clang-tidy

Adding a new check (live)

Jeremy Demeule, jeremy.demeule@gmail.com, @jeremydemeule

2019/06/15

Maintaining codeline

Keeping code up to date

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


static void print(std::auto_ptr<int> ptr) {
  std::cout << (ptr.get() ? *ptr : -1) << "\n";
}

int main(int argc, char** argv) {
  std::auto_ptr<int> ptr(new int(42));
  print(ptr);
  return 0;
}
```

Option 1

Search & Replace

Search & Replace

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



static void print(std::unique_ptr<int> ptr) {
  std::cout << (ptr.get() ? *ptr : -1) << "\n";
}

int main(int argc, char** argv) {
  std::unique_ptr<int> ptr(new int(42));
  print(p
  return
}
```

```
auto_ptr.cpp:11:9: error: call to implicitly-deleted copy
constructor of 'std::unique_ptr<int>'
  print(ptr);
        ^~~
```

Option 2

Dedicated Tools

clang-tidy

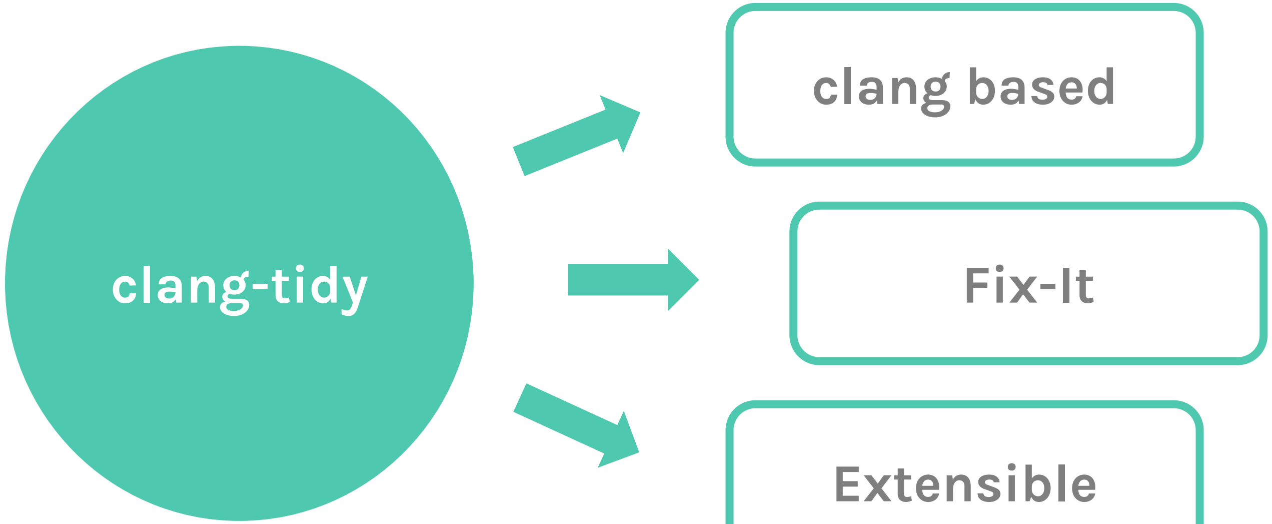
```cpp
#include <iostream>
#include <memory>
#include <utility>

static void print(std::unique_ptr<int> ptr) {
  std::cout << (ptr.get() ? *ptr : -1) << "\n";
}

int main(int argc, char** argv) {
  std::unique_ptr<int> ptr(new int(42));
  print(std::move(ptr));
  return 0;
}
```

