Index-While-Building and Refactoring in Clang

Alex Lorenz and Nathan Hawes, Apple.

Index-While-Building

Indexing, historically.

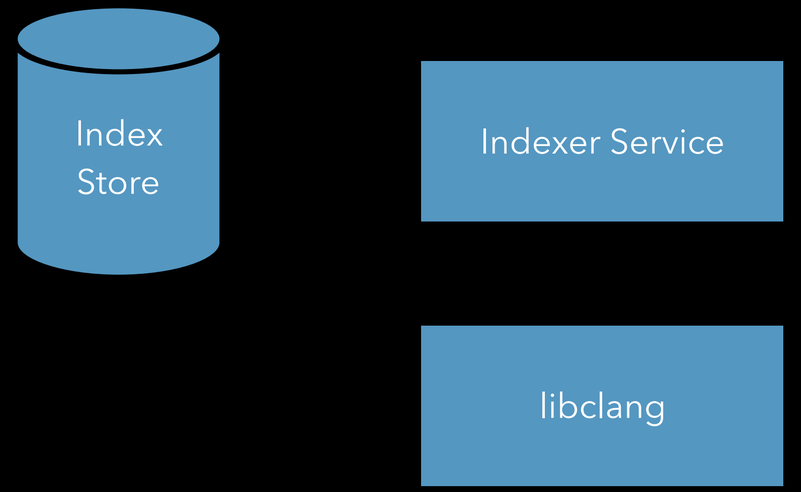
Indexing while building.

Its implementation in Clang.

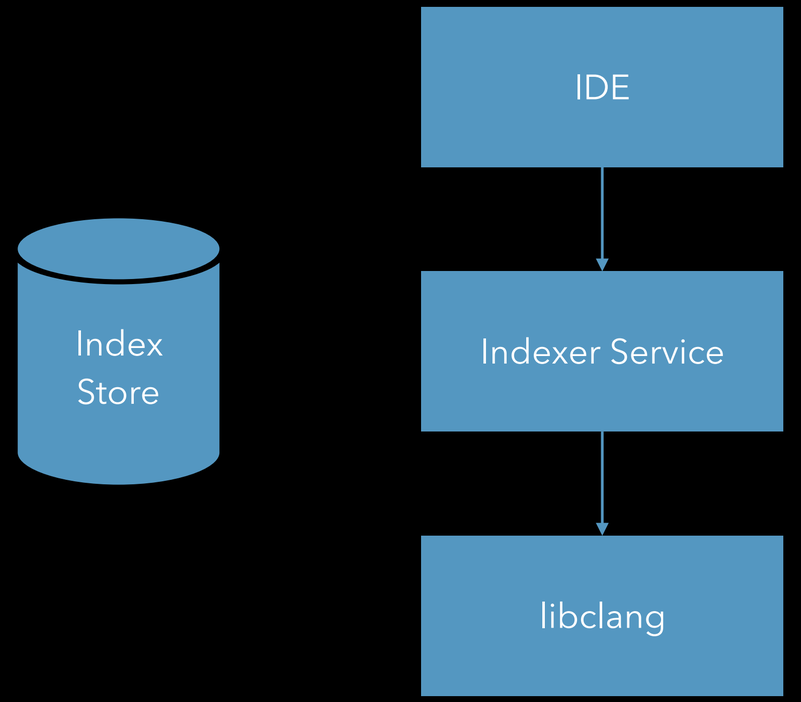
What the produced index data looks like.

How we use it in the indexer service.

