./clang-talk

Peter Goldsborough

March 2, 2017

Table of Members

```
class Talk {
  public:




};
```

Table of Members

```
class Talk {
  public:

  llvm::LLVM CompilerToolchain;









};
```

Table of Members

```
class Talk {
  public:

  llvm::LLVM CompilerToolchain;


  clang::Clang Frontend;









};
```

Table of Members

```
class Talk {
  public:

  llvm::LLVM CompilerToolchain;


  clang::Clang Frontend;


  clang::Tooling Tooling;




};
```

Table of Members

```
class Talk {
  public:

  llvm::LLVM CompilerToolchain;


  clang::Clang Frontend;


  clang::Tooling Tooling;


  clang::Tooling YourTool;

};
```

Table of Members

```
class Talk {
  public:

  llvm::LLVM CompilerToolchain;


  clang::Clang Frontend;


  clang::Tooling Tooling;


  std::vector<clang::Tooling> YourTools;

};
```

LLVM

LLVM

Low

LLVM

Low Level

LLVM

Low Level Virtual

LLVM

Low Level Virtual Machine

LLVM

~~Low Level Virtual Machine~~

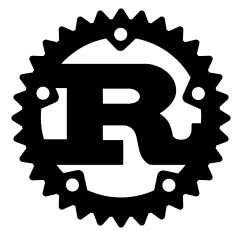
LLVM

~~Low Level Virtual Machine~~

lldb opt lld lljvm

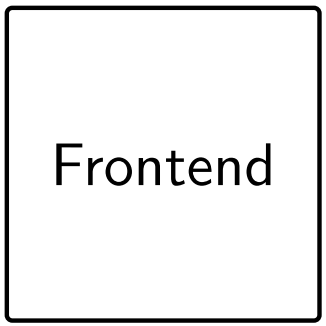
LLVM

~~Low Level Virtual Machine~~