Option 3

Do It Yourself

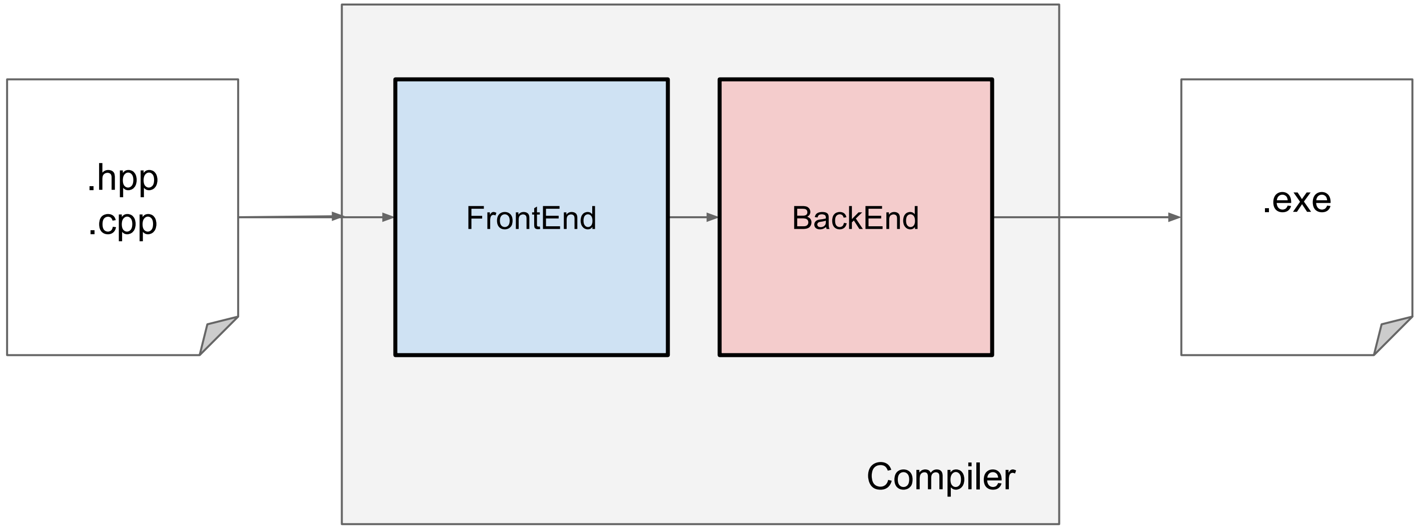
But not from scratch

llvm/clang framework

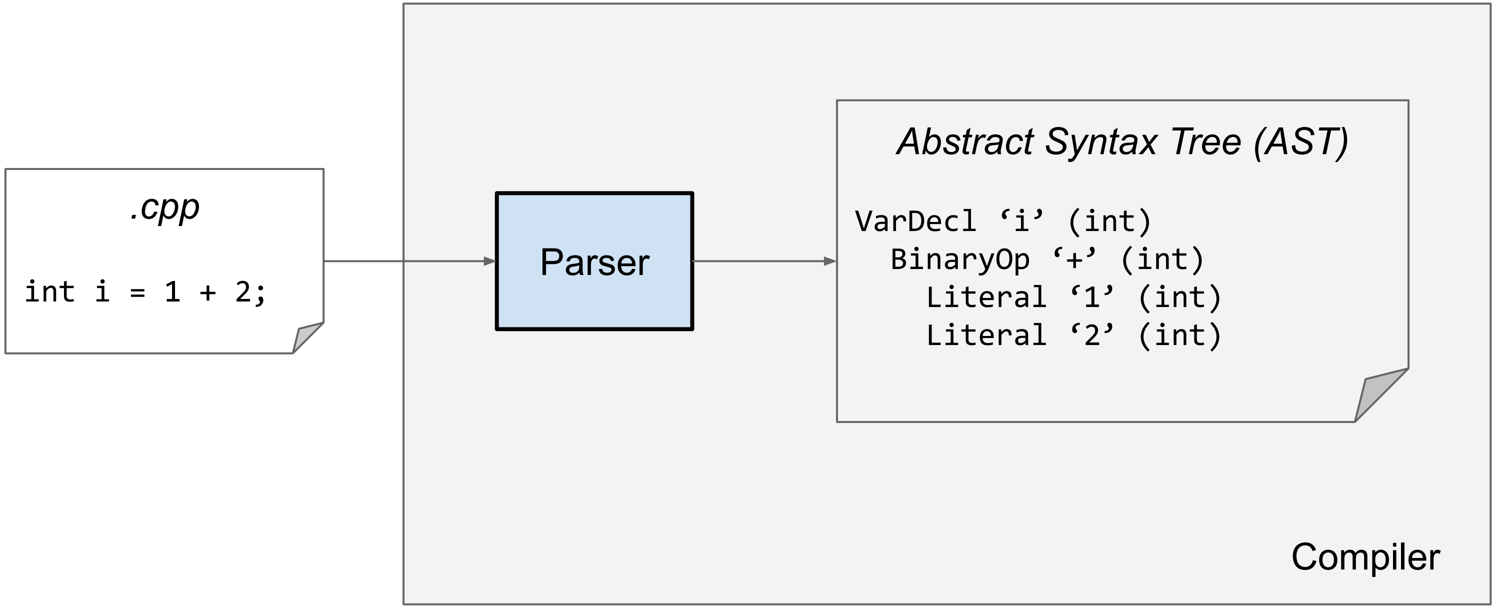


clang-tidy demo

How does a compiler works



Zoom on front end



```
int i = 1 + 2;
```

```
VarDecl ‘i’ (int)
  BinaryOp ‘+’ (int)
    Literal ‘1’ (int)
    Literal ‘2’ (int)
```

Focus on Abstract Syntax Tree (AST)

```
int i = 1 + 2;
```

```
VarDecl 0x7fd335854f18 </src/simple.cpp:1:1, col:13> col:5 i 'int' cinit
