Contents

This directory contains tools for developers working on this repository.

clang-format-diff.py

A script to format unified git diffs according to .clang-format.

Requires `clang-format`, installed e.g. via `brew install clang-format` on macOS, or `sudo apt install clang-format` on Debian/Ubuntu.

For instance, to format the last commit with 0 lines of context, the script should be called from the git root folder as follows.

```
git diff -U0 HEAD~1.. | ./contrib/devtools/clang-format-diff.py -p1 -i -v
```

copyright_header.py

Provides utilities for managing copyright headers of `The Bitcoin Core developers` in repository source files. It has three subcommands:

```
$ ./copyright_header.py report <base_directory> [verbose]
$ ./copyright_header.py update <base_directory>
$ ./copyright_header.py insert <file>
```

Running these subcommands without arguments displays a usage string.

copyright_header.py report <base_directory> [verbose]

Produces a report of all copyright header notices found inside the source files of a repository. Useful to quickly visualize the state of the headers. Specifying `verbose` will list the full filenames of files of each category.

copyright_header.py update <base_directory> [verbose]

Updates all the copyright headers of `The Bitcoin Core developers` which were changed in a year more recent than is listed. For example:

```
// Copyright (c) <firstYear>-<lastYear> The Bitcoin Core developers
```

will be updated to:

```
// Copyright (c) <firstYear>-<lastModifiedYear> The Bitcoin Core developers
```

where `<lastModifiedYear>` is obtained from the `git log` history.

This subcommand also handles copyright headers that have only a single year. In those cases:

```
// Copyright (c) <year> The Bitcoin Core developers
```

will be updated to:

`// Copyright (c) <year>-<lastModifiedYear> The Bitcoin Core developers`

where the update is appropriate.

copyright_header.py insert <file>

Inserts a copyright header for `The Bitcoin Core developers` at the top of the file in either Python or C++ style as determined by the file extension. If the file is a Python file and it has `#!` starting the first line, the header is inserted in the line below it.

The copyright dates will be set to be `<year_introduced>-<current_year>` where `<year_introduced>` is according to the `git log` history. If `<year_introduced>` is equal to `<current_year>`, it will be set as a single year rather than two hyphenated years.

If the file already has a copyright for `The Bitcoin Core developers`, the script will exit.

gen-manpages.py

A small script to automatically create manpages in ../../doc/man by running the release binaries with the -help option. This requires help2man which can be found at: https://www.gnu.org/software/help2man/

With in-tree builds this tool can be run from any directory within the repostitory. To use this tool with out-of-tree builds set `BUILDDIR`. For example:

```
BUILDDIR=$PWD/build contrib/devtools/gen-manpages.py
```

security-check.py and test-security-check.py

Perform basic security checks on a series of executables.

symbol-check.py

A script to check that release executables only contain certain symbols and are only linked against allowed libraries.

For Linux this means checking for allowed gcc, glibc and libstdc++ version symbols. This makes sure they are still compatible with the minimum supported distribution versions.

For macOS and Windows we check that the executables are only linked against libraries we allow.

Example usage:

```
find ../path/to/executables -type f -executable | xargs python3 contrib/devtools/symbol-chec
```

If no errors occur the return value will be 0 and the output will be empty.

If there are any errors the return value will be 1 and output like this will be printed:

```
.../64/test_bitcoin: symbol memcpy from unsupported version GLIBC_2.14
.../64/test_bitcoin: symbol __fdelt_chk from unsupported version GLIBC_2.15
.../64/test_bitcoin: symbol std::out_of_range::~out_of_range() from unsupported version GLIB
.../64/test_bitcoin: symbol _ZNSt8__detail15_List_nod from unsupported version GLIBCXX_3.4.1
```

circular-dependencies.py

Run this script from the root of the source tree (`src/`) to find circular dependencies in the source code. This looks only at which files include other files, treating the `.cpp` and `.h` file as one unit.

Example usage:

```
cd .../src
../contrib/devtools/circular-dependencies.py {*,*/*,*/*/*}.{h,cpp}
```