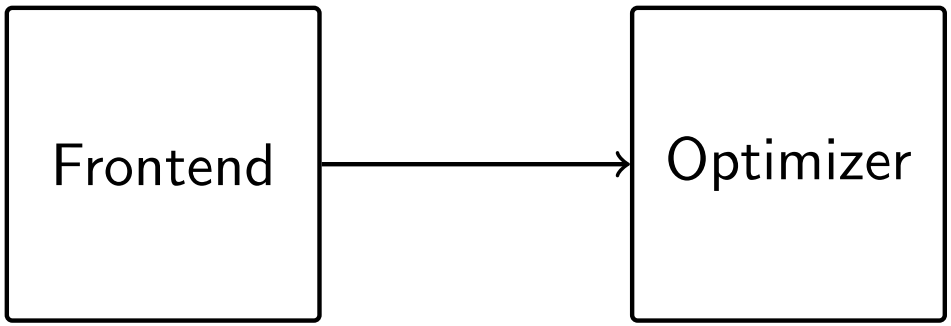


LLVM

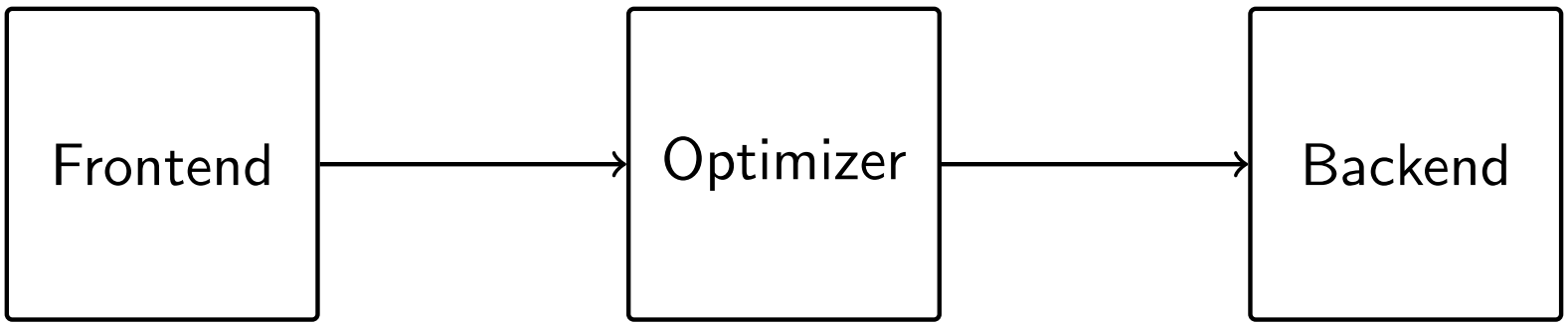
LLVM



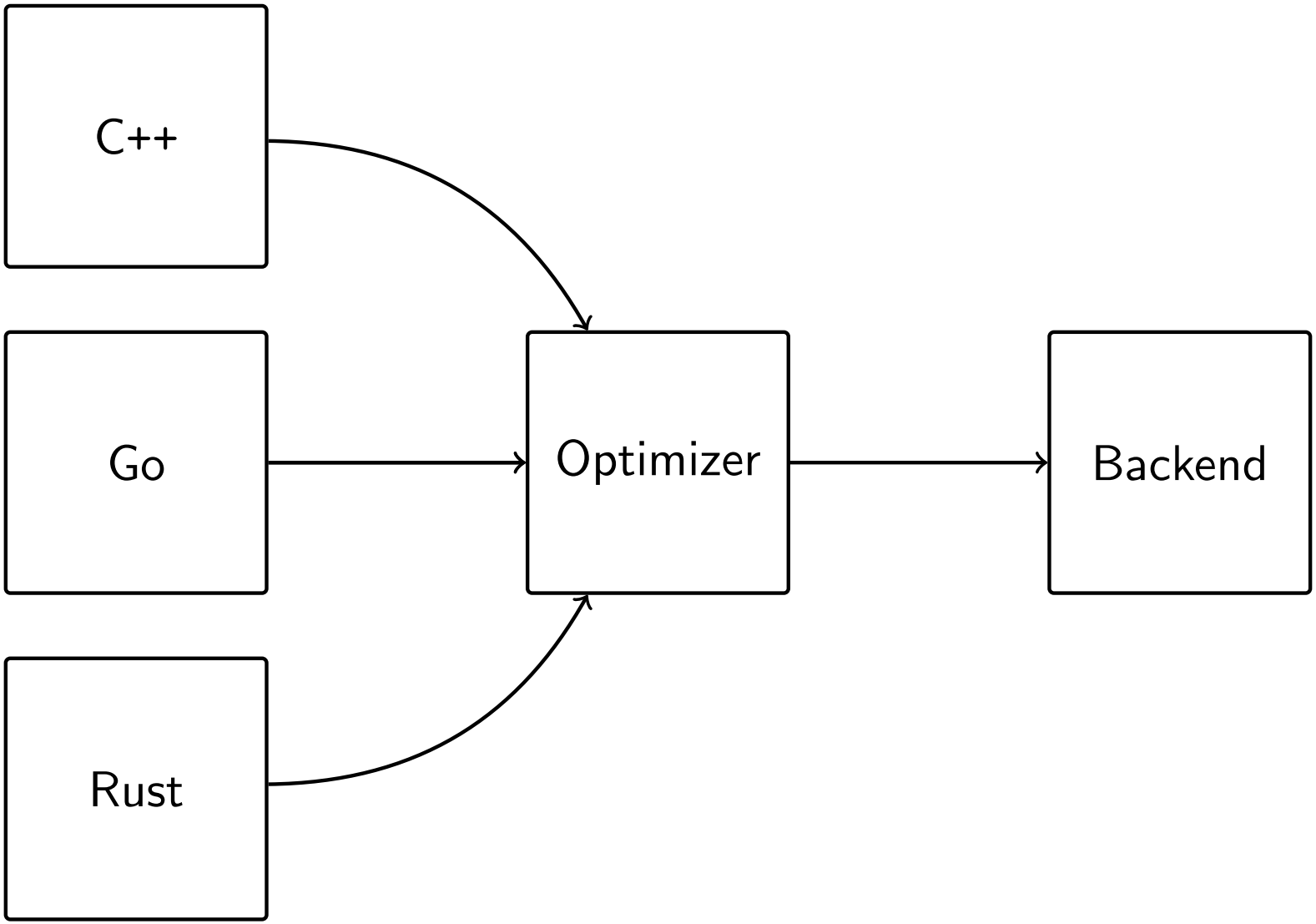
LLVM



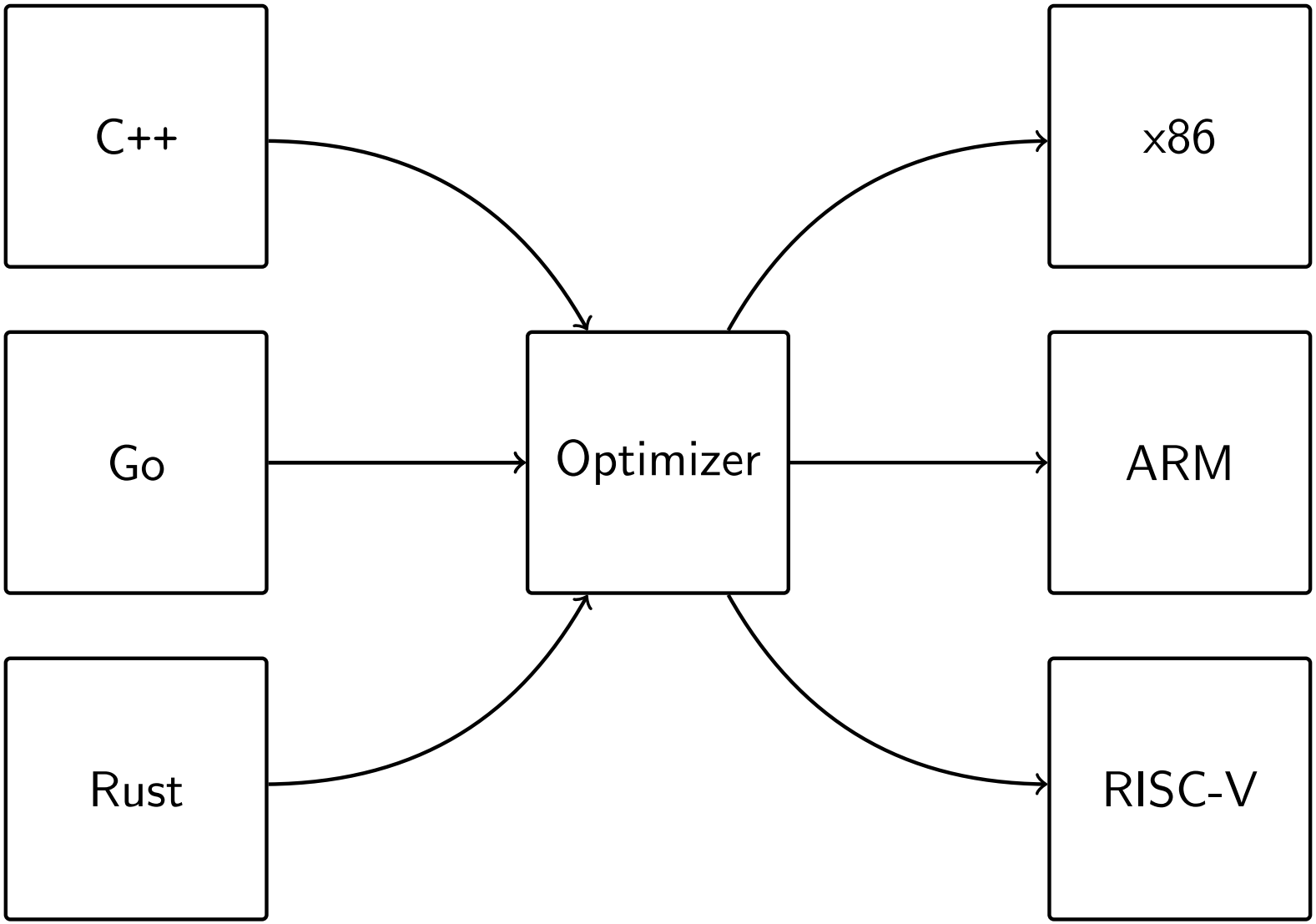
LLVM



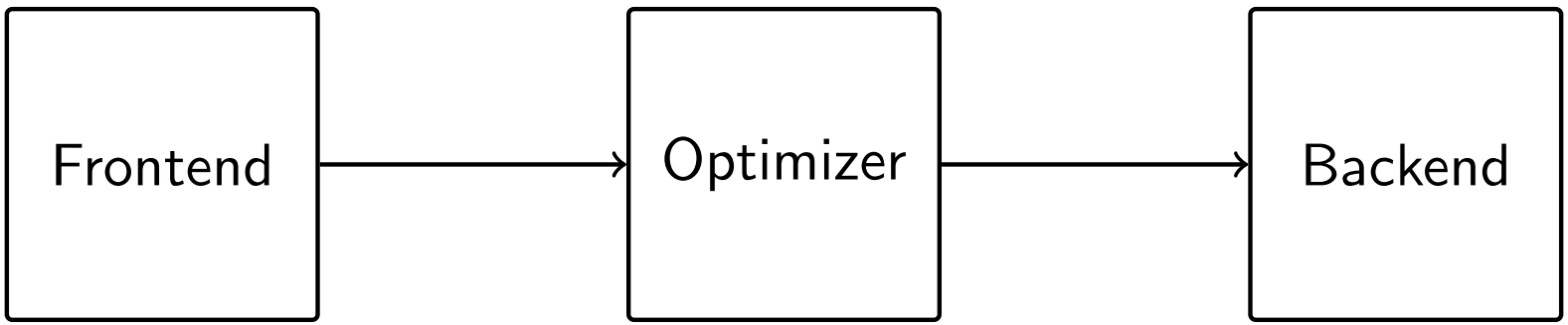
LLVM



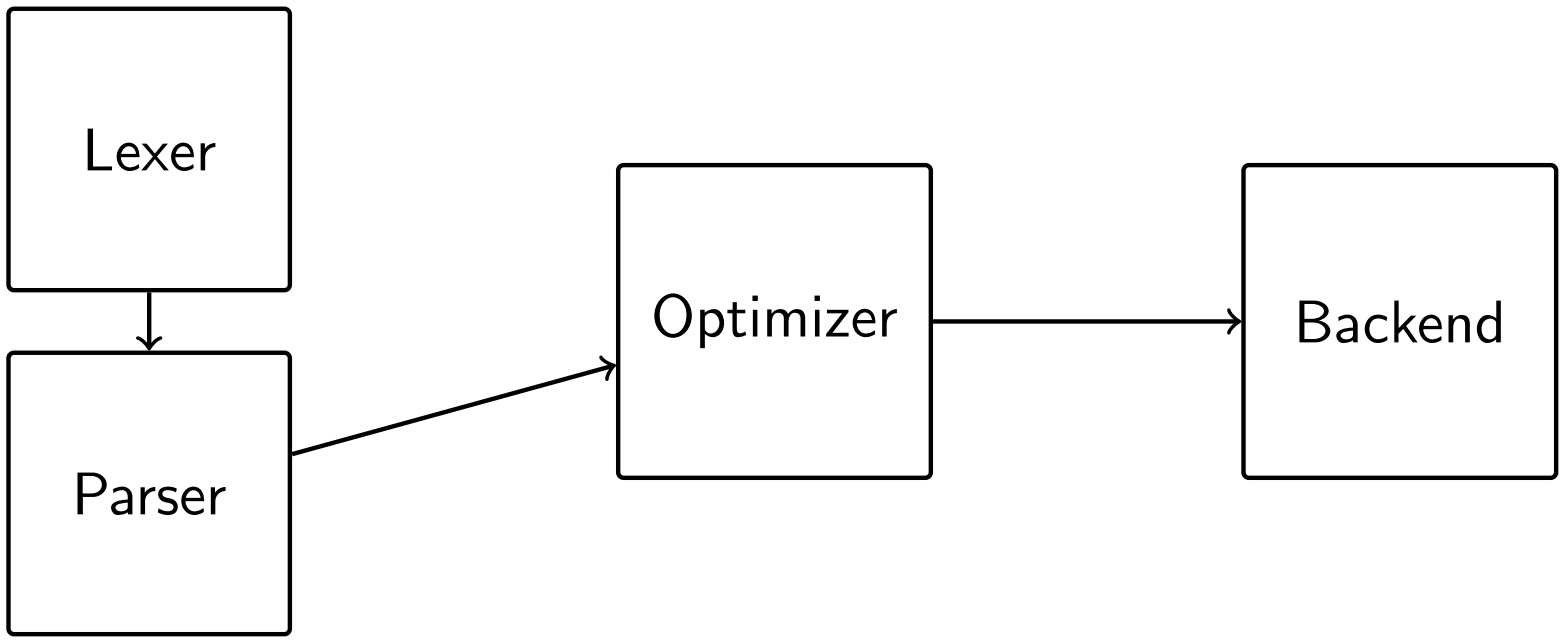
LLVM



LLVM

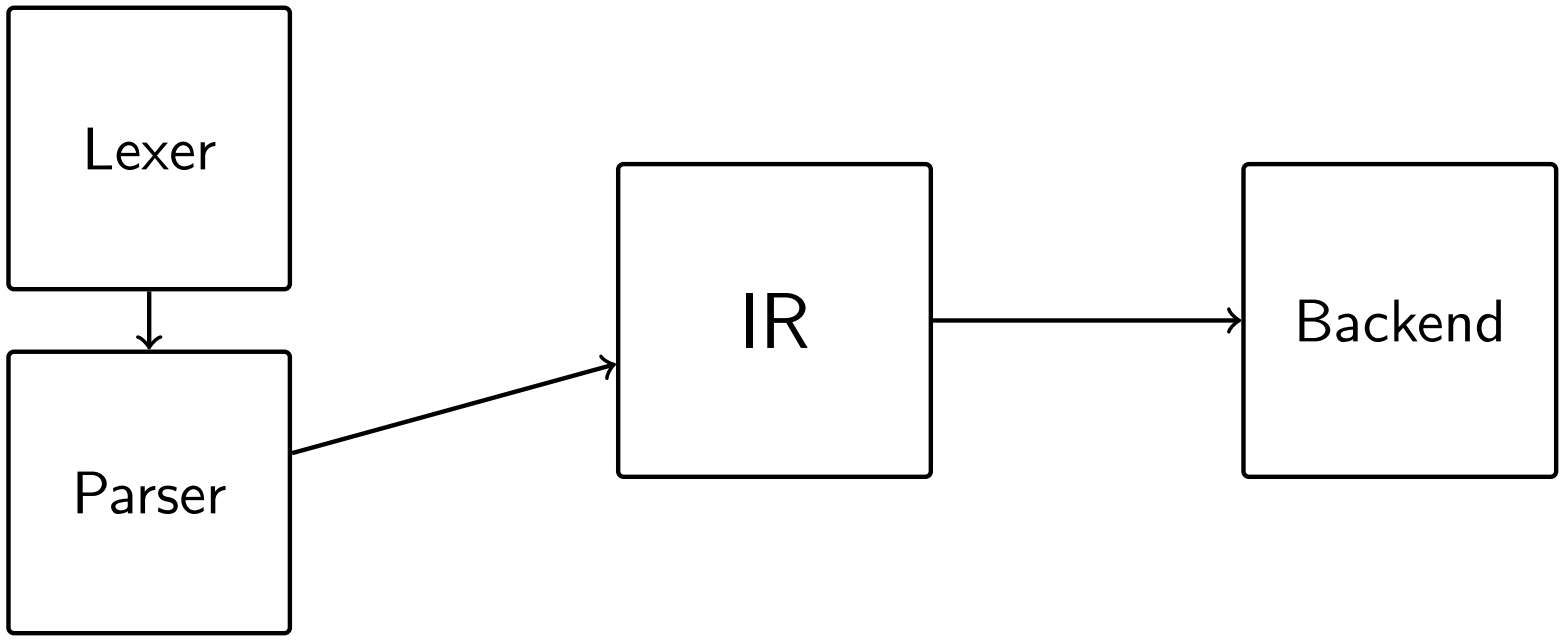


LLVM



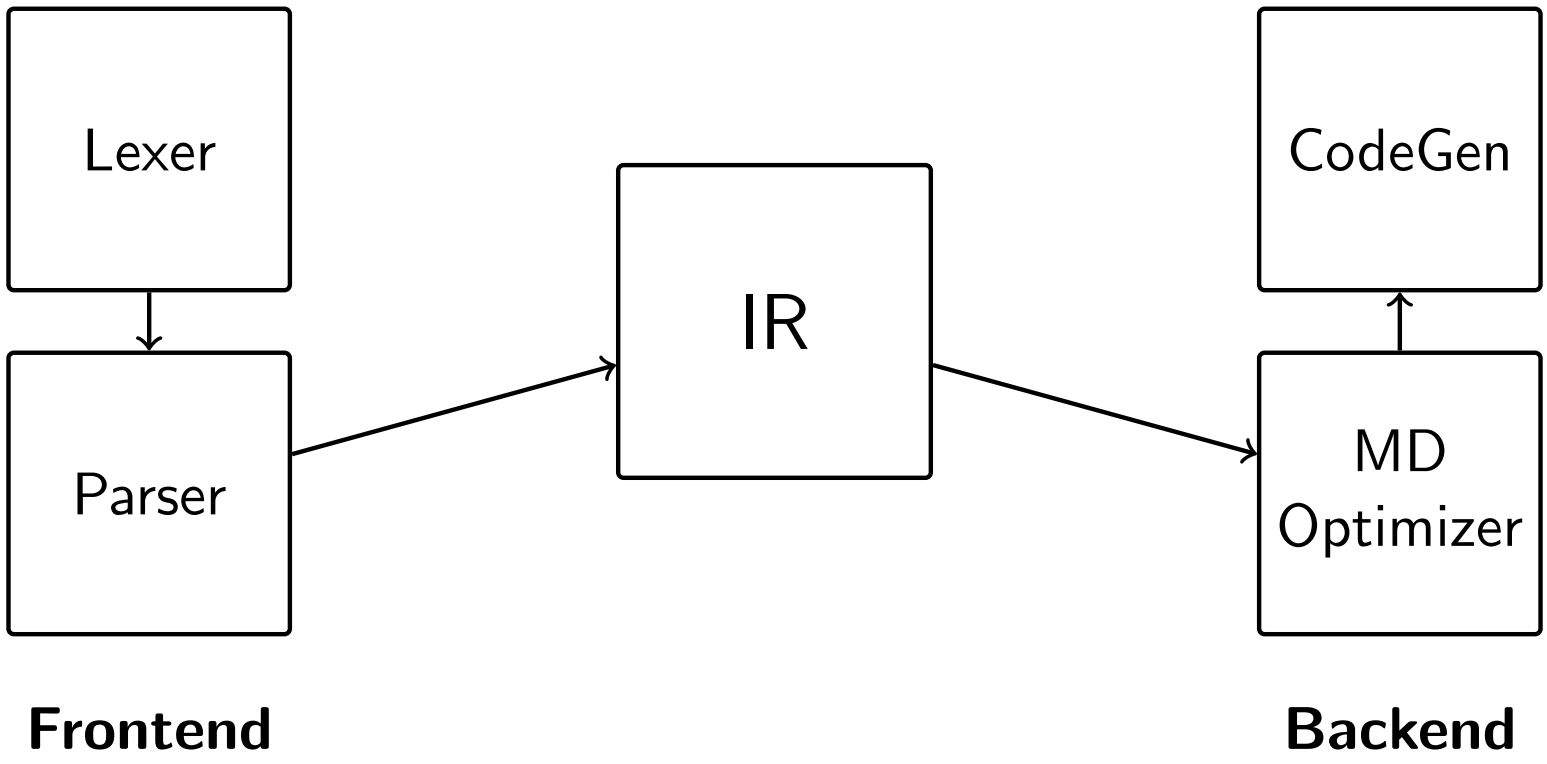
Frontend