  `-BinaryOperator 0x7fd335855008 <col:9, col:13> 'int' '+'
    |-IntegerLiteral 0x7fd335854fc8 <col:9> 'int' 1
    `-IntegerLiteral 0x7fd335854fe8 <col:13> 'int' 2
```

clang-query demo

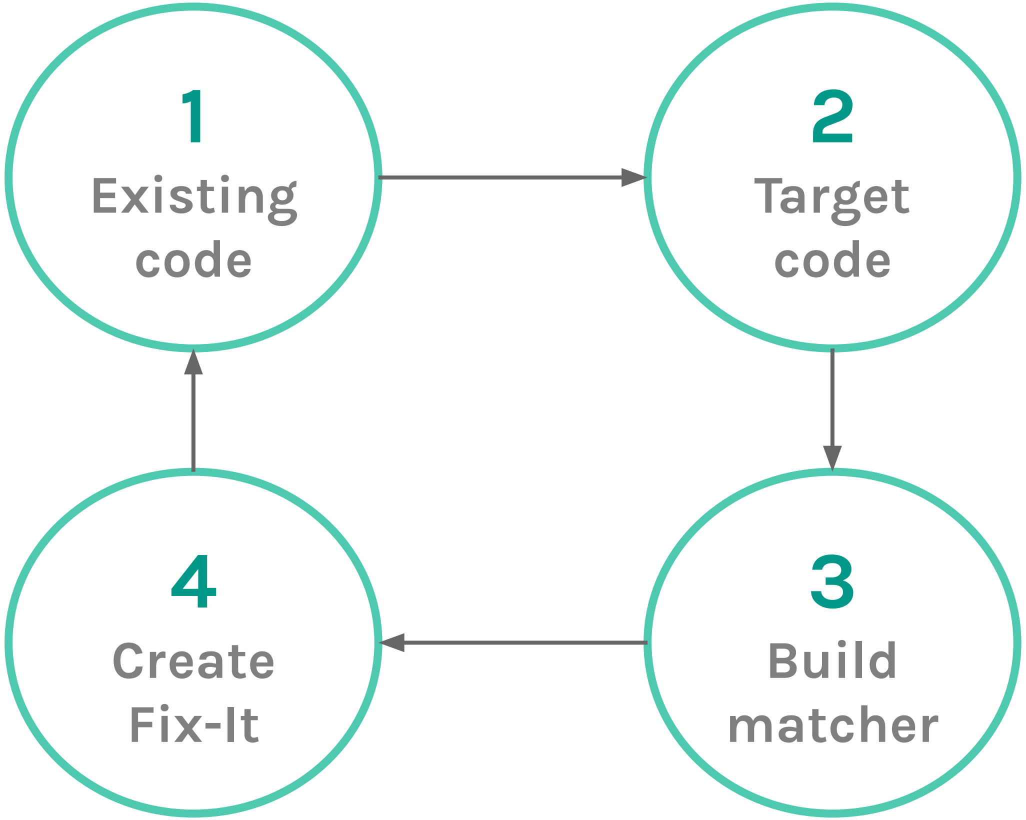
Our example

```
std::copy(v1.begin(), v1.end(), std::back_inserter(out));
```

$$\downarrow$$

```
std::ranges::copy(v1, std::back_inserter(out));
```

Building a clang-tidy check - Recipe



Our example - Step 1

~~std::copy~~(v1.begin(), v1.end(), std::back_inserter(out));

