# Indexing Large, Mixed-Language Codebases

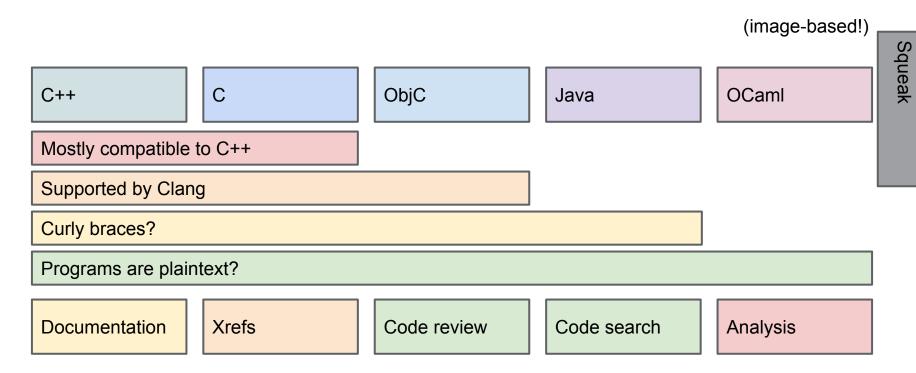
Luke Zarko <zarko@google.com>

# The Kythe project aims to establish open data formats and protocols for interoperable developer tools.

# Outline

- Introduction
- System structure
- C++ support via Clang
  - What does Kythe get?
  - What does Kythe propose to give back?
- Future work

# I use languages with property **X** and I'd like to do **Y**

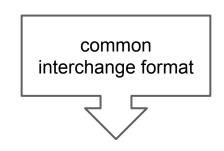


# I also use source code generator X, build system Y, repo Z

Google

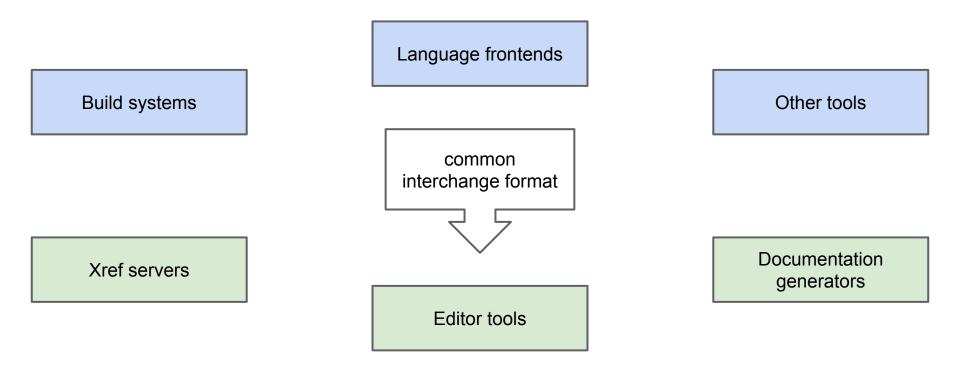
protobuf	cmake	git
thrift	gmake	svn
cap'n proto	omake	CVS
уасс	mvn	company filer
antlr	a bunch of shell scripts	local disk
jni?	ant?	someone's :80?





Documentation	Xrefs	Code review	Code search	Analysis
---------------	-------	-------------	-------------	----------

## I use tools that support Kythe data



# Outline

- Introduction
- System structure
- C++ support via Clang
  - What does Kythe get?
  - What does Kythe propose to give back?
- Future work