LLVM

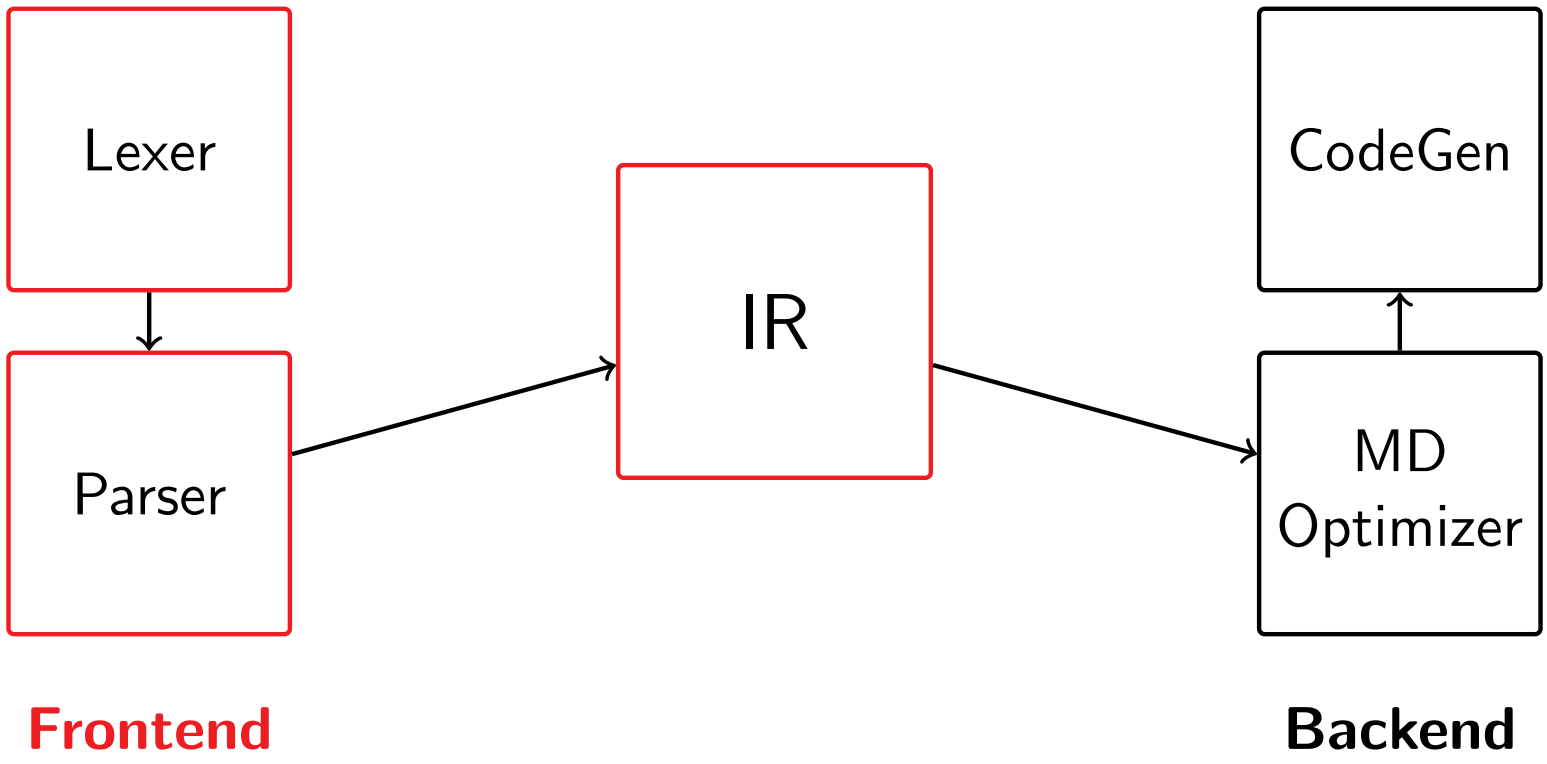


Frontend

LLVM



LLVM



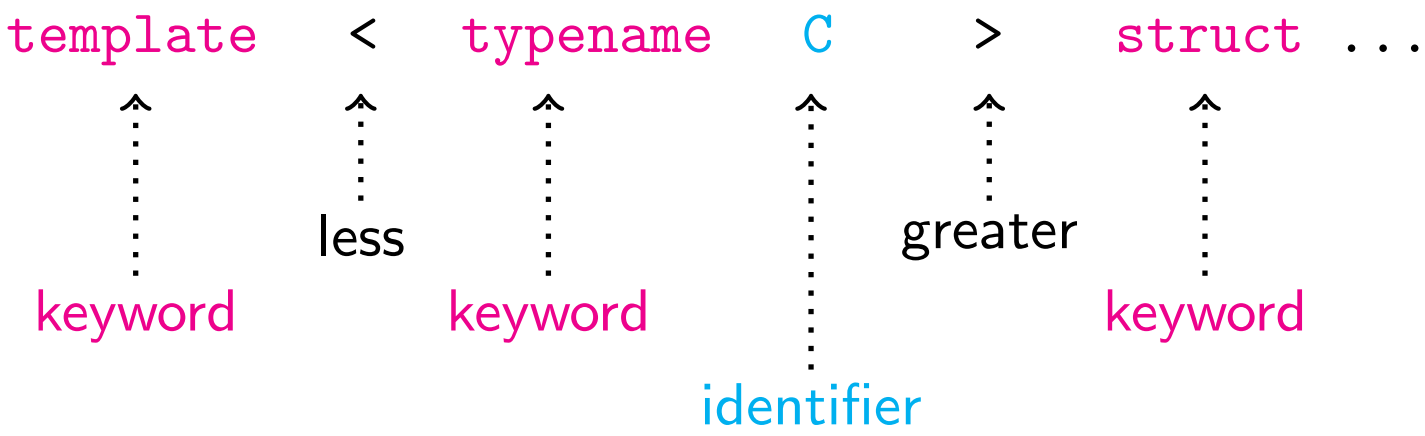
clang

```
template<typename C>
struct L {
  char a();
  void n(int) const;
  bool g;
};
```

Clang: Lexer

```
template   <   typename   C     >   struct ...
```

Clang: Lexer


template
<
typename
C
>
struct
...
keyword
less
keyword
identifier
greater
keyword