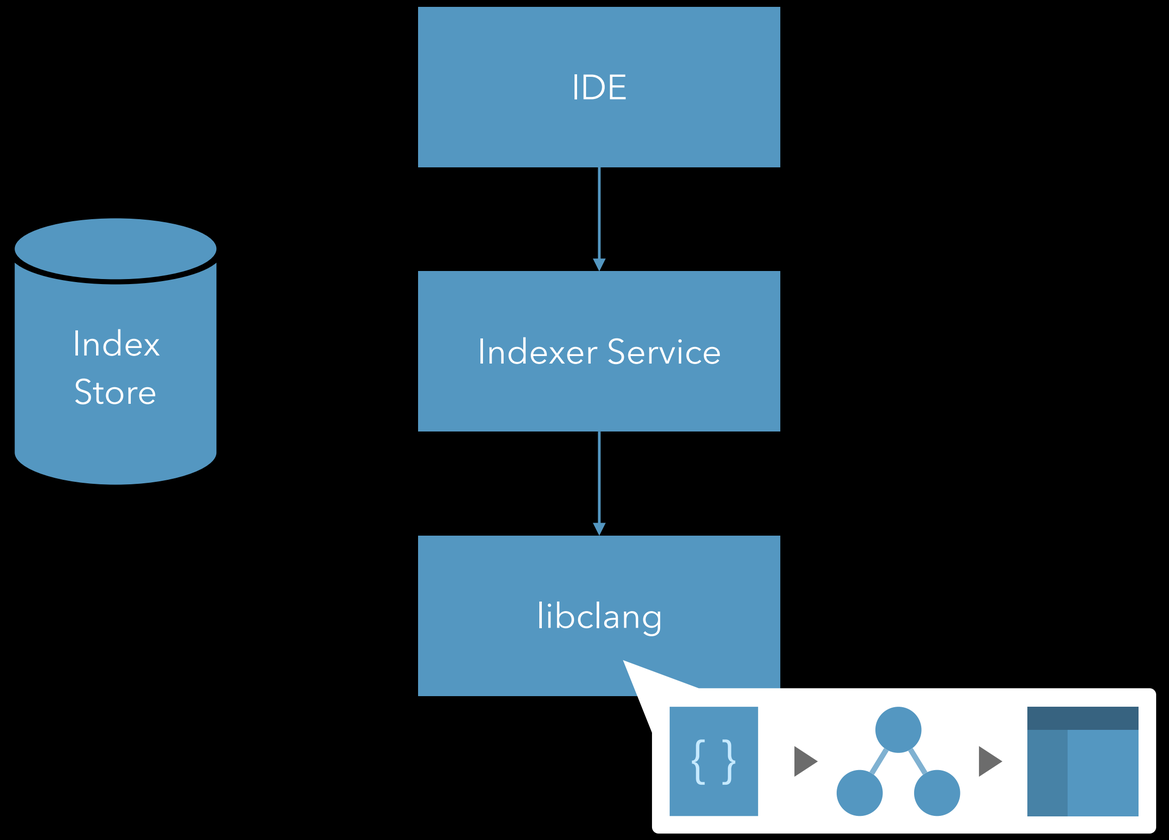
Indexing
in the background



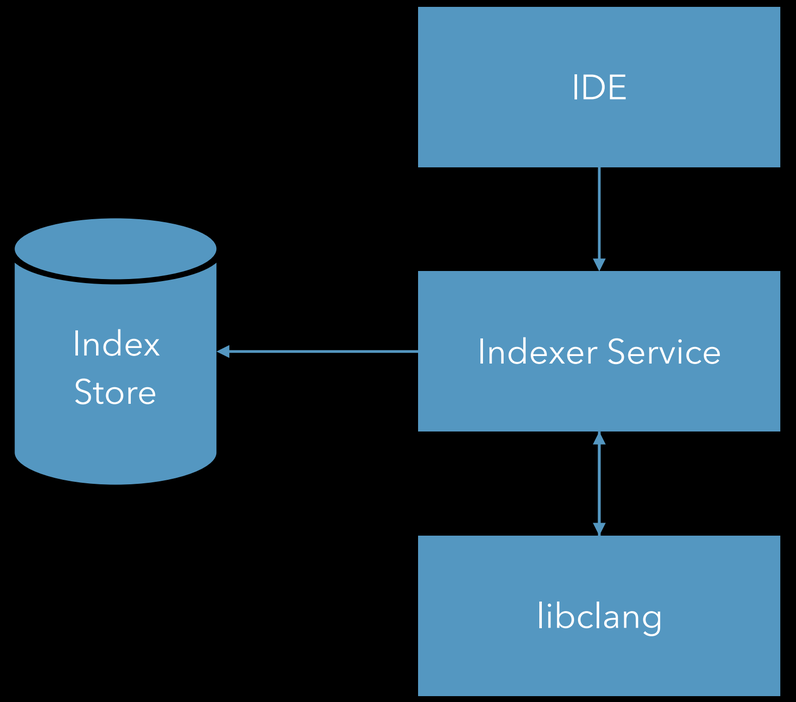
Indexing
in the background



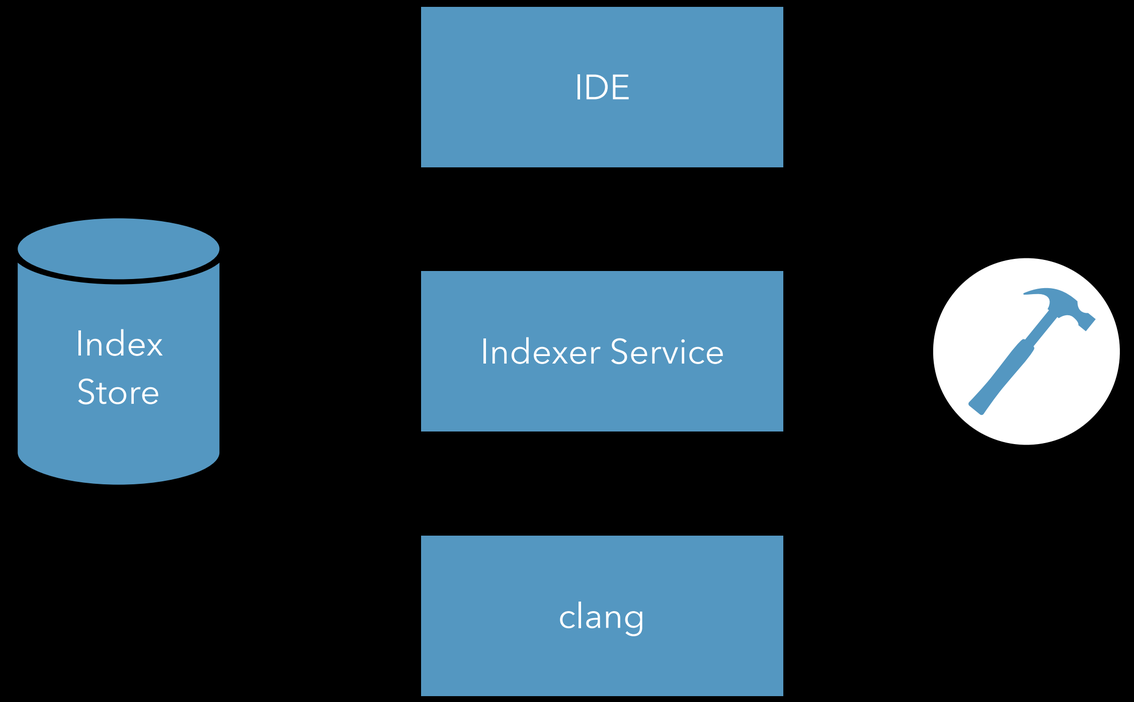
Indexing
in the background



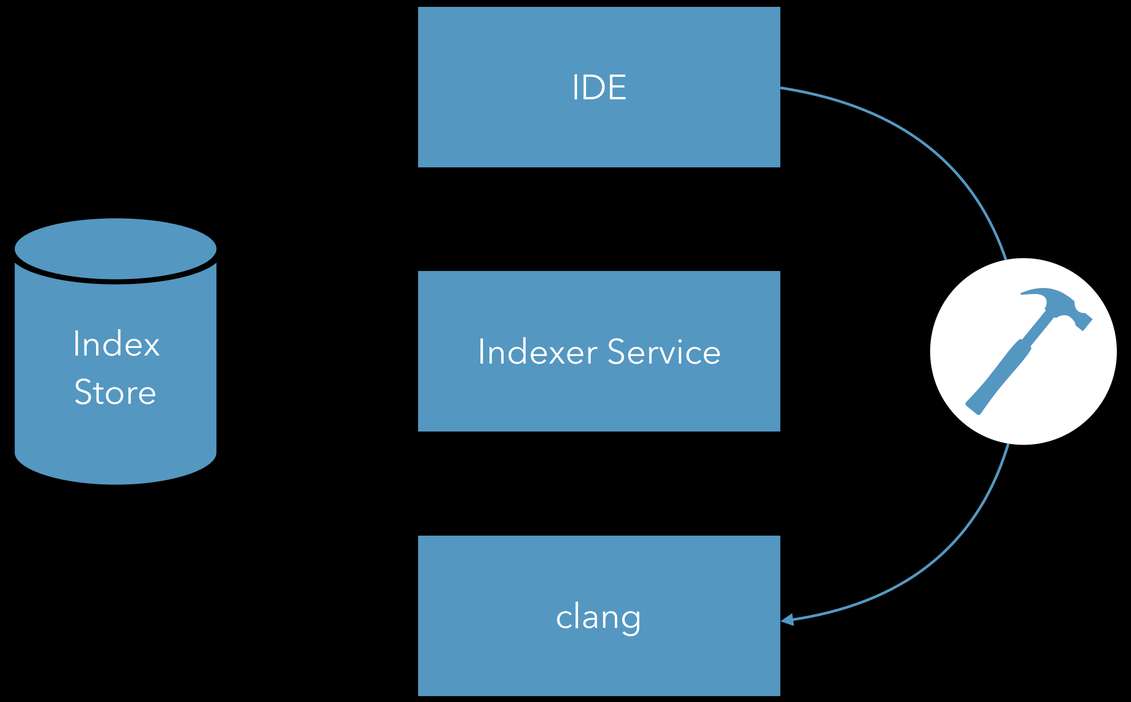
Indexing
in the background



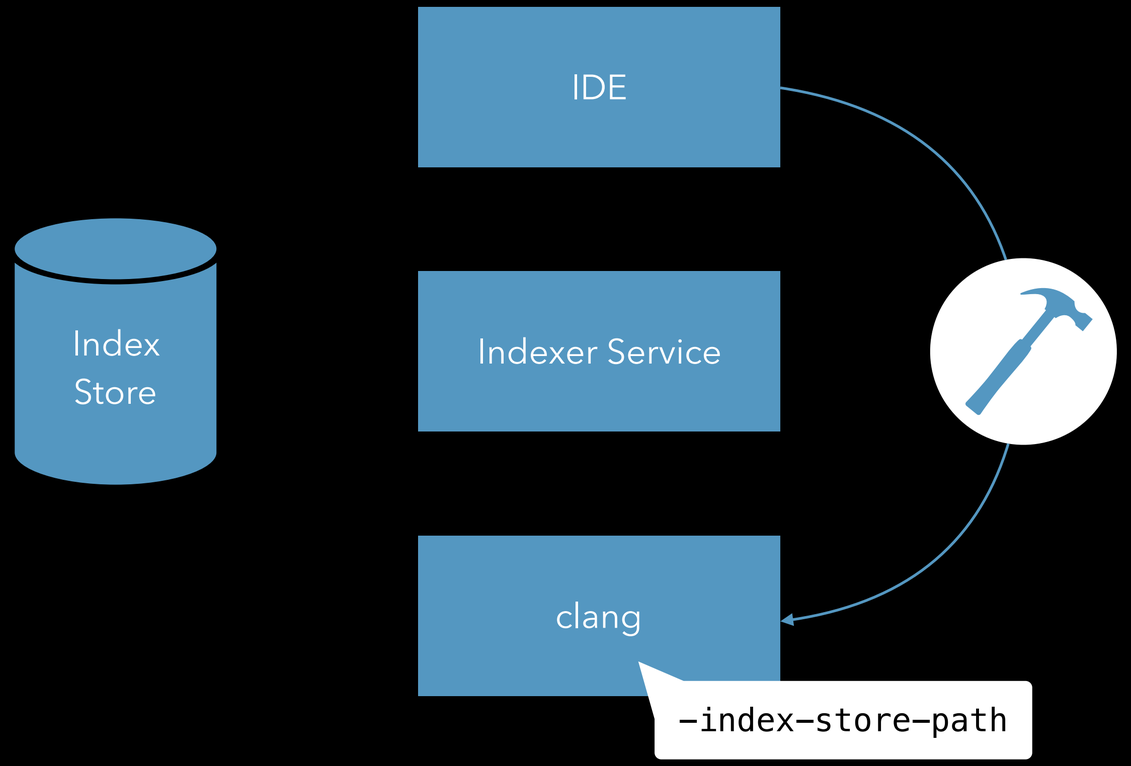
