



# Libclang Integration in the KDevelop IDE

Kevin Funk ([kfunk@kde.org](mailto:kfunk@kde.org))

April 14, 2015 | London | EuroLLVM 2015



## About KDevelop

- A free, open-source, plugin extensible IDE
- Started in 1998 – GPL
- Cross-platform – written in C++/Qt
- Supports many languages
  - C++
  - Python, PHP, Ruby, QML/JS, ...
- Debugger integration
  - GDB, Xdebug (PHP) – *no* LLDB yet! :(
- Known for its powerful code navigation/completion support





# History of C++ Language Support

- Issues with current C++ Support
  - **Custom parser** living inside KDevelop code base
  - Over **50 000 LOC**
  - **Hard to maintain**, even harder to extend
    - Hint: C++11, C++14, ...
  - Lots of issues with non-trivial C/C++ code
    - (designated initializers, ...)
  - Not possible to disambiguate between C vs. C++, or separate C++ standard versions
  - ...



# Clang to the Rescue!

- C/C++/ObjC language frontend for LLVM
- Features
  - Expressive diagnostics
  - Allows tight integration with IDEs
  - BSD-licensed
  - Highly active community
  - Stable API via **libclang**

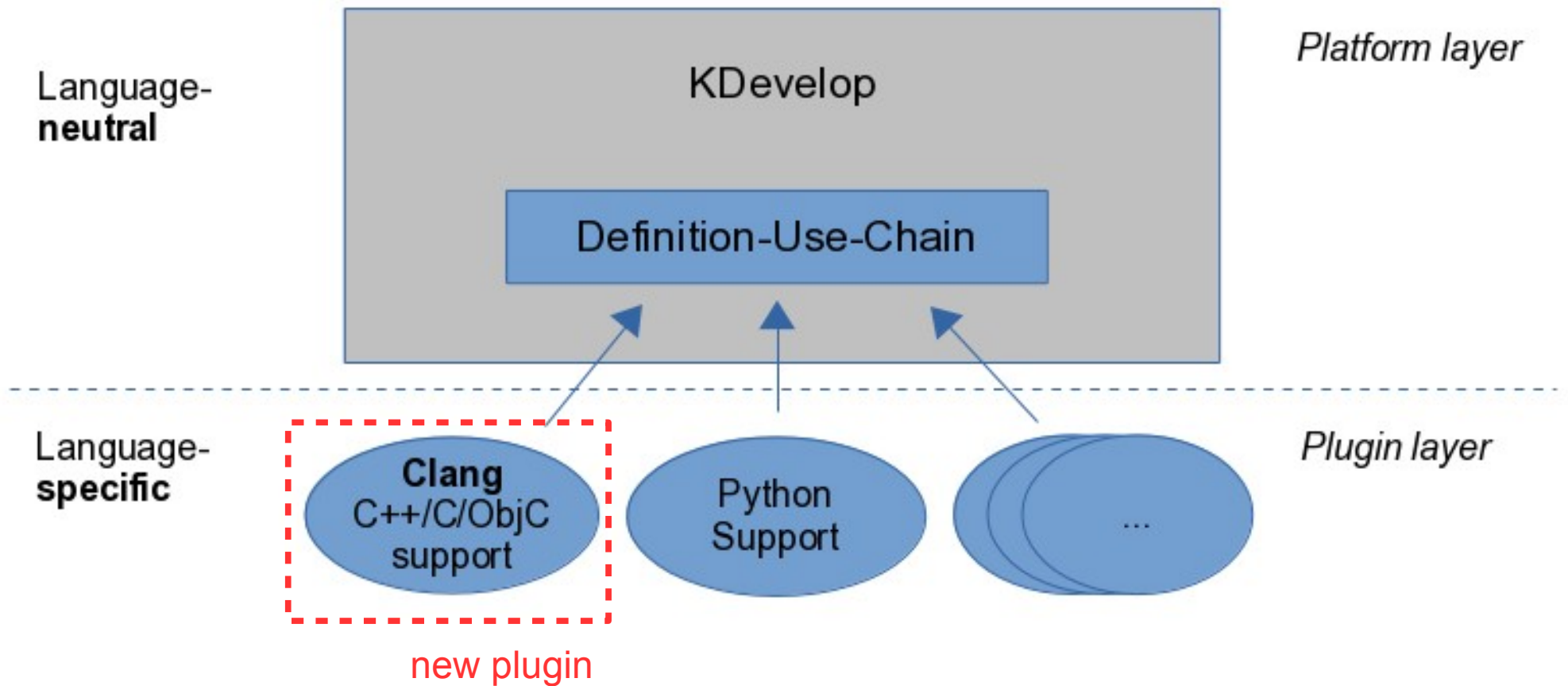




# Implementation overview



# Language Support Architecture





## Libclang Features Used

- C++ AST introspection
  - Extract definitions, uses, their attributes
- Diagnostic reporting
  - Spell-checking, fixits
- Macro definitions/expansions
- PCH generation
- Unsaved files for editor buffers
- Code completion