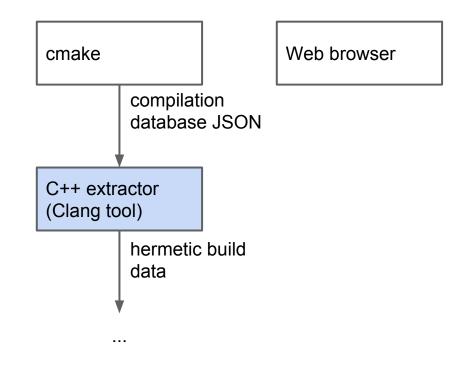
# A Kythe system

cmake

Web browser

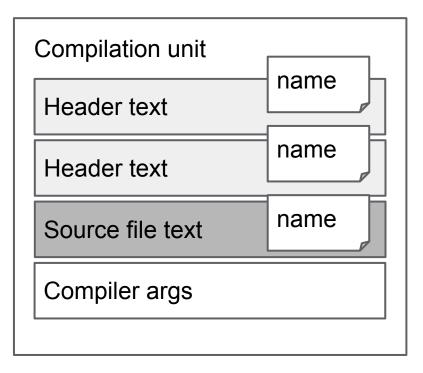
# A Kythe system

• Extractors pull compilation information from the build system



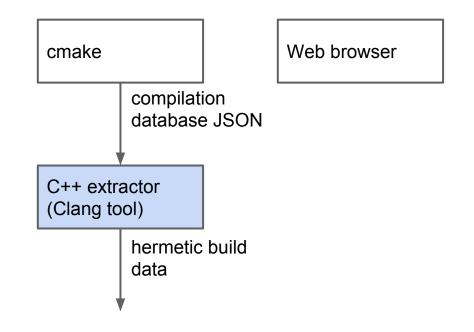
### Hermetic build data

- Contains every dependency the compiler needs for semantic analysis
- Gives files identifiers that can be used to locate them in repositories
- Allows for distribution of analysis tasks



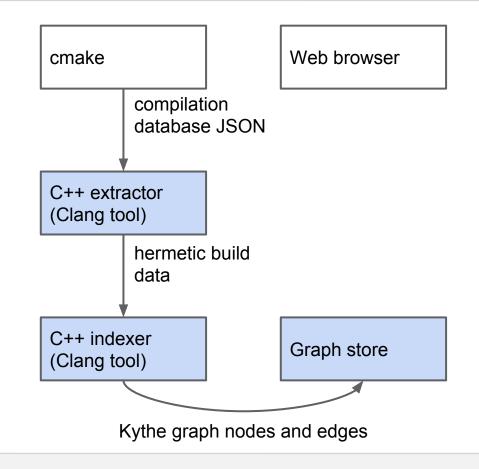
# A Kythe system

• Extractors pull compilation information from the build system



# A Kythe system

- Extractors pull compilation information from the build system
- Indexers use this information to construct a persistent graph



### Indexer implementation

- 1. Load hermetic build data into memory with mapVirtualFile
- 2. First pass: recover parent relationships for naming

### Nameless decls and shadowed names

 Clang omits parent edges in the AST because it doesn't need them

Google

- As best we can, we want to give stable names to any Decl we see referenced at any point
- We also want to distinguish between shadowed names
- Solution: build a map from AST nodes to (parent, visitation-index)\*

void foo() { x:0:0:fooint x; x:0:1:0:foo { int x; } x:0:2:0:foo { int x; }

### Indexer implementation

- 1. Load hermetic build data into memory with mapVirtualFile
- 2. First pass: recover parent relationships for naming

### Indexer implementation

- 1. Load hermetic build data into memory with mapVirtualFile
- 2. First pass: recover parent relationships for naming
- 3. Second pass: notify a GraphObserver about abstract program relationships

# The Kythe graph

All programs in Kythe are abstracted away to nodes and edges.

(some, unique, name)		
/kythe/node/kind	record	
/your/own/fact	some string	

# The Kythe graph

Nodes represent semantic information as well as syntactic information.

(some, unique, name)		
/kythe/node/kind record		
/your/own/fact	some string	

the class C

/kythe/edge/defines

"class C	" in a	particular	file
----------	--------	------------	------

(another, unique, name)

/kythe/node/kind	anchor

# The Kythe schema

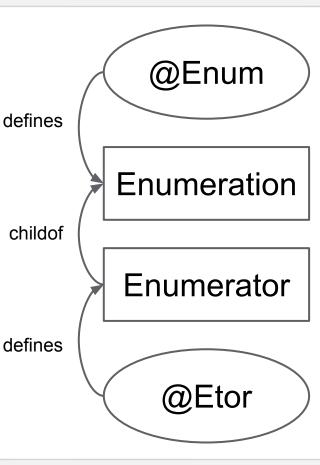
- We provide a base set of nodes and edges
- We also provide rules for naming certain kinds of nodes
- It is extensible: you're free to use your own node and edge kinds
- "Be conservative in what you send, be liberal in what you accept"
  - some data may be missing
  - $\circ$   $\,$  there may be more data than you can understand
  - others may produce incorrect data

