Clang: Libraries & Tools

Max Thrun - Spring 2014

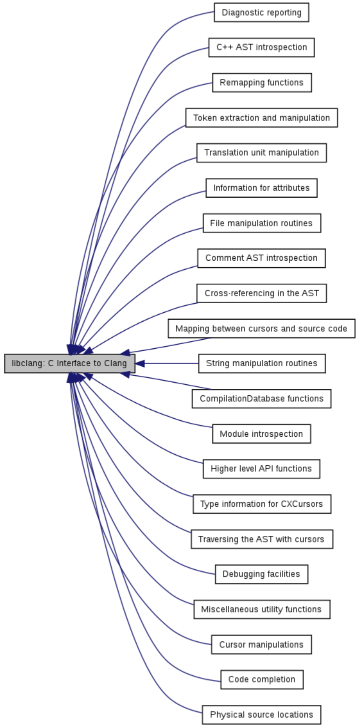
Clang

- C, C++, Objective C, and Objective C++ front-end for LLVM
- Modular library based architecture (10 libraries)
  - Allows for tight integration with IDEs
  - Easily extensible
- Focuses on being fast, low memory, and expressive errors/warnings

http://clang.llvm.org/index.html

LibClang

- High level C interface
- AST manipulation / traversing
- AST to source code association
- Diagnostic (errors/warnings) reporting
- Code completion
- Stable API
- Lots of language bindings (Python/Go/etc…)
- The go-to library



libclang: C Interface to Clang
Diagnostic reporting
C++ AST introspection
Remapping functions
Token extraction and manipulation
Translation unit manipulation
Information for attributes
File manipulation routines
Comment AST introspection
Cross-referencing in the AST
Mapping between cursors and source code
String manipulation routines
CompilationDatabase functions
Module introspection
Higher level API functions
Type information for CXCursors
Traversing the AST with cursors
Debugging facilities
Miscellaneous utility functions
Cursor manipulations
Code completion
Physical source locations

LibTooling

- C++ interface aimed at writing standalone tools
  - Syntax checkers
  - Refactoring tools
- Lower level than LibClang (direct AST access)
- Can be used in Clang Plugins

Standalone Tool Demo

- Want to insert comments before and after functions and if statements
- Need to:
    - Parse source code into an AST
    - Traverse AST
    - Rewrite the source code

http://eli.thegreenplace.net/2012/06/08/basic-source-to-source-transformation-with-clang/

Clang Plugins

- Run additional actions on AST as part of compilation
- Plugins are dynamic libraries loaded at runtime
- Easy to integrate into your build environment
- Examples
    - Special lint-style warnings
    - Creating additional build artifacts

http://clang.llvm.org/docs/ClangPlugins.html

Plugin Demo

- Want to throw error if method names are snake_case instead of CamelCase
- Need to:
  - Create an AST consumer
  - Use RecursiveASTVisitor to visit all method declarations
  - Check if it contains a ‘_’

https://github.com/AlexDenisov/ToyClangPlugin

clang-modernizer

- Convert C++ code written in older standard to use new features of newer standard
- Collection of independent transforms which can be independently enabled
- Can specify different coding styles

http://clang.llvm.org/extra/clang-modernize.html

clang-modernizer

Demo

clang-tidy / cang-check

- C++ linter
- Provides extensible framework for diagnosing and fixing typical errors
  - Style violations
  - Interface misuse
  - Bugs founds from static analysis

clang-format

- Automatically reformat C/C++/Obj-C code to match a style specification
- Figures out best place to break lines
- Fixes comment alignments
- Built on top of LibFormat
- Editor integration / Patch generation