Indexing while building



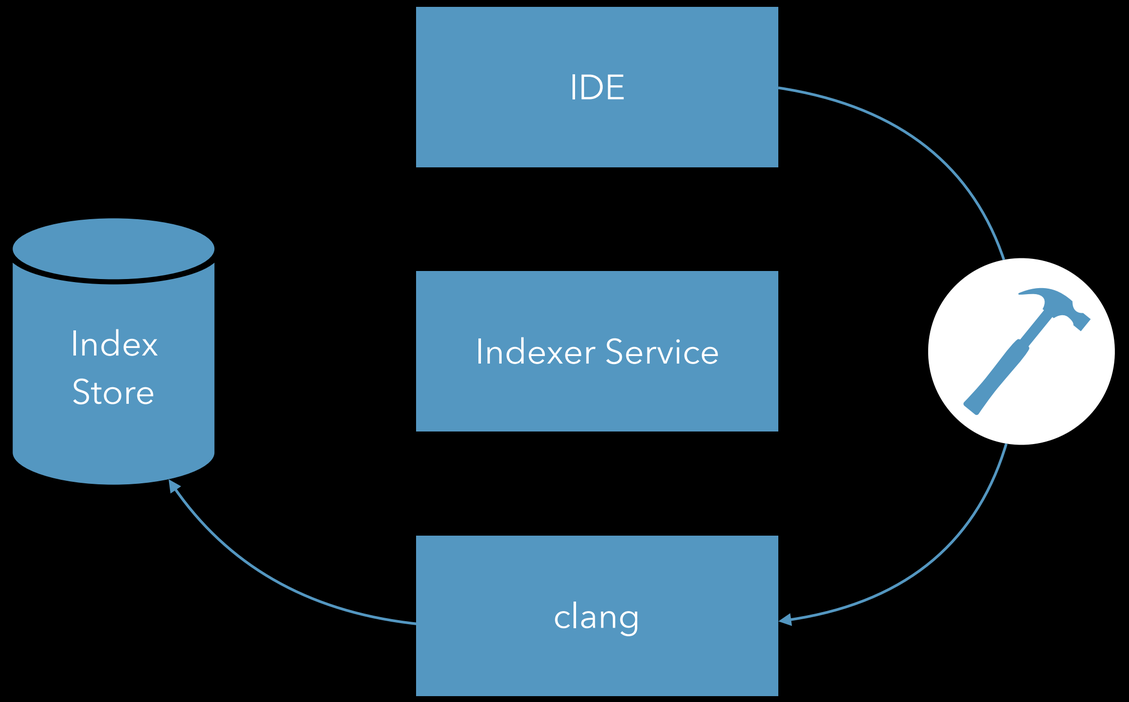
Indexing while building



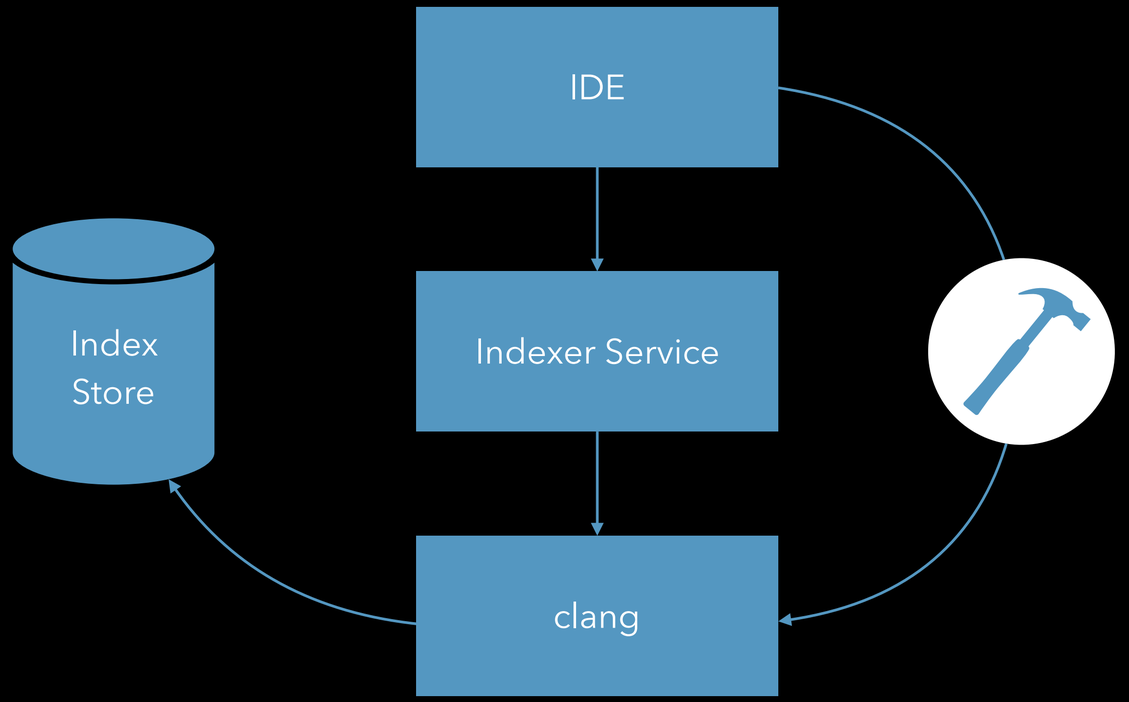
Indexing while building



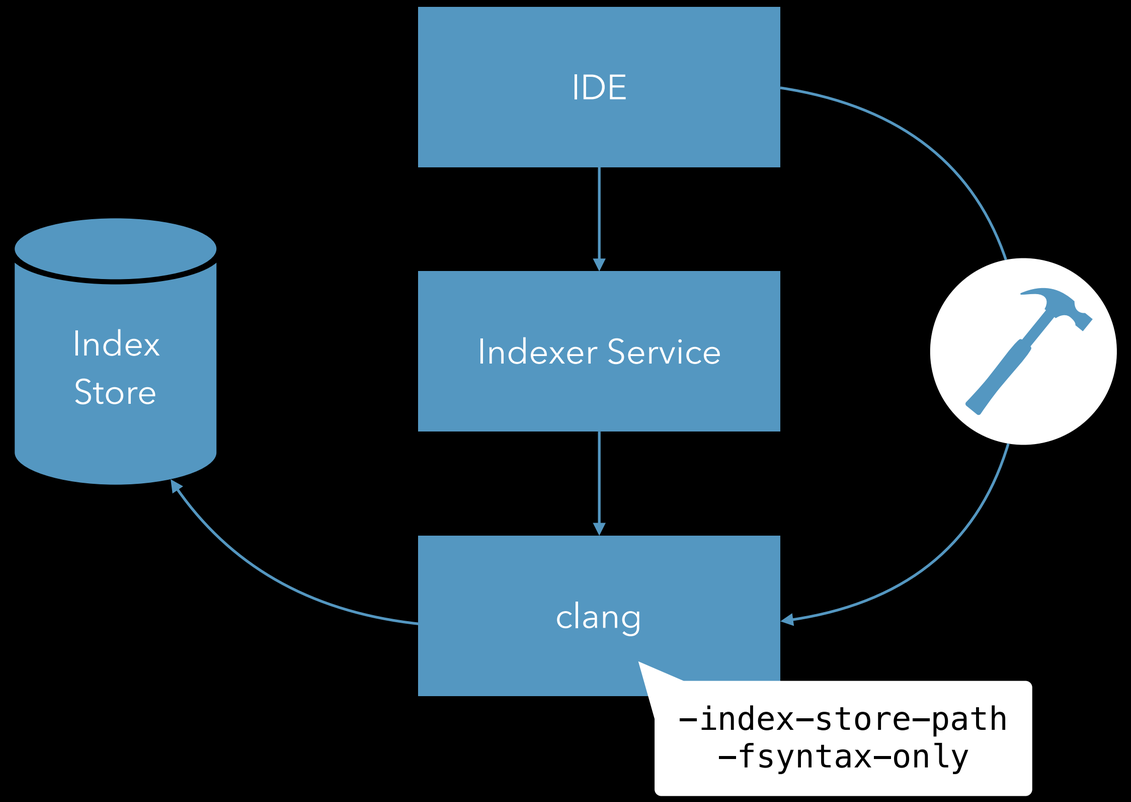
Indexing while building



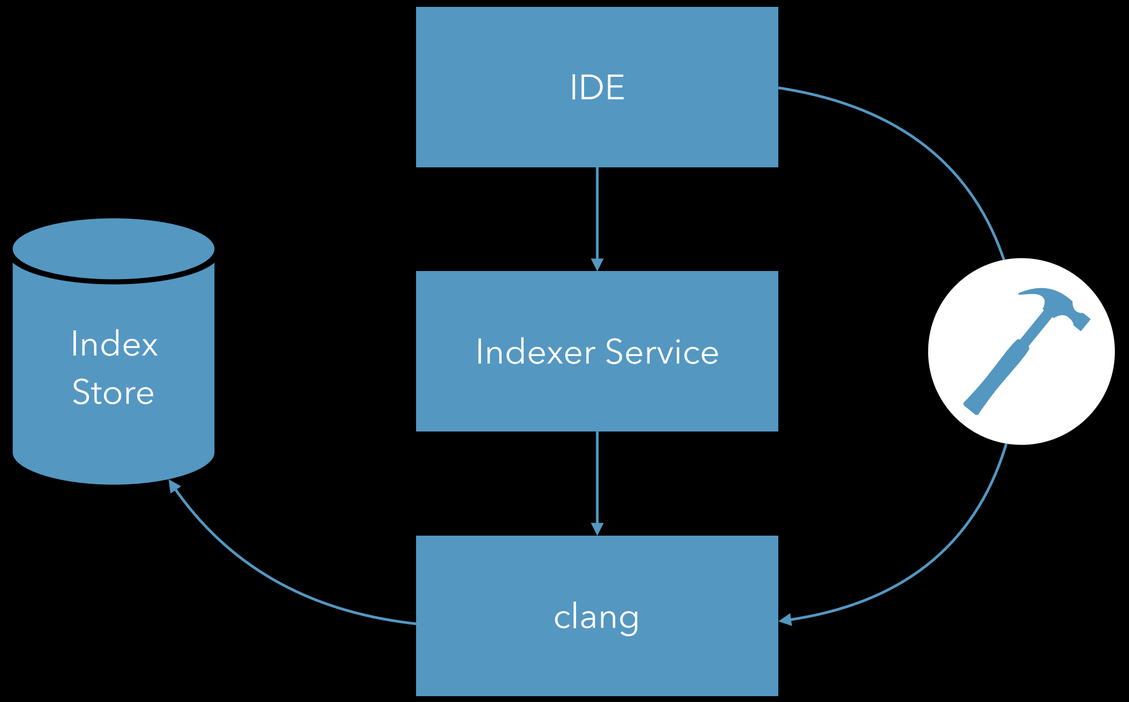
Indexing while building