### The schema provides checked examples

//- @Enum defines Enumeration
enum class Enum {
//- @Etor defines Enumerator
Etor

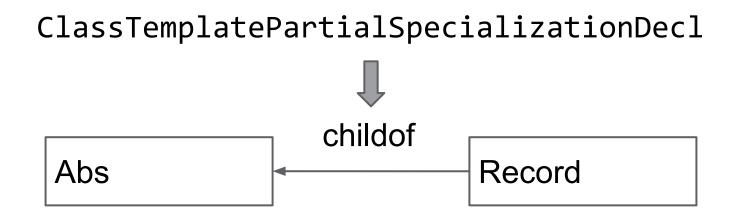
};

//- Enumerator childof Enumeration



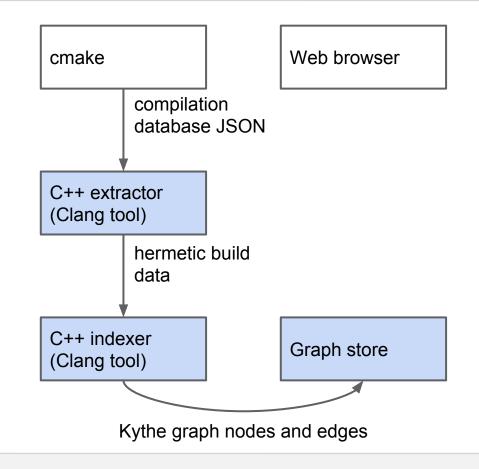
# The GraphObserver is notified about program structure

- The GraphObserver interface sees an abstract view of a program
- There is not a 1:1 mapping between AST nodes and program graph nodes



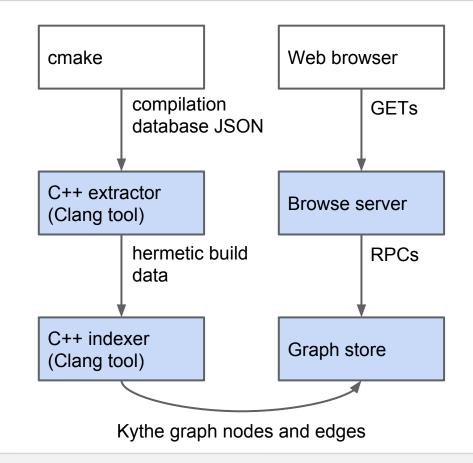
# A Kythe system

- Extractors pull compilation information from the build system
- Indexers use this information to construct a persistent graph



# A Kythe system

- Extractors pull compilation information from the build system
- Indexers use this information to construct a persistent graph
- Services use the graph to answer queries
  - $\circ$  code browsing
  - $\circ$  code review
  - documentation generation



# This design is known to scale

- Small dataset (Chromium)
  - ~22,600 C++ compilations
  - ~31G of serving data
- Internal code search is much larger
  - 100 million lines of code
- Other internal tools make use of build data for analysis