Clang: Parser

Clang: Parser

```
void n ( int arg = 42 ) const ;
```

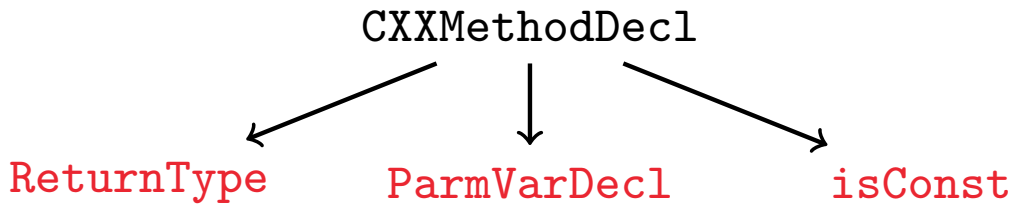
Clang: Parser

```
void n ( int arg = 42 ) const ;
```

CXXMethodDecl

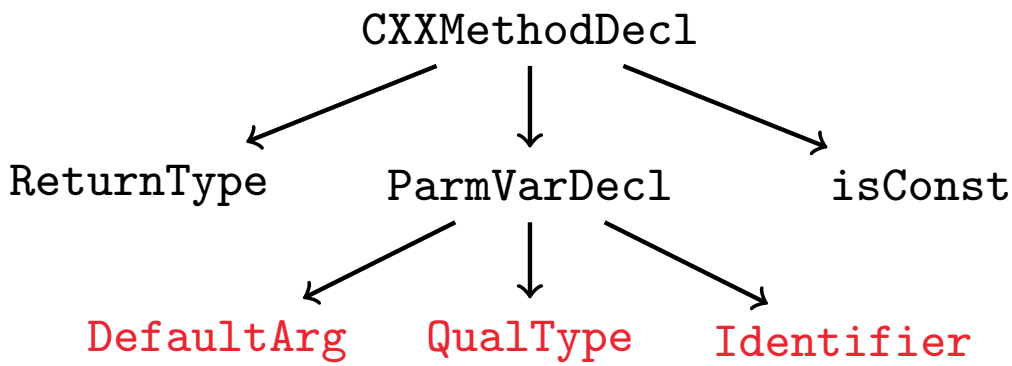
Clang: Parser

```
void n ( int arg = 42 ) const ;
```



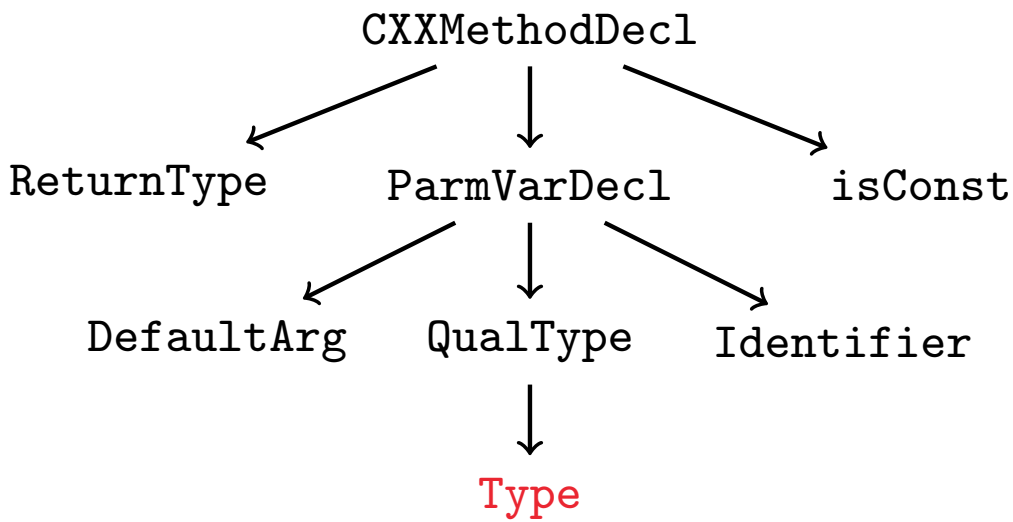
Clang: Parser

```
void n ( int arg = 42 ) const ;
```



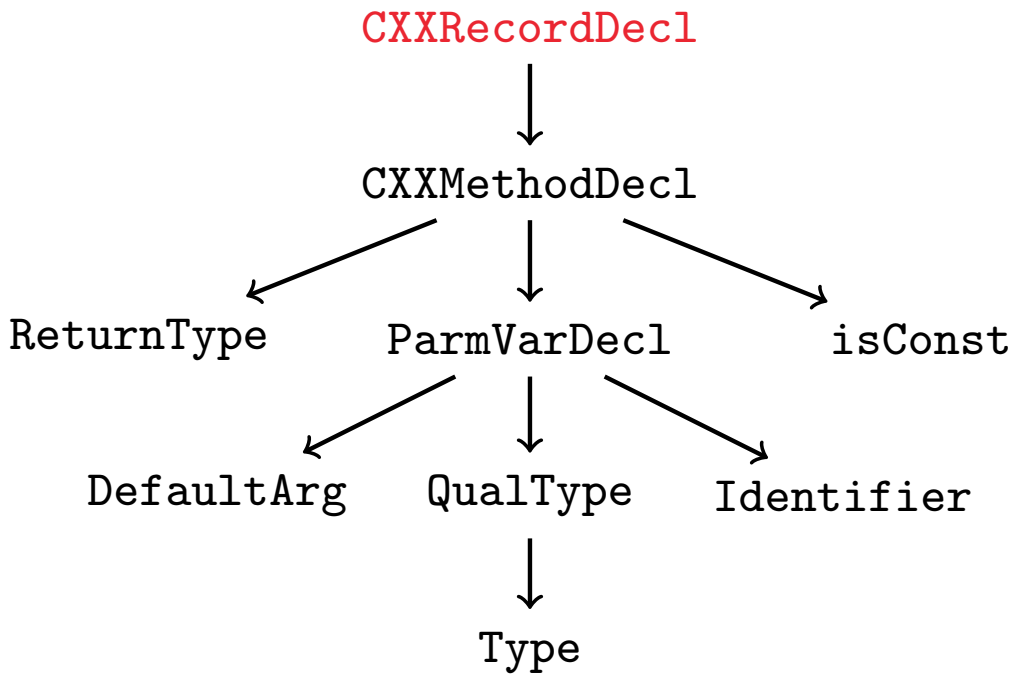
Clang: Parser

```
void n ( int arg = 42 ) const ;
```