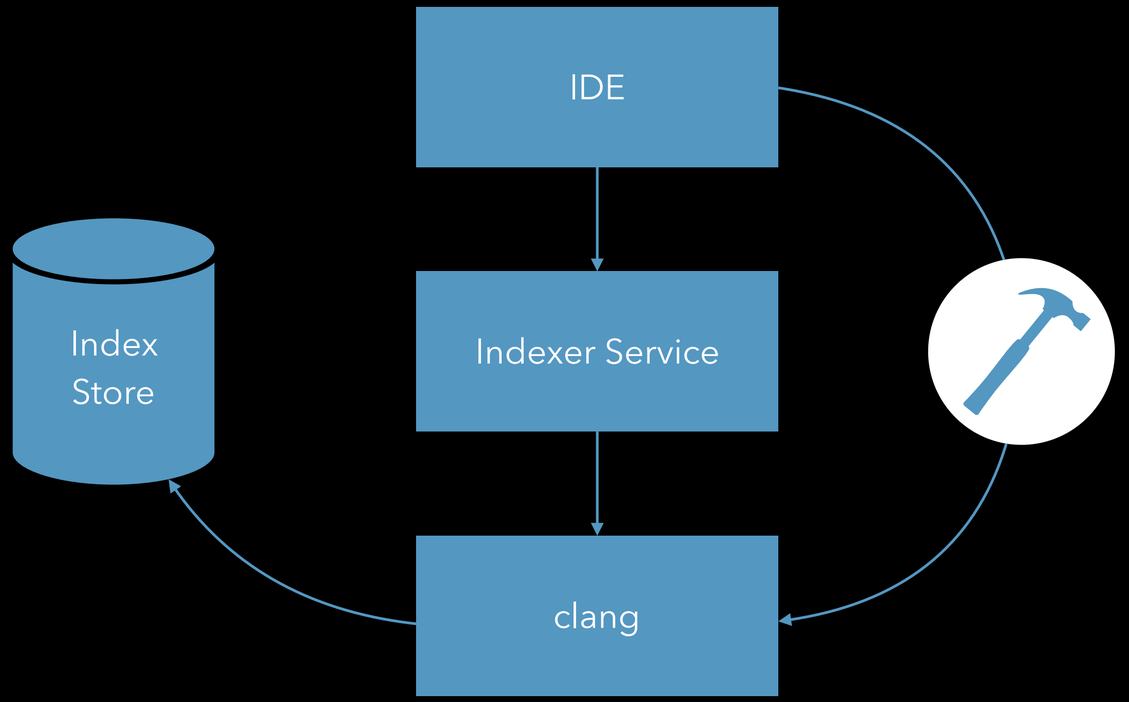
Indexing while building



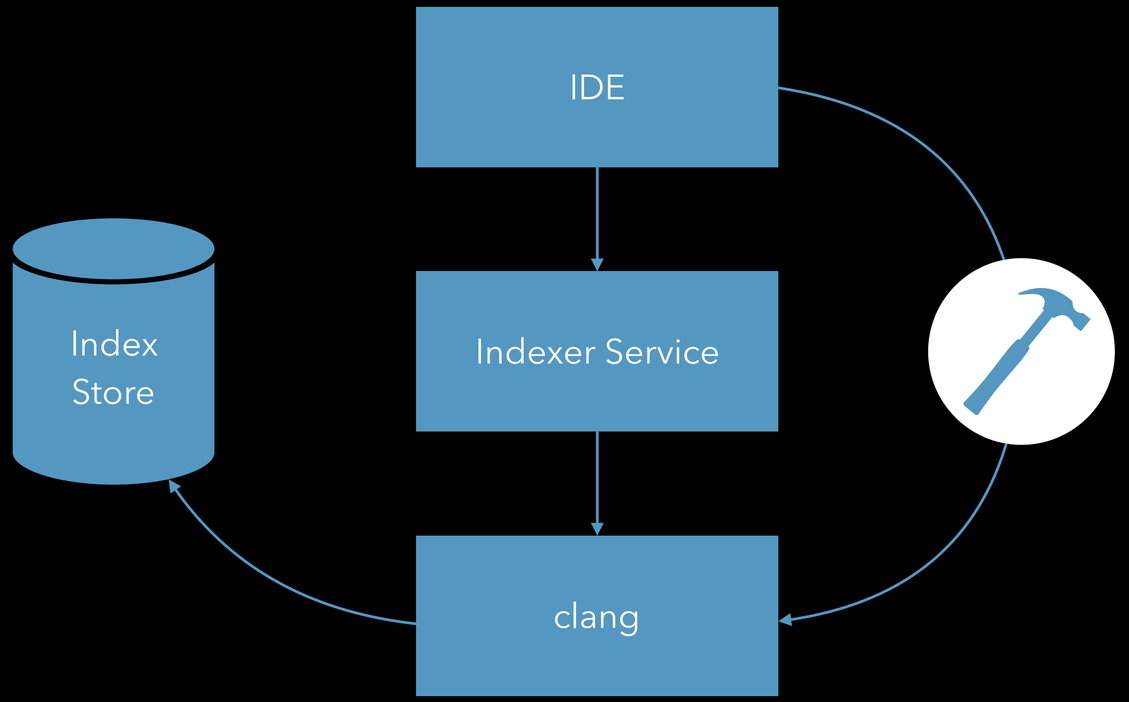
Indexing while building



Indexing
while building

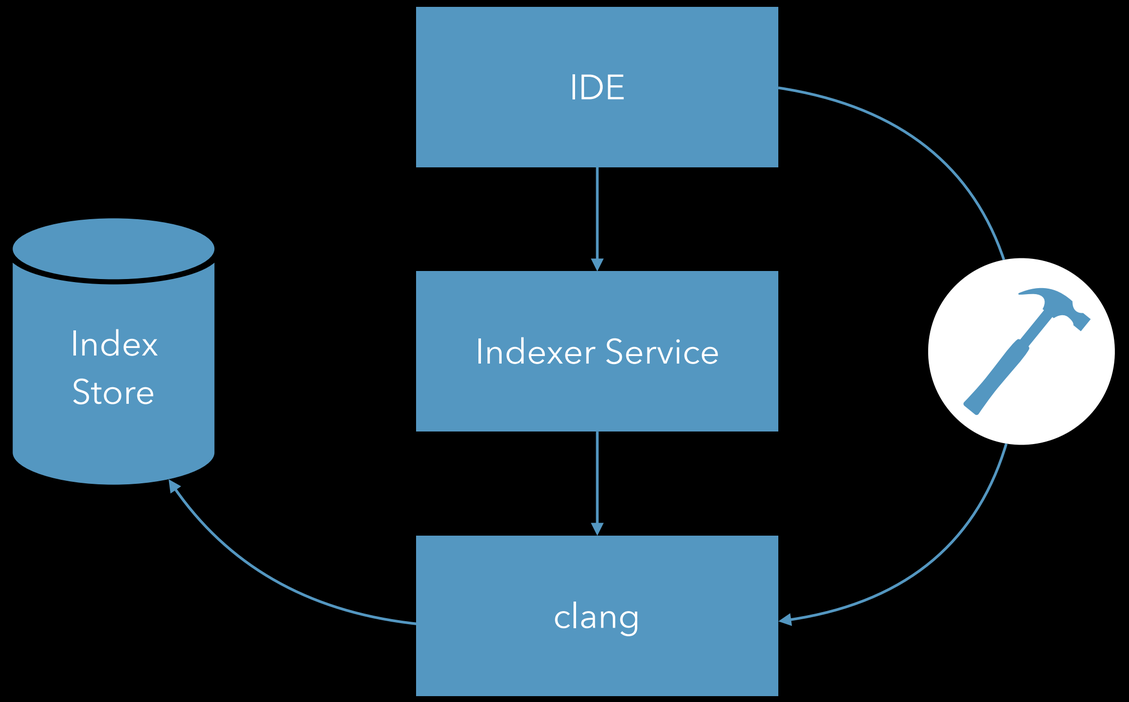


Indexing
while building



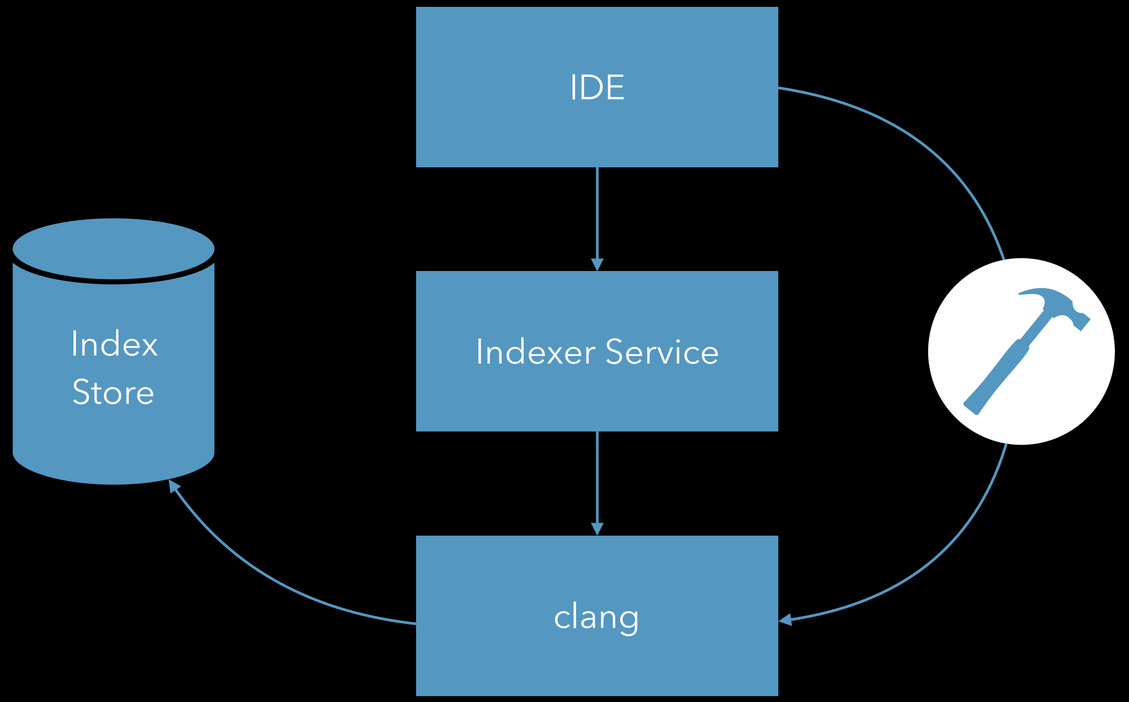
Indexing
while building

- User-initiated so unthrottled.