49       uint32_t*         6 shell       50         8 UlLDgn       51         DEPS       52         OWNERS       54         arguments.cc       56         araray_buffer.cc       58         array_buffer.cc       60         array_buffer.cc       61         converter.cc       64         converter.cc       64         converter.unittest.cc       66         debug_impl.cc       66         debug_impl.h       66         dictionary.cc       71         return Number::New(isolate, static_cast         71       return Number::New(isolate, static_cast		
Search Code		e web forward.
[chromium] // src / gin / converter.cc         Files   Outline       Converter.cc         @ modules       47         @ public       47         @ shell       50       if (lval->IsUint32_t>::FromV8(Isolate*:uint32_t*         @ shell       50       if (lval->IsUint32())         @ test       50       if (lval->IsUint32())         @ test       50       if (lval->IsUint32Value();         BUILD.gn       52       *out = val->Uint32Value();         DEPS       55       55         OWNERS       56       Handle <value> Converter<int64_t>::ToV8(Isolate* in int64_t* or int66_t* or int64_t* or int64_t* or int64_t* or int64_t* or int64_t* o</int64_t></value>	Project Home Downloads Wiki Issues	Code Search
Files   Outline       Converter.cc         @ modules       47         @ public       48         b shell       50         @ test       51         BUILD.gn       51         DEPS       54         OWNERS       55         README       56         arguments.cc       57         arguments.h       61         converter.cc       58         converter.cc       61         converter.cc       62         if (1val->IsNumber:)New(isolate, static_cast<		Search Code
Interformed       47         @ modules       47         @ public       48         @ shell       11         @ shell       50         If (1val->IsUint32())         @ test       50         BUILD.gn       51         DEPS         OWNERS         README         arguments.c         arguments.h         aray_buffer.cc         array_buffer.h         converter.cc         converter.cc         converter.cc         debug_impl.c         debug_impl.h         dictionary.cc         dictionary.h	[chromium] // src / gin / converter.cc	
Windows       48       bool Converter <uint32_t>::FromV8(Isolate*:         @ public       48       bool Converter<uint32_t>::FromV8(Isolate*:         @ shell       50       if (!val-&gt;IsUint32())         @ test       50       return false;         BUILD.gn       51       return false;         DEPS       54       }         OWNERS       55         README       556         arguments.c       57         array_buffer.cc       56         array_buffer.h       50         converter.cc       60         bool Converter<int64_t>::FromV8(Isolate* in int64_t* on int64_</int64_t></uint32_t></uint32_t>	Files   Outline	converter.cc
converter.h       65       // the full precision of int64_t, which n         converter.h       66       *out = val->IntegerValue();         converter_unittest.cc       67       return true;         debug_impl.cc       68       }         debug_impl.h       69       roture val->Converter <uint64_t>::ToV8(Isc         dictionary.cc       71       return Number::New(isolate, static_cast         dictionary.h       72       }</uint64_t>	public     public     shell     test     BUILD.gn     DEPS     OWNERS     README     arguments.cc     arguments.h     array_buffer.cc     array_buffer.h	<pre>48 bool Converter<uint32_t>::FromV8(Isolate* i 49 uint32_t* 50 if (!val-&gt;IsUint32()) 51 return false; 52 *out = val-&gt;Uint32Value(); 53 return true; 54 } 55 Handle<value> Converter<int64_t>::ToV8(Isol 57 return Number::New(isolate, static_cast<cast) 58 } 59 bool Converter<int64_t>::FromV8(Isolate* is 61 int64_t* ou 62 if (!val-&gt;IsNumber()) 63 return false;</int64_t></cast) </int64_t></value></uint32_t></pre>
bb     voit = val-> <u>integervalue();</u> converter_unittest.cc     67_/2       debug_impl.cc     68 }       debug_impl.h     69       dictionary.cc     71       dictionary.h     72 }		
debug_impl.cc     68       debug_impl.h     69       dictionary.cc     71       dictionary.h     72		
debug_impl.h     70     Handle <value> Converter<uint64_t>::ToV8(Isc       dictionary.cc     71     return Number::New(isolate, static_cast       dictionary.h     72     }</uint64_t></value>	-	68 }
<pre>dictionary.cc 71 return Number::New(isolate, static_cast<d 72="" dictionary.h="" pre="" }<=""></d></pre>	debug_impl.h	
	dictionary.cc	_ `
	dictionary.h	72 }

# Outline

- Introduction
- Rough system structure
- C++ support via Clang
  - What does Kythe get?
  - What does Kythe propose to give back?
- Future work

# Clang made C++ tooling possible

- A tooling-friendly compiler leads to an ecosystem of software tools
  - ASan, TSan, MSan
  - clang-format, clang-tidy
  - Doxygen libclang integration
- Clang's code is eminently hackable
  - The interface to the typed AST is clean
  - $\circ$   $\,$  The preprocessor is easy to tool as well  $\,$

Clang has excellent template support

```
template <typename T> class C
{ typename T::Foo foo; }; // ClassTemplateDecl (of CXXRecordDecl)
```

template <typename S> class C<S\*>
{ typename S::Bar bar; }; // ClassTemplatePartialSpecializationDecl