# Impressions





# Clang Assistants

- Providing diagnostics/fixits from Clang

The screenshot shows a code editor window titled 'main.cpp' with the cursor at 'Line: 3 Col: 12'. The code is as follows:

```
1 int main()
2 {
3   int foo
4 }
5
```

A red highlight is under the line 'int foo'. At the bottom of the editor, a 'Fix-it Hints' box is visible, containing the following text:

Fix-it Hints	1	Insert ";" at line: 2, column: 11	0	Hide
--------------	---	-----------------------------------	---	------



# Clang Code Completion

- Show viable expressions for current context

```
main.cpp Line: 6 Col: 12
1 #include <cstring>
2
3 int main()
4 {
5     int i = 1; const char* str = "c";
6     strlen();

```

Best matches

- const char \* str
- const char \* basename(const char \* \_\_filename)
- const char \* index(const char \* \_\_s, int \_\_c)
- const char \* rindex(const char \* \_\_s, int \_\_c)
- const char \* strcasestr(const char \* \_\_haystack, const char \* \_\_needle)
- const char \* strchr(const char \* \_\_s, int \_\_c)

Local

- int i
- const char \* str

Global

- char \* basename(char \* \_\_filename)



## Clang Code Completion cont'd

- Special completion: Enum-case labels

```
1 enum SomeEnum { aaa, bbb };
2
3 enum AnotherEnum { ccc, ddd };
4
5 int foo, bar;
6
7 int main()
8 {
9     SomeEnum e;
10    switch (e) {
11    case |
```

The completion popup shows the following items:

- ▶ SomeEnum ○ aaa
- ▶ SomeEnum ○ bbb
- Macros



# Clang Macro Navigation

- Show definition text and uses of Macro definitions

```
1 #define FOO_BAR(x) int x = 0; (void)x;
2
3 int main(int argc, char** argv)
4 {
5     FOO_BAR(a);
6     FOO_BAR(b);
7 }
8
```

Function macro: **FOO\_BAR(x)**, defined in [main.cpp :1](#) [Show uses](#)

Body

```
int x = 0; (void)x;
```

Uses of **FOO\_BAR**

2 uses found • [\[Expand all\]](#) • [\[Collapse all\]](#)

- mainwindow-cmake-template/src/main.cpp: 2 uses [\[Collapse\]](#)
  - Declaration
    - [Line 1](#) #define **FOO\_BAR(x)** int x = 0; (void)x;
  - In [Global](#):
    - [Line 5](#) **FOO\_BAR(a)**;
    - [Line 6](#) **FOO\_BAR(b)**;



# Clang Objective-C Support

- A little bit of Objective-C support

```
Car.m x Foundation.h x NSDebug.h x main.cpp x
1  #import <Foundation/Foundation.h>
2
3  @interface HelloWorldExample: NSObject
4  - (void) printMethod;
5  @end
6
7  @implementation HelloWorldExample
8  - (void) printMethod {
9  NSLog(@"Hello World \n");
10 }
11 @end
12
13 int main ()
14 {
15 /* Main method a starting point
16 in every Objective-C programs */
17 HelloWorldExample * foo = [[foo alloc] init];

```

class [HelloWorldExample](#) ( resolved forward-declaration: [HelloWorldExample](#))  
Kind: **Forward Declaration**  
Decl.: [Car.m :3](#) [Show uses](#)



# Clang Helpers

- Clang parsing Doxygen-style comments

```
1  /**
2   * @param bar The first parameter
3   */
4  int func(int foo)
5  {
6   ....|
7  }
8
9
```

Line: 6 Col: 5

Documents

- ...make-t
- main.

Problem in **Semantic analysis**:  
Parameter 'bar' not found in the function declaration [-Wdocumentation]

Hint: Did you mean 'foo'?

Solve: Fix-it Hints



# What's up next?



# TODO in our C++ support

- Libclang
  - Fix C++11-auto type deduction
  - Get preprocessed contents in macro expansions
  - “Identifiable” diagnostics  
Introduce: `int clang_getDiagnosticId(...)`?
- KDevelop Clang support plugin
  - Backport of our custom parser's capabilities
    - Example: Auto-transform of `.` to `->` in `string* s = ...; s.<cursor>`
  - Make it work on Windows (MSVC):  
No `constexpr` support yet! :(





## Join Us!

- **kdev-clang is manageable** (below 10 KLOC)
  - Clean, C++11-aware code base
  - Fully unit-tested
- KDevelop provides **powerful** language-agnostic interfaces, saves precious **manpower**
- Easy to implement **new assistants/warnings**, ...



**Thanks!**



# References

- **Contact**
  - IRC: #kdevelop on Freenode
  - Mailing list: kdevelop-devel@kde.org
- **GSoC: Clang Integration in KDevelop:**  
<http://kfunk.org/2014/04/28/gsoc-2014-improving-the-clang-integration-in-kdevelop/>  
(+ follow-up posts)
- **Try it out yourself:**  
<http://milianw.de/blog/katekdevelop-sprint-2014-let-there-be-clang> [Section: Take my Code]