Indexing
while building

- User-initiated so unthrottled.
- Reuses ASTs from build.



Build overhead: 3–5%

Build overhead: 3–5%

LLVM+CLANG

3.2%

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

- Implemented via a new `FrontendAction`: `WrappingIndexRecordAction`.

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

- Implemented via a new `FrontendAction`: `WrappingIndexRecordAction`.
- Uses `IndexASTConsumer` to collect symbol information from the AST.

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

- Implemented via a new `FrontendAction`: `WrappingIndexRecordAction`.
- Uses `IndexASTConsumer` to collect symbol information from the AST.
- Uses `IndexDependencyProvider` to track source and module dependencies.

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

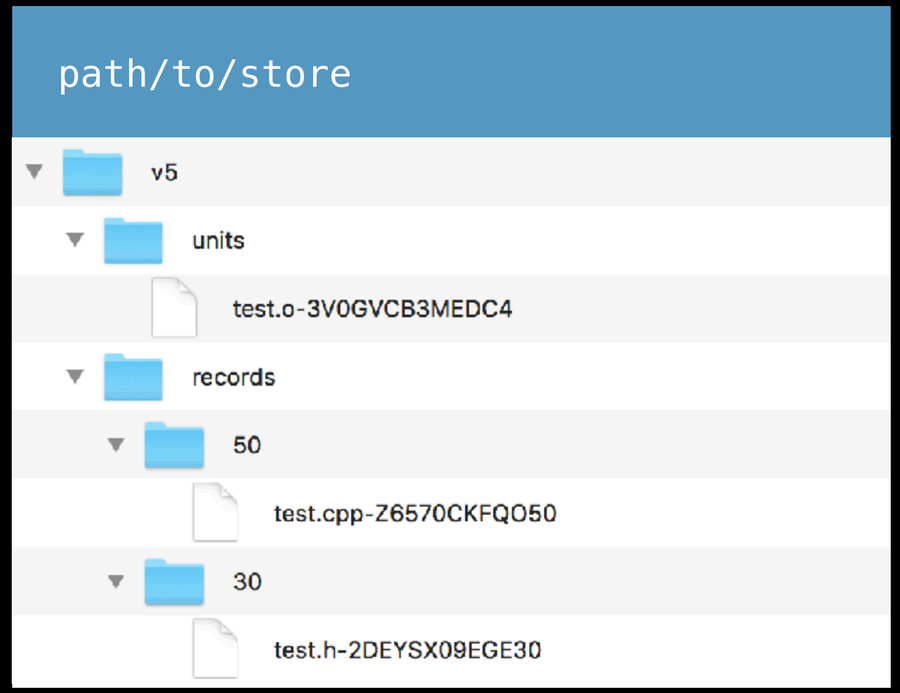
- Implemented via a new `FrontendAction`: `WrappingIndexRecordAction`.
- Uses `IndexASTConsumer` to collect symbol information from the AST.
- Uses `IndexDependencyProvider` to track source and module dependencies.
- Writes this out to the provided store path once complete.

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```

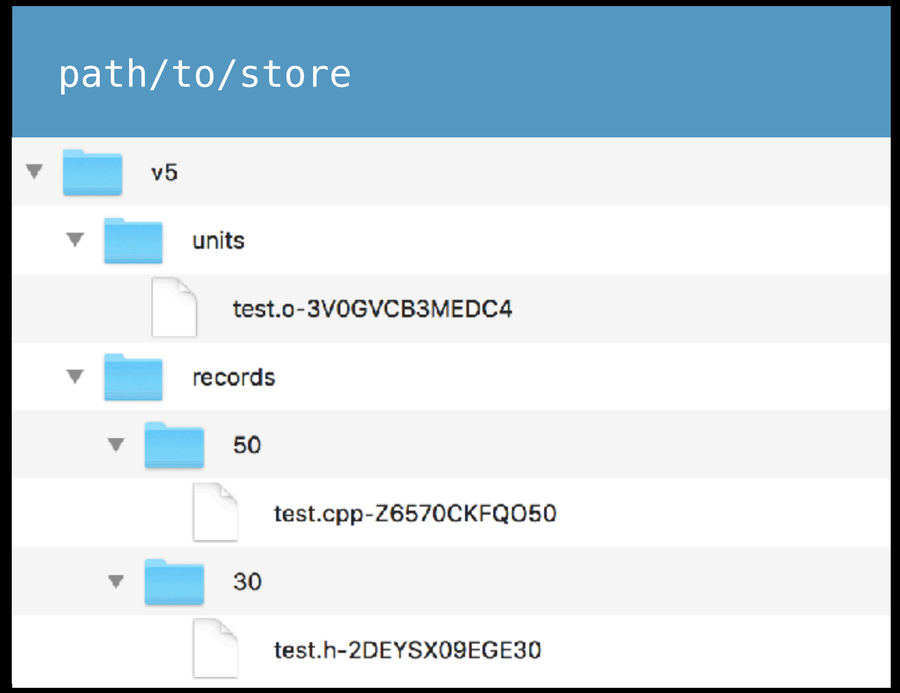
Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```



Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```



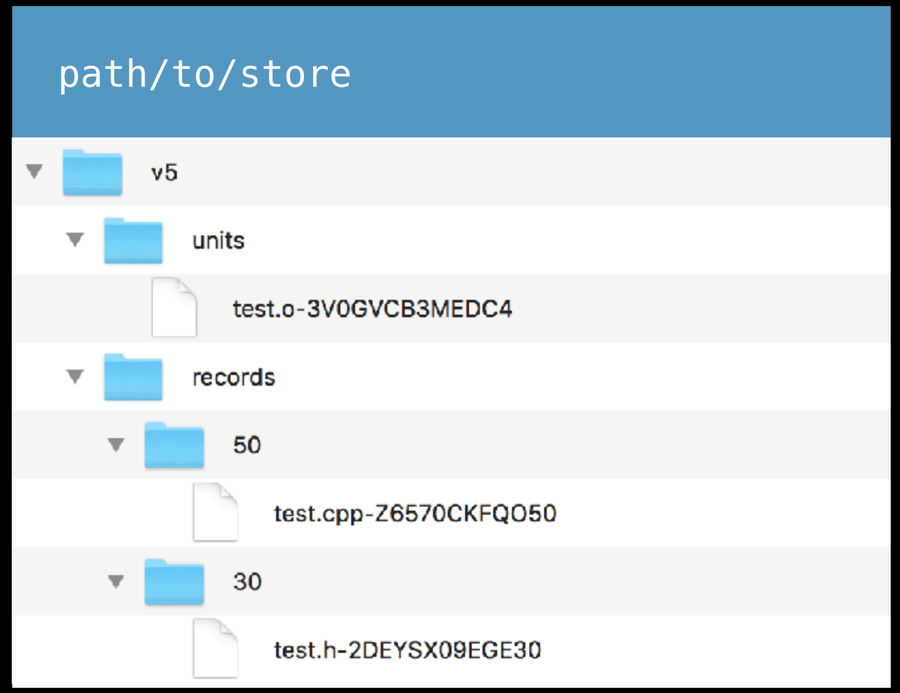
Unit

Provider
IsSystem
IsModule
HasMain
MainPath
WorkingDir
OutputFile
Target

Dependencies

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```



Unit

Provider
IsSystem
IsModule
HasMain
MainPath
WorkingDir
OutputFile
Target

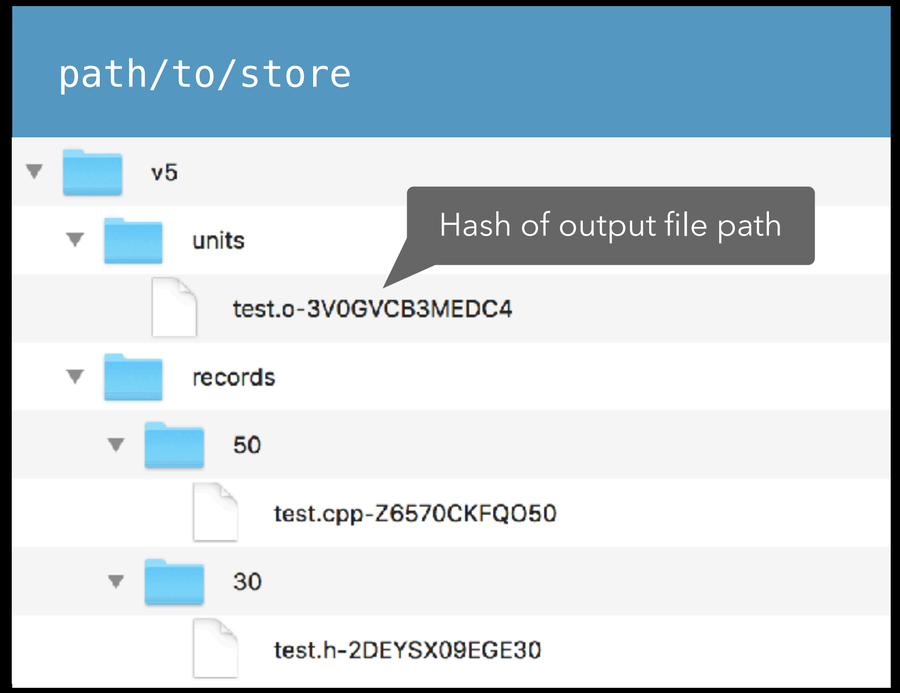
Dependencies

Record

Symbols
Occurrences

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```



Unit

Provider
IsSystem
IsModule
HasMain
MainPath
WorkingDir
OutputFile
Target

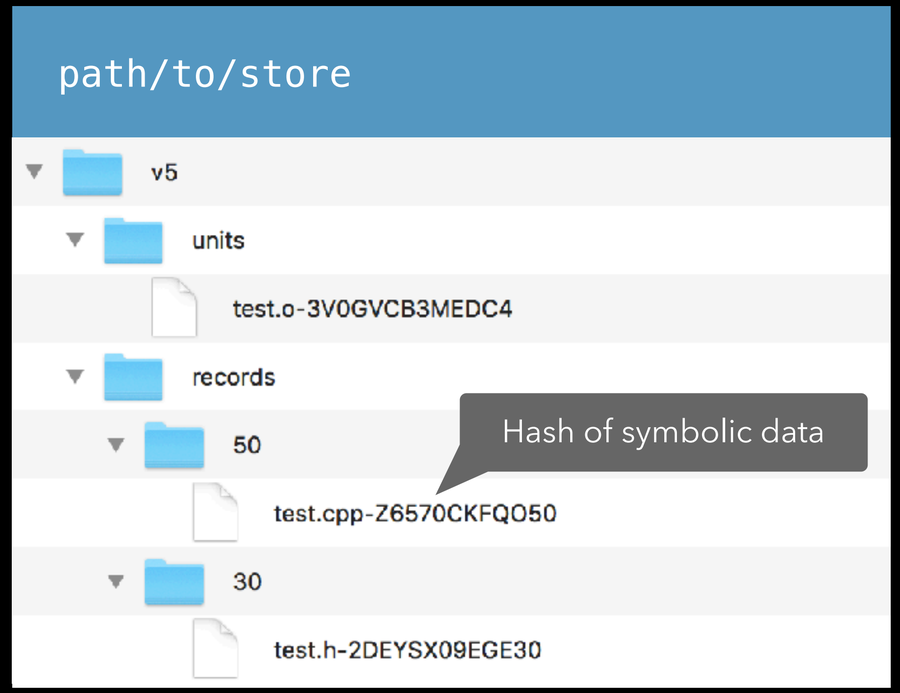
Dependencies

Record

Symbols
Occurrences

Generating the index data

```
$ clang -o test.o -c test.cpp -index-store-path path/to/store
```



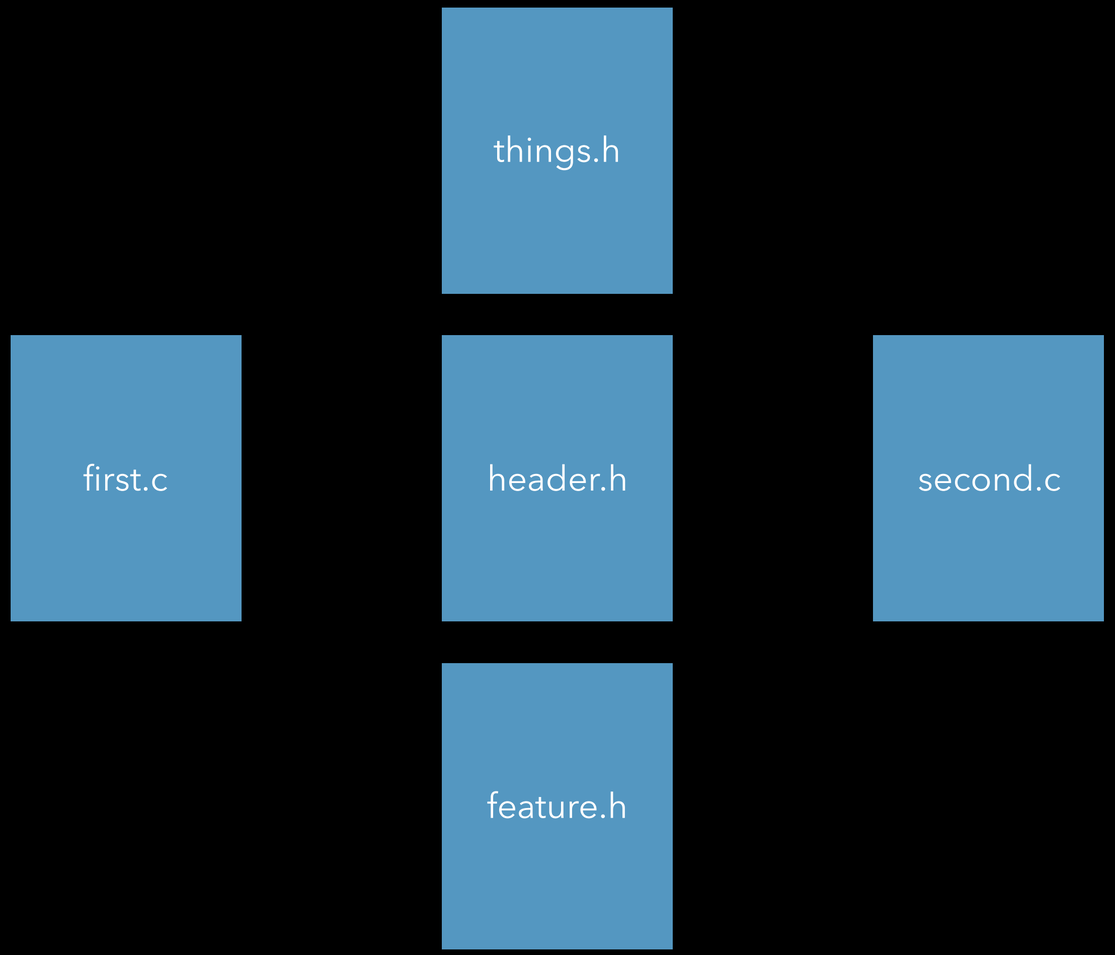
Unit

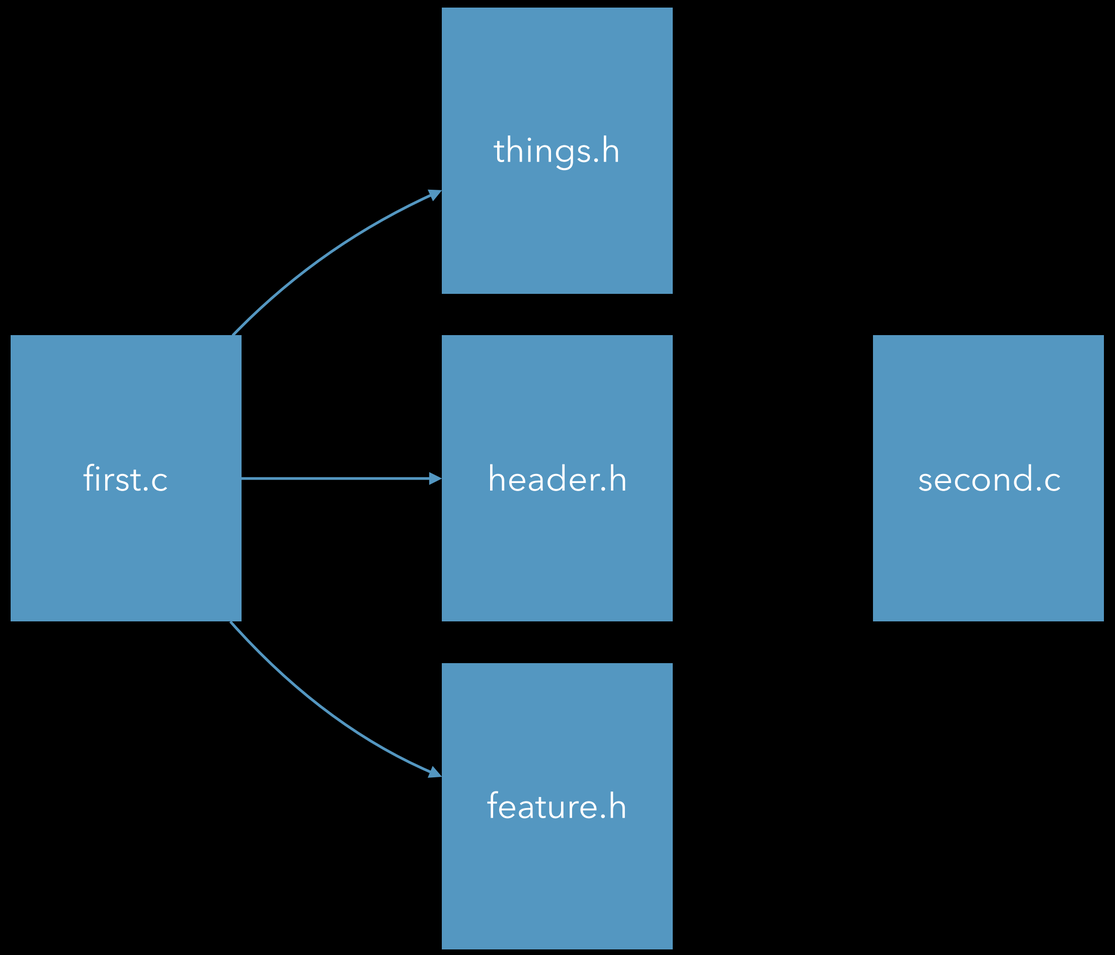
Provider
IsSystem
IsModule
HasMain
MainPath
WorkingDir
OutputFile
Target

