C++ Code Formatting with `clang-format`

Nick Sarten

November 8, 2016

Code formatting

Enforce coding standards. Automatically.

Compared to other enforcement options:

- Impartial and objective
- Uses a full C++ parsing frontend (`clang`)
- No effect on code semantics

clang-format

```
OVERVIEW: A tool to format C/C++/Java/JavaScript/Objective-C/Protobuf 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> ...]
```

clang-format style files

The command-line option you will almost always specify is `-style=file`.

Style files

- YAML format.
- Filename: `.clang-format` or `_clang-format`.
- `clang-format` will automatically search for this file when `-style=file` specified.

clang-format style files

```
# General settings
BasedOnStyle: Mozilla
IndentWidth: 4
TabWidth: 4
UseTab: Never
---
Language: Cpp
# C++-specific settings
ColumnLimit: 120
KeepEmptyLinesAtTheStartOfBlocks: false
AlwaysBreakBeforeMultilineStrings: true
AlwaysBreakTemplateDeclarations: true
```

Formatting code locally

```
clang-format -style=file -i [<file> ...]
```

Disable formatting

```
// clang-format off
// clang-format on

/* clang-format off */
/* clang-format on */
```

Visual Studio plugin

Visual Studio Integration

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

clang-format plugin for Visual Studio

We also provide a standalone Visual Studio plugin for clang-format. It requires Visual Studio 2010 Professional or later. Notably, the Express editions do not support plugins.

Visual Studio plugin installer, based on SVN r284979.

http://llvm.org/builds/

Whole-file formatting

vs.

Diff-based formatting

Formatting diffs

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

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

```
usage: git clang-format [OPTIONS] [<commit>] [--] [<file>...]

Run clang-format on all lines that differ between the working directory
and <commit>, which defaults to HEAD.  Changes are only applied to the working
directory.
```

Integrating formatting with source control

CI Builder

Fail against pull requests or branches on improperly formatted code.

git pre-push/pre-receive hook

Reject pushes with improperly formatted code changes.

git pre-commit

Format code automatically on commit or reject improperly formatted code changes.

git pre-commit hook

Example pre-commit hook script:

`github.com/genbattle/grand-central-format/blob/master/git-hooks/pre-commit-local`

Config

`hooks.formatoncommit` Automatically format code on commit.

`hooks.rejectunformattedcommit` Reject unformatted commit attempts.

`hooks.formatwholefiles` Format the entirety of modified files.

`clang.clangformatdiff` Path to clang-format-diff.py.

Setting configuration options

```
git config --local --add hooks.formatoncommit true
```

Avoid invoking pre-commit hooks

```
git commit --no-verify
```

git pre-push/pre-receive hook

pre-push

- pre-push added in git 1.8.2.
- Run on the client, so shares some of pre-commit's issues.
- Some boilerplate required for diff-based formatting.

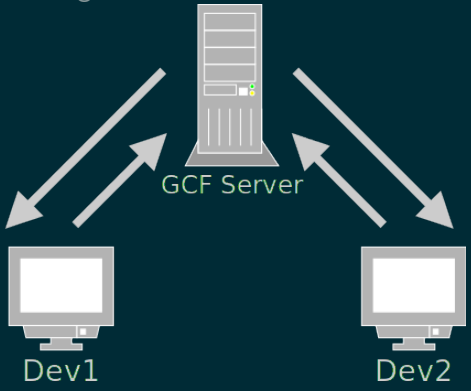
pre-receive

- Different implementation on every major self-hosted SCM.
- Not implemented for most cloud SCM services.

The problem with push hooks and CI builders for formatting is that improperly formatted code has already polluted your source history.

Consistency across heterogeneous dev. environments

Solution: centralized formatting



https://github.com/genbattle/grand-central-format

Thanks for listening!

Granularity

- Diff-only formatting
- Whole file formatting
- Whole codebase formatting

Enforcement

- CI builder
- pre-push/pre-receive
- pre-commit

`https://github.com/genbattle/grand-central-format`
`http://clang.llvm.org/docs/ClangFormat.html`

Twitter: @NickSarten
email: gen.battle@gmail.com

Thanks to WhereScape for hosting this meetup!