Clang-Tidy

Fast linting and static analysis for your C++ project

Adam Graliński

C++ Friends, December 2022

What is clang-tidy?

It is a **linter** tool for C++[1], based on Clang toolset.

- Its purpose is to provide an extensible framework for diagnosing and fixing programming errors, like style violations, interface misuse, or bugs that can be deduced via static analysis.
- greatly customizable
- setup once (per project), write more correct code forever.

https://releases.llvm.org/6.0.0/tools/clang/tools/extra/docs/clang-tidy/index.html

[1] primarily. But C and some other languages are partially supported too

Installing Clang-Tidy on target system

Clang-Tidy is already packaged by most popular Linux distributions.

- Debian/Ubuntu

```
sudo apt-get install clang-tidy
```

Clang-Tidy is also part of Clang toolset

- Arch Linux

```
sudo pacman -S clang
```

https://archlinux.org/packages/extra/x86_64/clang/files/

Or provided by clang-tools-extra

- Fedora 29

```
sudo dnf install clang-tools-extra
```

Or one can always build it from source.

Getting a list of all available checkers

```
clang-tidy --list-checks -checks='*' | grep "modernize"
```

```
modernize-avoid-bind                                      modernize-unary-static-assert
modernize-avoid-c-arrays                                  modernize-use-auto
modernize-concat-nested-namespaces                        modernize-use-bool-literals
modernize-deprecated-headers                              modernize-use-default-member-init
modernize-deprecated-ios-base-aliases                     modernize-use-emplace
modernize-loop-convert                                    modernize-use-equals-default
modernize-make-shared                                     modernize-use-equals-delete
modernize-make-unique                                     modernize-use-nodiscard
modernize-pass-by-value                                   modernize-use-noexcept
modernize-raw-string-literal                              modernize-use-nullptr
modernize-redundant-void-arg                              modernize-use-override
modernize-replace-auto-ptr                                modernize-use-trailing-return-type
modernize-replace-disallow-copy-and-assign-macro          modernize-use-transparent-functors
modernize-replace-random-shuffle                          modernize-use-uncaught-exceptions
modernize-return-braced-init-list                         modernize-use-using
modernize-shrink-to-fit
```

Structure of clang-tidy invocation

```
clang-tidy -checks='*' file.cpp -- -Isrc/include -DMY_DEFINES ...
```

Let's check modernize-use-override checker!

code/my_game.cpp

```
#include <iostream>
#include <memory>

class GameObject {
 public:
   virtual void draw() {
     // Reimplement this method in derived classes.
   }
};

class Starship: public GameObject {
 public:
  virtual void draw() {
    // Draw the engine(s) and engine exhaust
    // Draw the chassis
    // Draw the weapons
  }
};

int main() {
  Starship my_ship{};
  my_ship.draw();
}
```

Spot the problem?

Let's check modernize-use-override checker!

code/my_game.cpp

```
#include <iostream>
#include <memory>

class GameObject {
 public:
   virtual void draw() {
     // Reimplement this method in derived classes.
   }
};

class Starship: public GameObject {
 public:
  virtual void draw() {
    // Draw the engine(s) and engine exhaust
    // Draw the chassis
    // Draw the weapons
  }
};

int main() {
  Starship my_ship{};
  my_ship.draw();
}
```

Spot the problem?

Since C++11, one should use **`override`** keyword to mark functions in derived classes that override functions defined in the base class.

But let's have `clang-tidy` tell us that.

```
clang-tidy --checks='modernize-use-override'
         code/my_game.cpp --
```

Let's check modernize-use-override checker!

```
clang-tidy --checks='modernize-use-override' code/my_game.cpp --
```

```
Error while trying to load a compilation database:
Could not auto-detect compilation database for file "code/my_game.cpp"
No compilation database found in /mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code
    or any parent directory
json-compilation-database: Error while opening JSON database: No such file or directory
fixed-compilation-database: Error while opening fixed database: No such file or directory
Running without flags.
90 warnings generated.
/mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code/my_game.cpp:13:16: warning:
    prefer using 'override' or (rarely) 'final' instead of 'virtual' [modernize-use-override]

  virtual void draw() {
  ~~~~~~~~      ^
                     override
Suppressed 89 warnings (89 in non-user code).
Use -header-filter=.* to display errors from all non-system headers.
    Use -system-headers to display errors from system headers as well.
```

Let's check modernize-use-override checker!

```
clang-tidy --checks='modernize-use-override' code/my_game.cpp --
```

```
Error while trying to load a compilation database:
Could not auto-detect compilation database for file "code/my_game.cpp"
No compilation database found in /mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code
    or any parent directory
json-compilation-database: Error while opening JSON database: No such file or directory
fixed-compilation-database: Error while opening fixed database: No such file or directory
Running without flags.
90 warnings generated.
/mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code/my_game.cpp:13:16: warning:
    prefer using 'override' or (rarely) 'final' instead of 'virtual' [modernize-use-override]

  virtual void draw() {
  ~~~~~~~~     ^
                      override
Suppressed 89 warnings (89 in non-user code).
Use -header-filter=.* to display errors from all non-system headers.
    Use -system-headers to display errors from system headers as well.
```

OK. Do you know how to fix it?

- Y (*congratulations!*)
- N (*don't worry, it can be fixed automatically!*)

Fixing the indicated problem

```
clang-tidy --checks='modernize-use-override' code/my_game_fixed.cpp -fix --
```

```
Error while trying to load a compilation database:
Could not auto-detect compilation database for file "code/my_game_fixed.cpp"
No compilation database found in /mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code or any parent direct
json-compilation-database: Error while opening JSON database: No such file or directory
fixed-compilation-database: Error while opening fixed database: No such file or directory
Running without flags.
90 warnings generated.
/mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code/my_game_fixed.cpp:13:16: warning: prefer using 'overr
  virtual void draw() {
  ~~~~~~~~     ^
                      override
/mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code/my_game_fixed.cpp:13:3: note: FIX-IT applied suggeste
  virtual void draw() {
  ^
/mnt/vault/Repos/agral/Lectures/CPP_FFFE/26_ClangTidy/code/my_game_fixed.cpp:13:22: note: FIX-IT applied suggest
  virtual void draw() {
                     ^
clang-tidy applied 2 of 2 suggested fixes.
Suppressed 89 warnings (89 in non-user code).
Use -header-filter=.* to display errors from all non-system headers. Use -system-headers to display errors from
```

Fixing the indicated problem

before

```
#include <iostream>
#include <memory>

class GameObject {
 public:
   virtual void draw() {
     // Reimplement this method in derived classes.
   }
};

class Starship: public GameObject {
 public:
  virtual void draw() {
    // Draw the engine(s) and engine exhaust
    // Draw the chassis
    // Draw the weapons
  }
};

int main() {
  Starship my_ship{};
  my_ship.draw();
}
```

after

```
#include <iostream>
#include <memory>

class GameObject {
 public:
   virtual void draw() {
     // Reimplement this method in derived classe
   }
};

class Starship: public GameObject {
 public:
  void draw() override {
    // Draw the engine(s) and engine exhaust
    // Draw the chassis
    // Draw the weapons
  }
};

int main() {
  Starship my_ship{};
  my_ship.draw();
}
```

Key takeaways

- You are probably already using it.
- If you don't, *start using it*.
- **`modernize-*`** checks especially useful in legacy codebases.

Thank you!