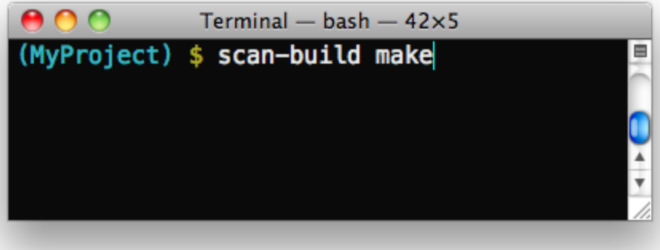
http://clang.llvm.org/docs/ClangFormat.html

http://llvm.org/devmtg/2013-04/jasper-slides.pdf

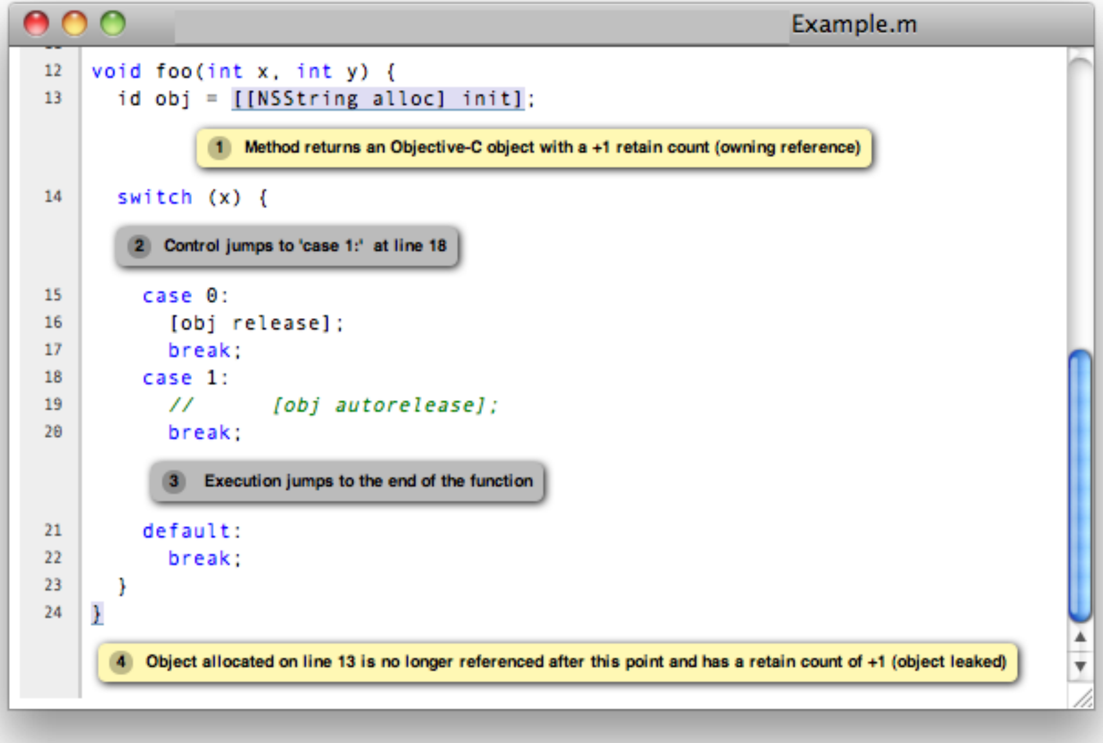
clang-format

Demo

Static Analyzer



```
(MyProject) $ scan-build make
```



```
12  void foo(int x, int y) {
13    id obj = [[NSString alloc] init];
          1  Method returns an Objective-C object with a +1 retain count (owning reference)
14    switch (x) {
        2  Control jumps to 'case 1:'  at line 18
15      case 0:
16        [obj release];
17        break;
18      case 1:
19        //      [obj autorelease];
20        break;
          3  Execution jumps to the end of the function
21      default:
22        break;
23    }
24  }
      4  Object allocated on line 13 is no longer referenced after this point and has a retain count of +1 (object leaked)
```

http://clang-analyzer.llvm.org/

Links

https://docs.google.com/presentation/d/1dg6hWuYrED6netcPlggvyq5DhjQMWdem_bRMfL_XUVk/edit?usp=sharing

https://github.com/bear24rw/EECE6083_Presentation