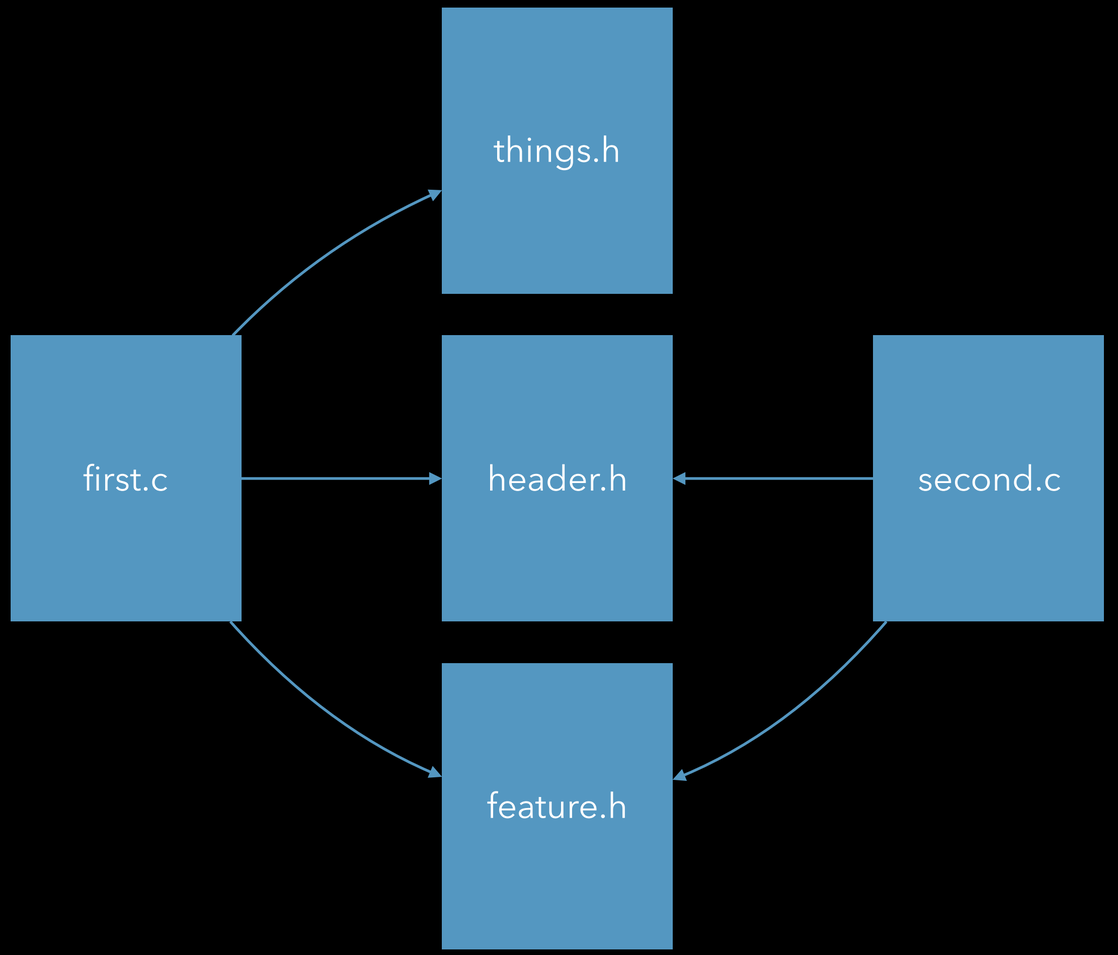
Dependencies

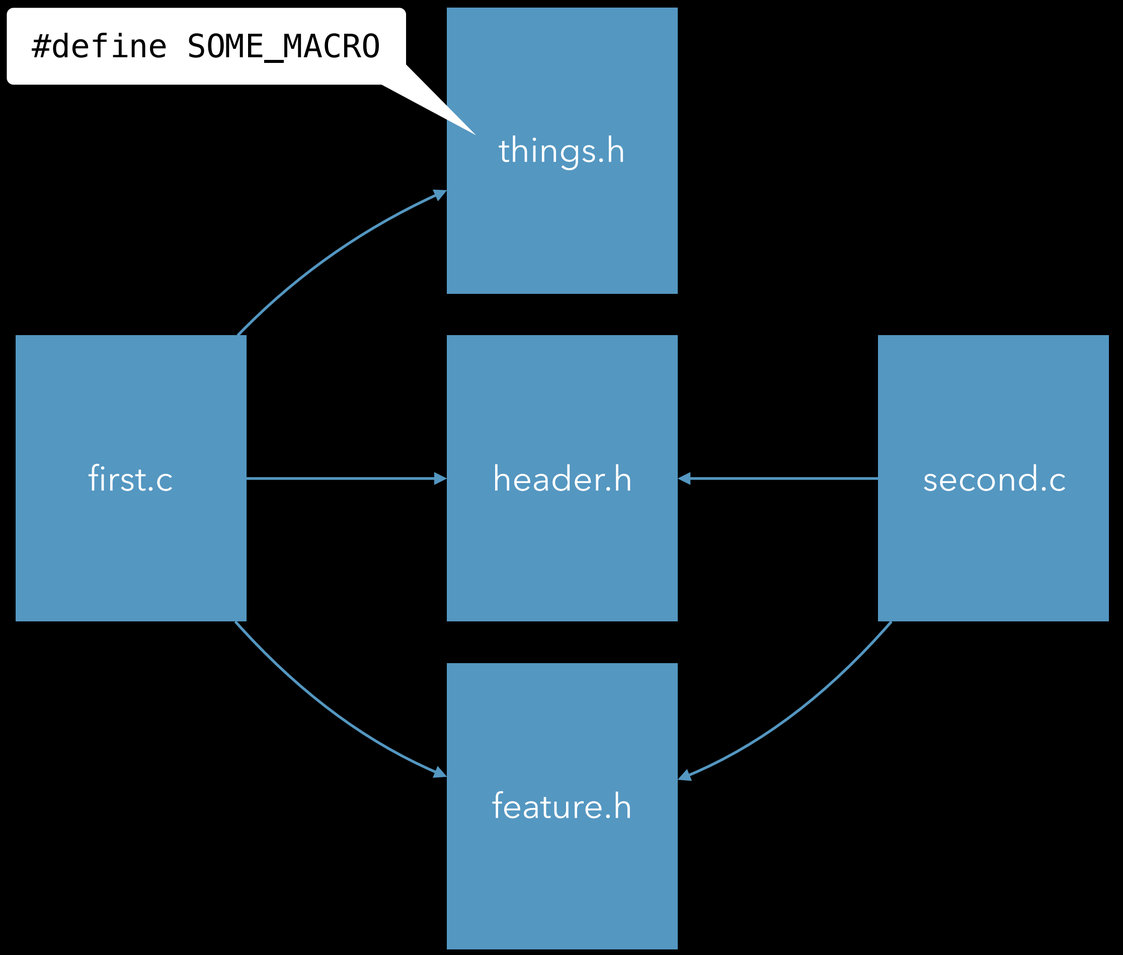
Record

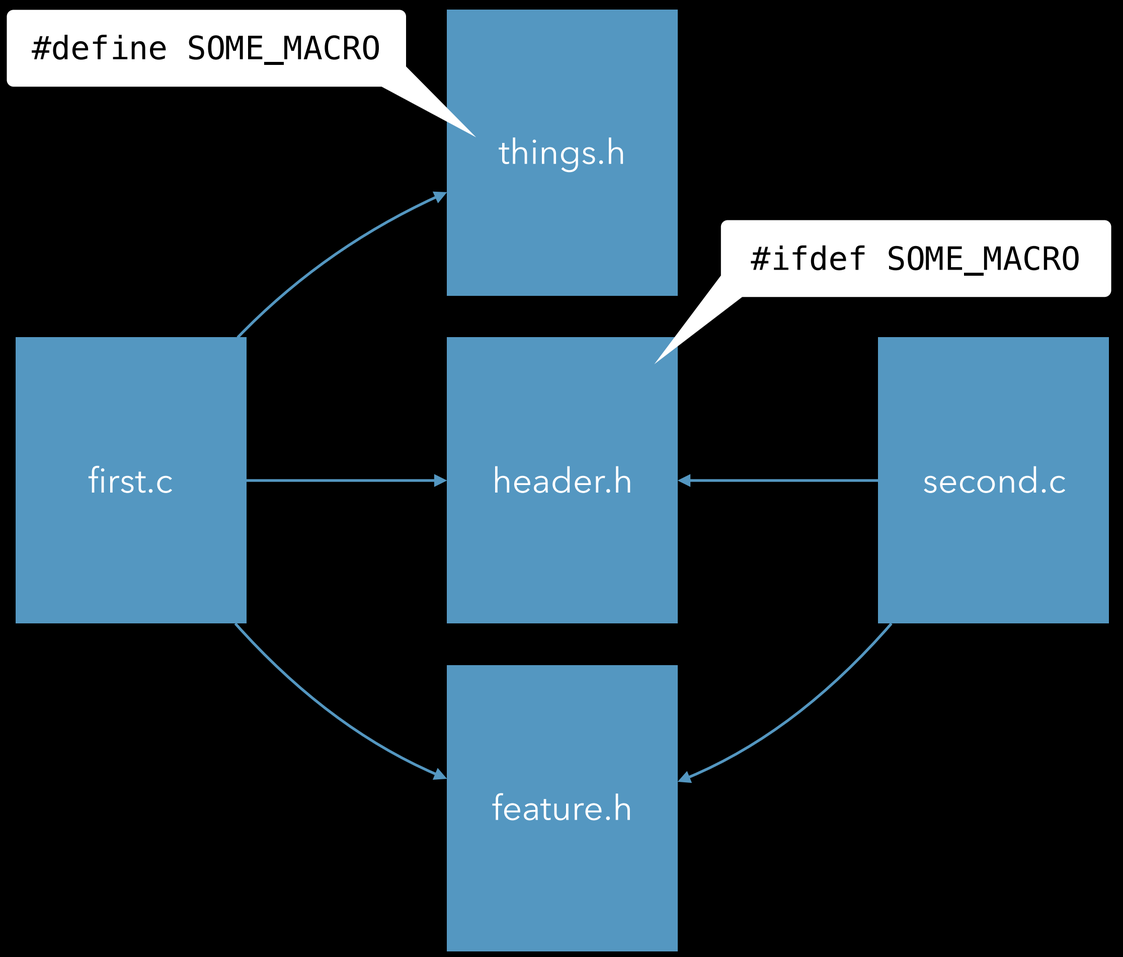
Symbols
Occurrences

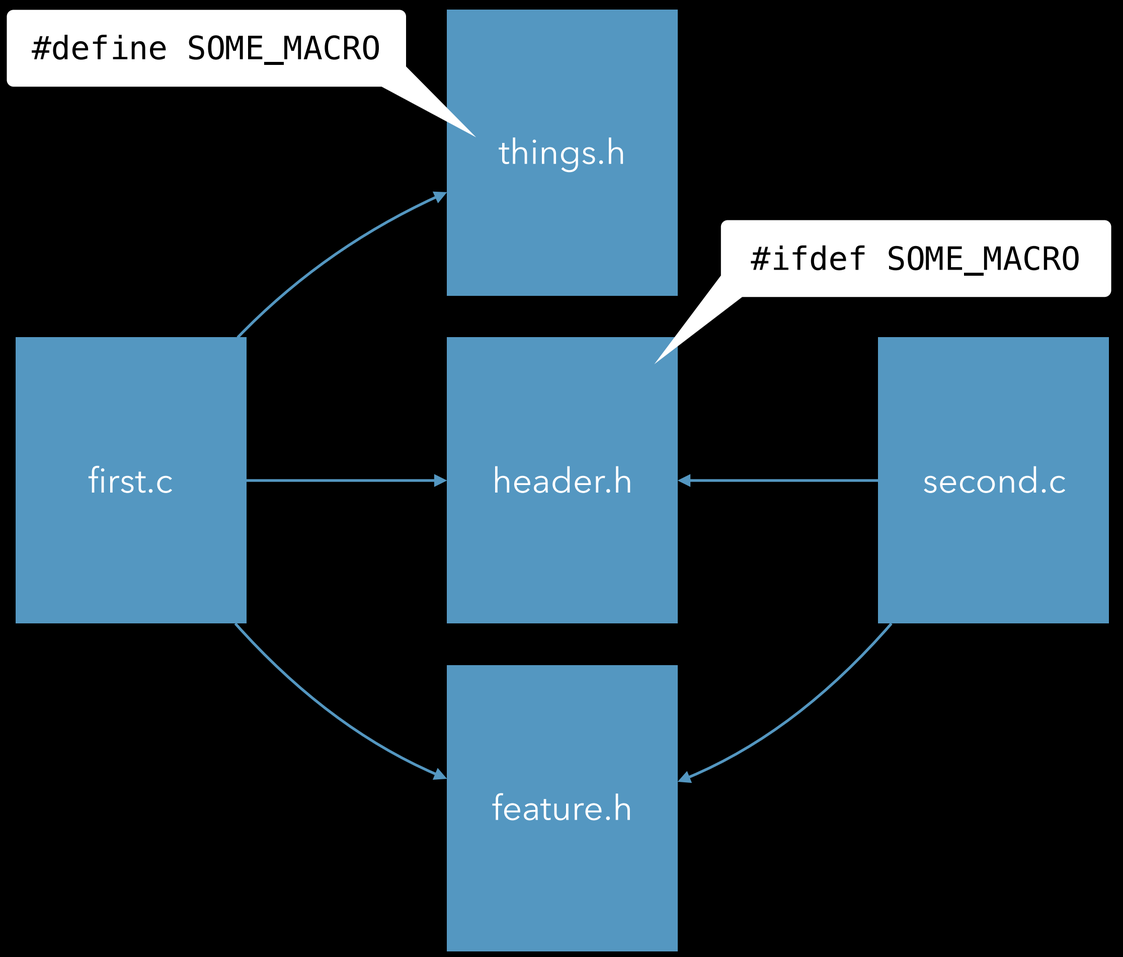

things.h
first.c