template <> class C<int> { }; // ClassTemplateSpecializationDecl

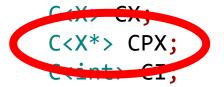
```
C<X> CX;
C<X*> CPX;
C<int> CI;
```

// implicit ClassTemplateSpecializationDecl

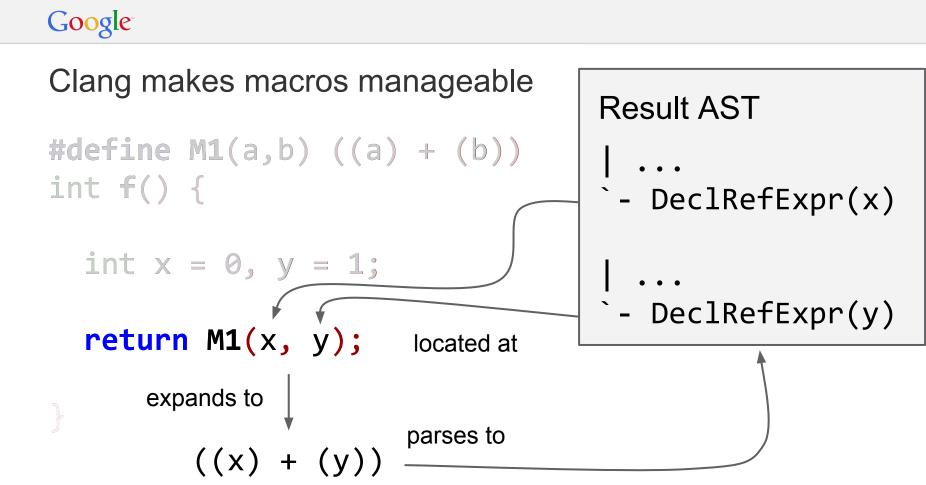
### Clang has excellent template support

```
template <typename T> class C = getSpecializedTemplate
{ typename T::Foo foo; };
```

template <typename S> class C<S\*> = getSpecializedTemplateOrPartial
{ typename S::Bar bar; };



.getTemplateArgs => { X\* } "template <X\*=T> class C" .getTemplateInstantiationArgs => { X } "template <X=S> class C<X\*>"



# Clang supports other compilers' extensions: GCC

- We want to index real world code!
- Just some of the GCC extensions clang supports:
  - o indirect-goto (goto \*bar;)
  - o address-of-label (void \*bar = &&foo;)
  - statement-expression

Google

(string s("?"); ({for(;;); s;}).size();)

- conditional expression without middle operand (f() ? : g())
- case labels with ranges (case 'A' ... 'Z':)
- ranges in array initializers

int a[] = { [0 ... 9] = 1, [10 ... 99] = 2, [100] = 3 };

# Clang can build extension-heavy software

- Building the Linux kernel works (modulo some patches: <u>http://llvm.</u> <u>linuxfoundation.org/index.php/Main\_Page</u>)
- Hairiest GCC "feature" unsupported: variable length arrays in structs

struct {struct shash\_desc shash; char ctx[crypto\_shash\_descsize(tfm)];} desc;

 Support for MSVC extensions (and ABI...) is developing too; some success with Chromium on Windows (<u>https://code.google.</u> <u>com/p/chromium/wiki/Clang</u>)

# Kythe adds to Clang's tooling support

- Persistence for abstract program data: records, not CXXRecordDecls.
- Hermetic storage of compilation units
- Unambiguous naming for more program entities
- Abstract AST traversal

### C++ is a first-class citizen

- The Kythe schema is intended to support all of C++14 (templates, (generic) lambdas, auto, ...)
- We expect support for Concepts Lite will not be difficult
- To get this into Clang:
  - Nothing Kythe-specific goes into the LLVM tree
  - Just a library in clang/tools/extra that calls appropriate members on an abstract GraphObserver
  - The Kythe indexer is a particular implementation of GraphObserver

# Outline

- Introduction
- System structure
- C++ support via Clang
  - What does Kythe get?
  - What does Kythe propose to give back?
- Future work

# Things left to do

- UI/IDE integration
- Support for other languages
  - Including one or two that are supported by Clang already
- Other analyses that work over or contribute to the graph
  - Use Kythe information as sparse data to drive whole-project analysis
- Adding more build information (eg, who links to whom)
- Quick incremental updates

# Summary

- The open Kythe data format enables interoperable tooling
- The Kythe pipeline is designed to scale
- C++ support is possible thanks to the work done on Clang tooling
- Simpler languages (Go, Java) aren't necessarily easier to tool
- The code we will propose to upstream does not depend on Kythe
- There are lots of opportunities for community development

# Mailing list

https://groups.google.com/forum/#!forum/kythe-early-interest