An update on Clang-based C++ Tooling

Manuel Klimek
Daniel Jasper
Tomorrowland (from Euro LLVM DevMeeting 2012)

- **Tools**
  - clang-format
  - clang-lint
  - clang-rename

- **Libraries**
  - Tooling
  - Refactoring
  - ASTMatchers

- **Editor integration**
  - Emacs
  - Vim
  - Eclipse

- **IDE’ish Services**
  - ClangD
Today

• Tools
  o clang-format ✔
  o clang-lint-tidy ✔
  o clang-rename ✘

• Libraries
  o Tooling ✔
  o Refactoring ✔
  o ASTMatchers ✔

• Editor integration
  o Emacs ✔  YouCompleteMe
  o Vim ✔
  o Eclipse ✔  ycmd (libclang)

• IDE’ish Services
  o ClangD ✔
clang-format

- Automatic formatting for C++, ObjC, ...
- New features
  - More languages: JavaScript, Java, Protocol Buffers
  - Include sorting
- Widely used across the world
- Plugins for many editors and IDEs

http://clang.llvm.org/docs/ClangFormat.html
YouCompleteMe

- Code completion and more for vim, emacs, sublime text etc.
- Many languages (C++, Java, Python, Go)
- C++ support based on libclang
  - Code completion
  - Fast syntax checks
  - GoToDeclaration, GoToDefinition
  - Apply FixIt hints

https://github.com/Valloric/YouCompleteMe
clang-tidy

● clang-based C++ linter tool (and much more)
● >50 checks
  ○ Readability, efficiency, correctness, modernize, …
  ○ Highly configurable per (sub-)project
  ○ Can automatically fix the code in many cases
● Easy access to ASTMatchers and preprocessor hooks

http://clang.llvm.org/extra/clang-tidy/
AST matchers

● DSL to create predicates on Clang’s AST

● New features
  ○ More matchers (types, parents, ..)
  ○ Back-references (equalsNode("X") )
  ○ Starting nested matches within the callback

● clang-query
  ○ Quickly write and test AST matchers
  ○ Analyze translation units

http://clang.llvm.org/docs/LibASTMatchers.html
Demo

- From the LLVM Coding Standards: “Use Early Exits and continue to Simplify Code”

```c
if (!isa<TerminatorInst>(I) && I->hasOneUse() && doOtherThing(I)) {
    ... some long code ....
}
```

```c
if (isa<TerminatorInst>(I))
    return;
if (!I->hasOneUse())
    return;
if (!doOtherThing(I))
    return;
... some long code ....
```