]

☐ Use custom clang-format.exe file

OK Cancel

Visual Studio Integration

- Can also set up as an External Tool...



External Tools

Menu contents:

Clang-Format
Create &GUID
Error Loo&kup
Spy&++

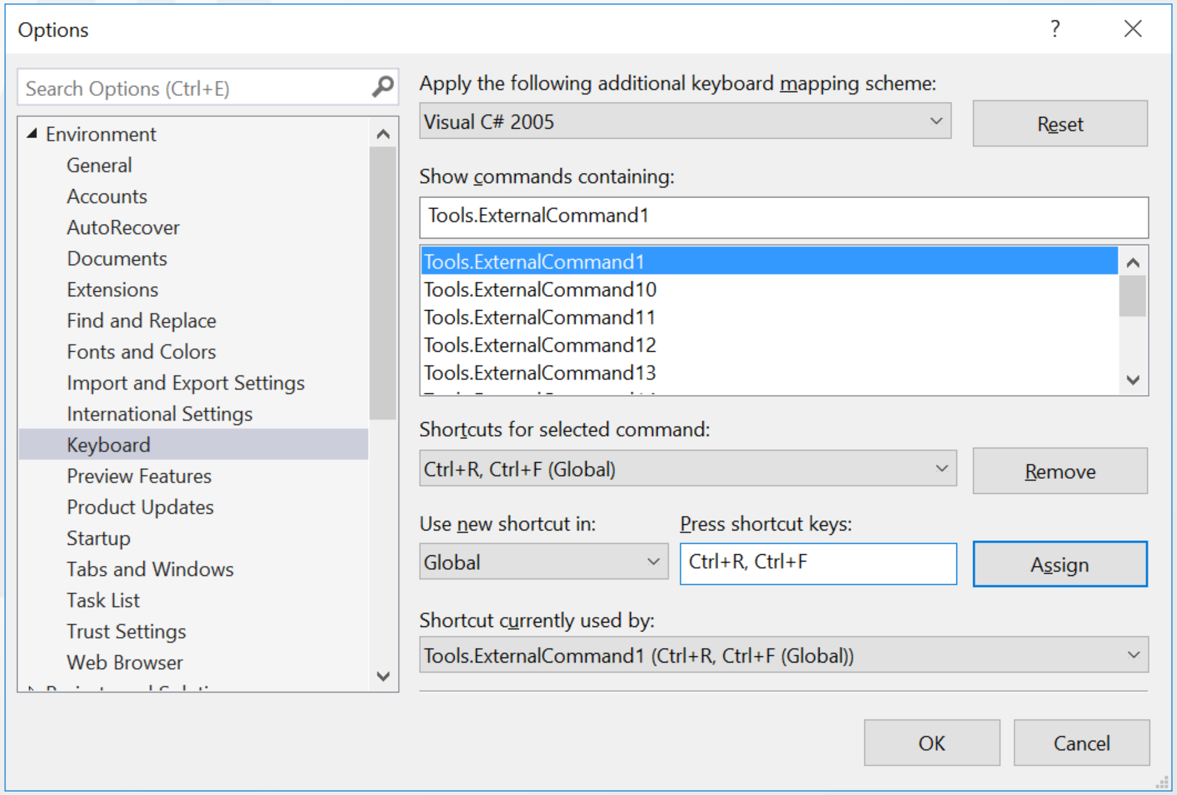
Title: Clang-Format

Command: C:\Program Files\LLVM\bin\clang-format.exe

Arguments: -i -style=file $(ItemPath)

Initial directory: $(ItemDir)

☑ Use Output window ☐ Prompt for arguments
☐ Treat output as Unicode ☐ Close on exit



Options

Apply the following additional keyboard mapping scheme: Visual C# 2005

Show commands containing: Tools.ExternalCommand1

Tools.ExternalCommand1
Tools.ExternalCommand10
Tools.ExternalCommand11
Tools.ExternalCommand12
Tools.ExternalCommand13

Shortcuts for selected command: Ctrl+R, Ctrl+F (Global)

Use new shortcut in: Global

Press shortcut keys: Ctrl+R, Ctrl+F

Shortcut currently used by: Tools.ExternalCommand1 (Ctrl+R, Ctrl+F (Global))

CMake Pre-Build Step Integration

- Best practice is to ensure code is formatted before adding it to a project repo.
  - We can accomplish that by integrating into the build process.
- Four steps to integrate with CMake
  1. Define which files to format; put names in a variable
  2. Find the location of the `clang-format` utility
  3. Add a custom build target that runs `clang-format`
  4. Set the build target as a dependency of the primary project target

CMake Integration – Step 1

- Define a variable with the files to format
  - Only a single file here, but more can be specified

```
set(SOURCE_FILES main.cpp)
```

CMake Integration – Step 2

- Find the `clang-format` utility
  - Use the `find_program` function in CMake
  - Will search the `PATH` environment variable, and others folders if specified.

```
find_program(CLANG_FORMAT "clang-format")
```

CMake Integration – Step 3

- Create custom target; two parts
  1. Build the full pathnames to all the source files
  2. Define the custom target

```
unset(SOURCE_FILES_PATHS)
foreach(SOURCE_FILE ${SOURCE_FILES})
    get_source_file_property(WHERE ${SOURCE_FILE} LOCATION)
    set(SOURCE_FILES_PATHS ${SOURCE_FILES_PATHS} ${WHERE})
endforeach()
```

```
add_custom_target(
    ClangFormat
    COMMAND ${CLANG_FORMAT}
    -i
    -style=file
    ${SOURCE_FILES_PATHS})
```

CMake Integration – Step 4

- Set the custom target project dependency
  - Set it to run anytime the project is built

```
add_dependencies(HelloWorld ClangFormat)
```

CMake Integration – Combined

```
cmake_minimum_required(VERSION 3.12)
project(HelloWorld)

set(SOURCE_FILES main.cpp)

add_executable(HelloWorld ${SOURCE_FILES})
set_property(TARGET HelloWorld PROPERTY CXX_STANDARD 20)

find_program(CLANG_FORMAT "clang-format")
if (CLANG_FORMAT)
    unset(SOURCE_FILES_PATHS)
    foreach(SOURCE_FILE ${SOURCE_FILES})
        get_source_file_property(WHERE ${SOURCE_FILE} LOCATION)
        set(SOURCE_FILES_PATHS ${SOURCE_FILES_PATHS} ${WHERE})
    endforeach()

    add_custom_target(
        ClangFormat
        COMMAND ${CLANG_FORMAT}
        -i
        -style=file
        ${SOURCE_FILES_PATHS})

    add_dependencies(HelloWorld ClangFormat)
endif()
```