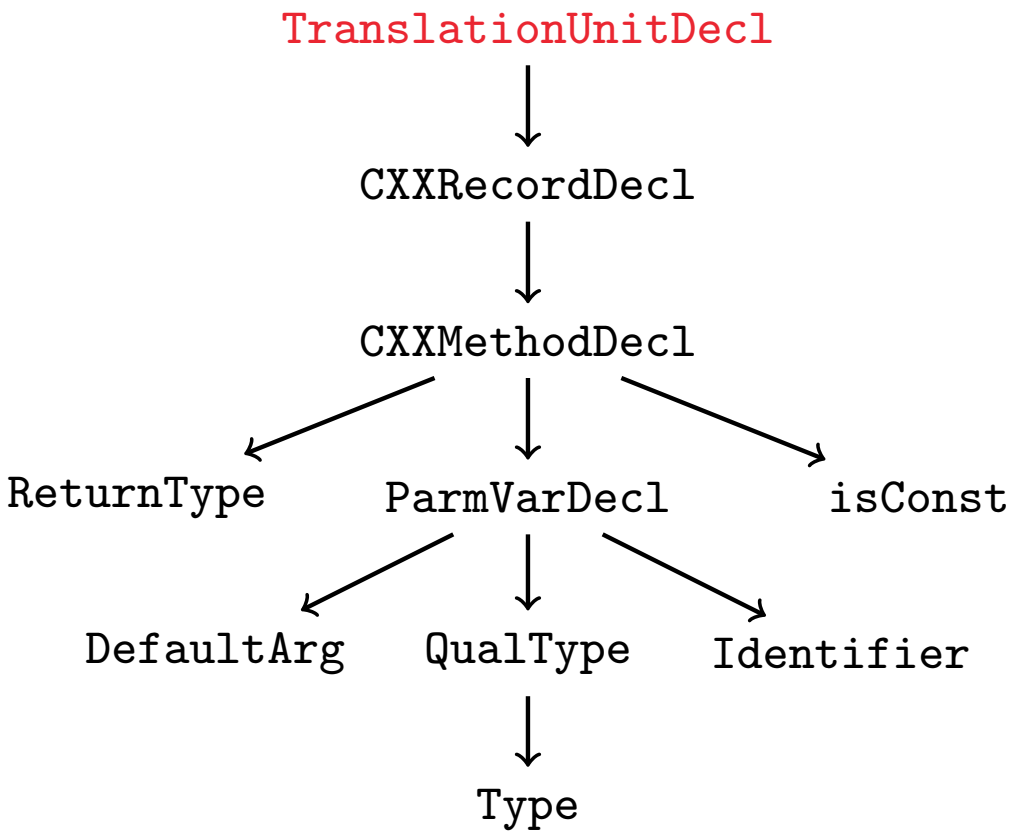
Clang: Parser

```
void n ( int arg = 42 ) const ;
```



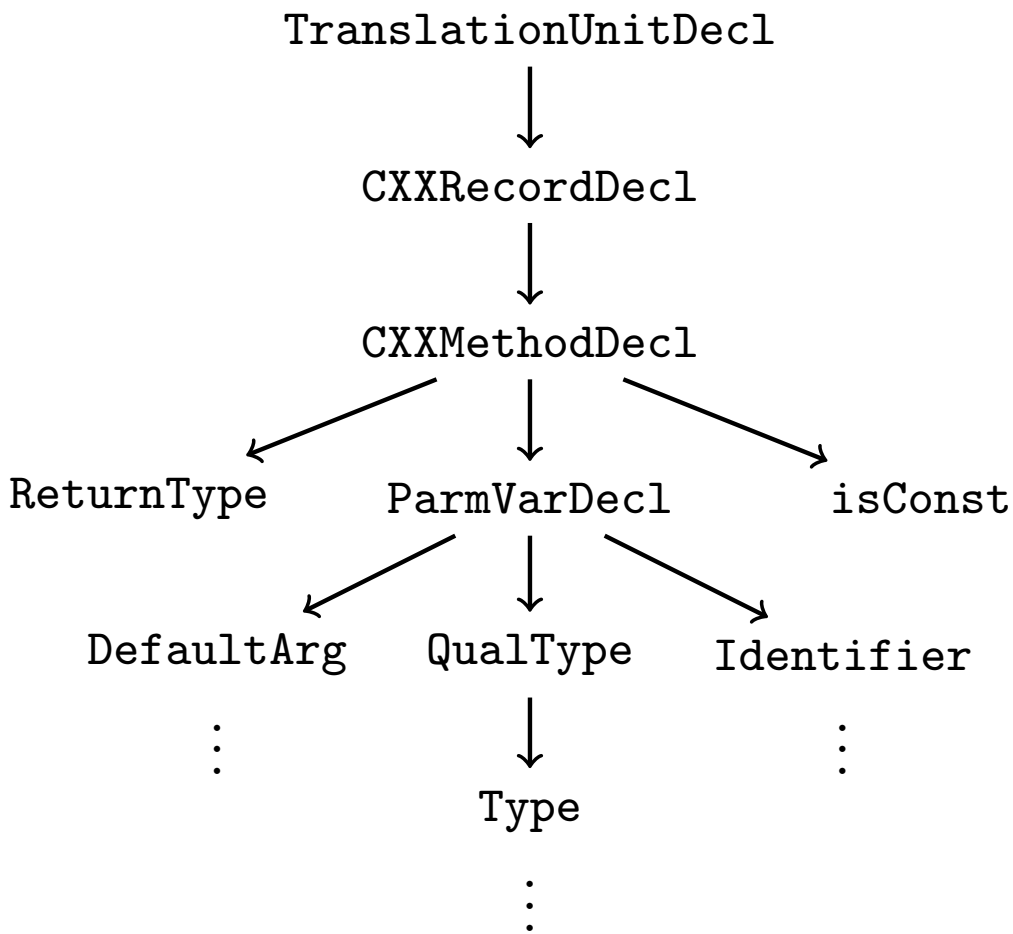
Clang: Parser

```
void n ( int arg = 42 ) const ;
```