↓

```
std::ranges::copy(v1.begin(), v1.end(), std::back_inserter(out));
```

Our example - Step 2

```
std::ranges::copy(~~v1.begin(), v1.end()~~, std::back_inserter(out));
```

$\downarrow$

```
std::ranges::copy(v1, std::back_inserter(out));
```

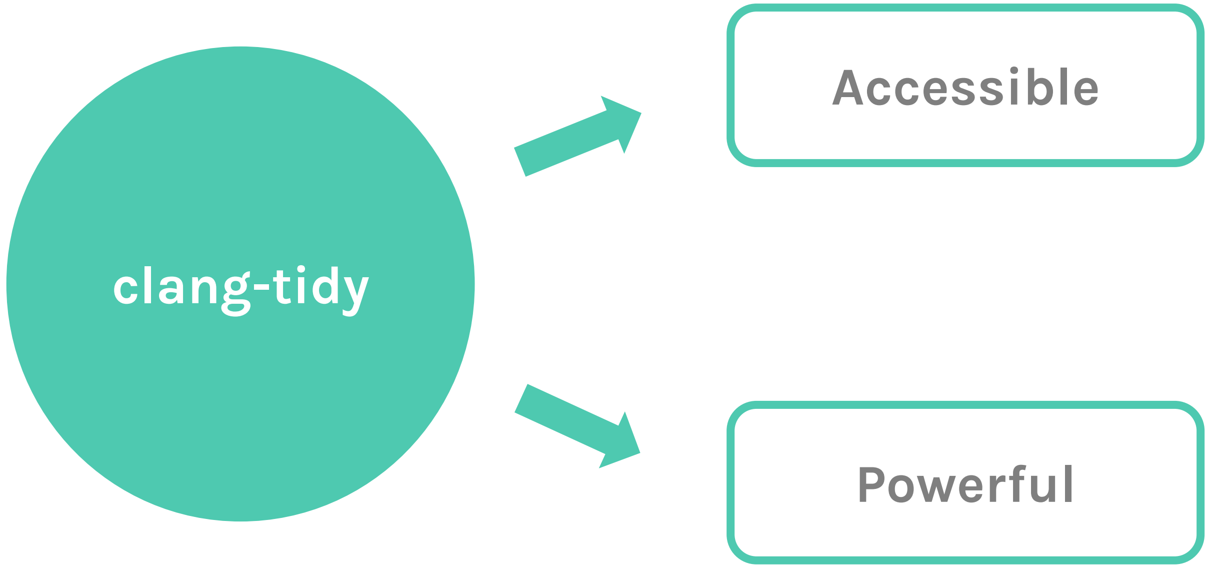
Demo

Conclusion

Maintaining codeline

Need tools

Maintaining codeline



THANKS!

Any questions?

Jeremy Demeule, jeremy.demeule@gmail.com, @jeremydemeule

References

llvm / clang documentation

Sources: https://github.com/llvm/llvm-project

Building steps: http://llvm.org/docs/CMake.html

clang-tidy documentation:
https://clang.llvm.org/extra/clang-tidy/index.html

Articles

Stephen Kelly blog: https://steveire.wordpress.com/

Stephen Kelly MSDN articles:

https://devblogs.microsoft.com/cppblog/exploring-clang-tooling-part-1-extending-clang-tidy/

https://devblogs.microsoft.com/cppblog/exploring-clang-tooling-part-2-examining-the-clang-ast-with-clang-query/

https://devblogs.microsoft.com/cppblog/exploring-clang-tooling-part-3-rewriting-code-with-clang-tidy/