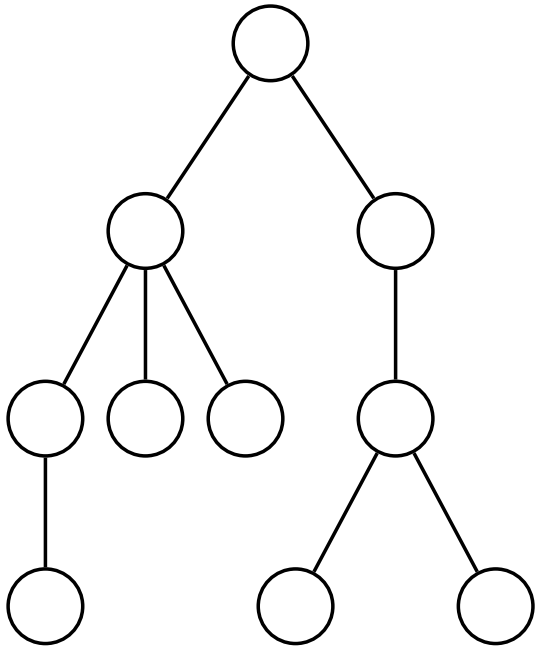
Clang: Parser

```
void n ( int arg = 42 ) const ;
```



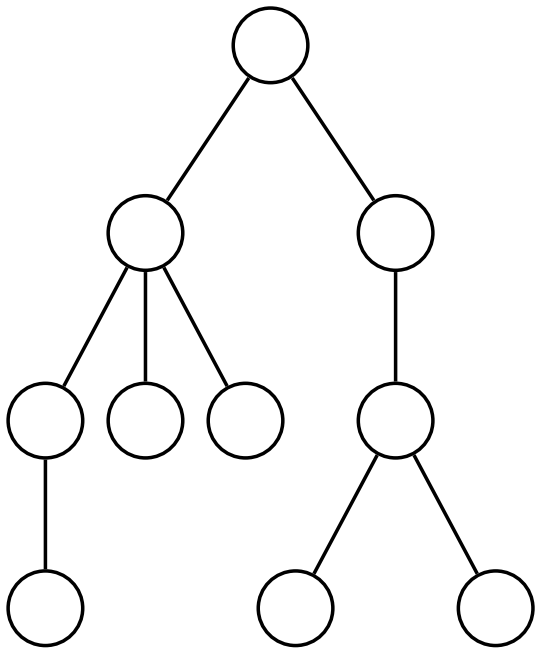
Clang: IR Generation

Clang: IR Generation



AST

Clang: IR Generation



AST

```
; Function Attrs: nounwind ssp
define i32 @_Z1fi(i32) #0 {
  %2 = alloca i32, align 4
  store i32 %0, i32* %2, align 4
  %3 = load i32, i32* %2, align 4
  ret i32 %3
}
```

Clang: AST

Clang: AST

```
if (0xc > 1) {
  const char* a = "ng";
}
```

Clang: AST

```
if (0xc > 1) {
  const char* a = "ng";
}
```

```
Stmt
```

Clang: AST

```
if (0xc > 1) {
  const char* a = "ng";
}
```

Stmt Decl

Clang: AST

```
if (0xc > 1) {
  const char* a = "ng";
}
```

```
Stmt  Decl  Expr
```

Clang: AST

```
if (0xc > 1) {
  const char* a = "ng";
}
```

Stmt Decl Expr Type

CLANG
ALL THE THINGS
memegenerator.net

Clang Tooling

libClang

Clang Tooling: libClang

```
#include <stdio.h>
#include <clang-c/Index.h>

CXChildVisitResult
visit(CXCursor cursor, CXCursor, CXClientData data) {
  const CXSourceLocation location = clang_getCursorLocation(cursor);
  if (!clang_Location_isFromMainFile(location)) {
    return CXChildVisit_Continue;
  }

  const CXString spelling = clang_getCursorSpelling(cursor);
  printf("%s", clang_getCString(spelling));
  clang_disposeString(spelling);

  return CXChildVisit_Recurse;
}
```

libTooling

Clang Tooling: libTooling

Clang Tidy

Clang Tooling: libTooling

Clang Tidy

Clang Plugin

Clang Tooling: libTooling

Clang Tidy

Clang Plugin

Clang Tool

Clang Tooling: libTooling

Clang Tidy

```
./clang-tidy -checks="*,my-check" file.cpp
```

Clang Plugin

Clang Tool

Clang Tooling: libTooling

Clang Tidy

```
./clang-tidy -checks="*,my-check" file.cpp
```

Clang Plugin

```
./clang++ -Xclang load -Xclang my-check.so \
-Xclang -add-plugin -Xclang my-check file.cpp
```

Clang Tool

Clang Tooling: libTooling

Clang Tidy

```
./clang-tidy -checks="*,my-check" file.cpp
```

Clang Plugin

```
./clang++ -Xclang load -Xclang my-check.so \
-Xclang -add-plugin -Xclang my-check file.cpp
```

Clang Tool

```
./my-check file.cpp
```

Clang Tooling: libTooling

Clang Tooling: libTooling

```
const
```

Clang Tooling: libTooling

```
const auto lambda = [] () {


};
```

Clang Tooling: libTooling

```
const auto lambda = [] (auto) {


};
```

Clang Tooling: libTooling

```
const auto lambda = [] (auto) noexcept {


};
```

Clang Tooling: libTooling

```
const auto lambda = [] (auto) noexcept {
    bool done = true;
    flip: done = !done;
    if (!done) goto flip;
};
```

Clang Tooling: libTooling

```
const auto clang_lambda = [] (auto) noexcept {
    bool done = true;
    flip: done = !done;
    if (!done) goto flip;
};
```

How do I continue?

Resources

- `eli.thegreenplace.net`
- `clang.llvm.org/docs/InternalsManual.html`
- `llvm.org/docs/ProgrammersManual.html`
- Random blogs on Google
- `goldsborough.me` & `github.com/goldsborough`
- Source Code!

Resources

- `eli.thegreenplace.net`
- `clang.llvm.org/docs/InternalsManual.html`
- `llvm.org/docs/ProgrammersManual.html`
- Random blogs on Google
- `goldsborough.me` & `github.com/goldsborough`
- Source Code!

`github.com/peter-can-talk/cpp-london-2017`

Q & A