header.h
second.c
feature.h


things.h
first.c
header.h
second.c
feature.h

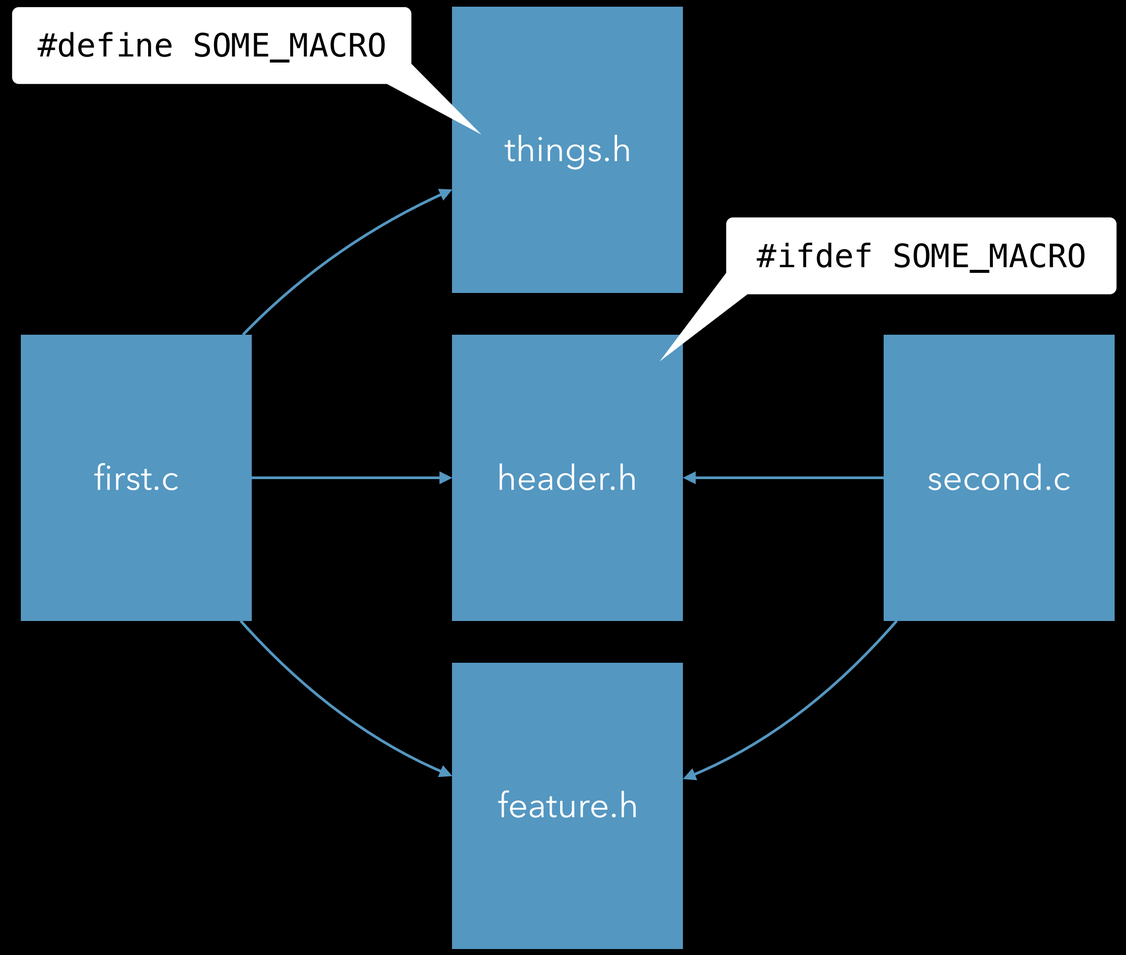

things.h
first.c
header.h
second.c
feature.h


#define SOME_MACRO
things.h
first.c
header.h
second.c
feature.h

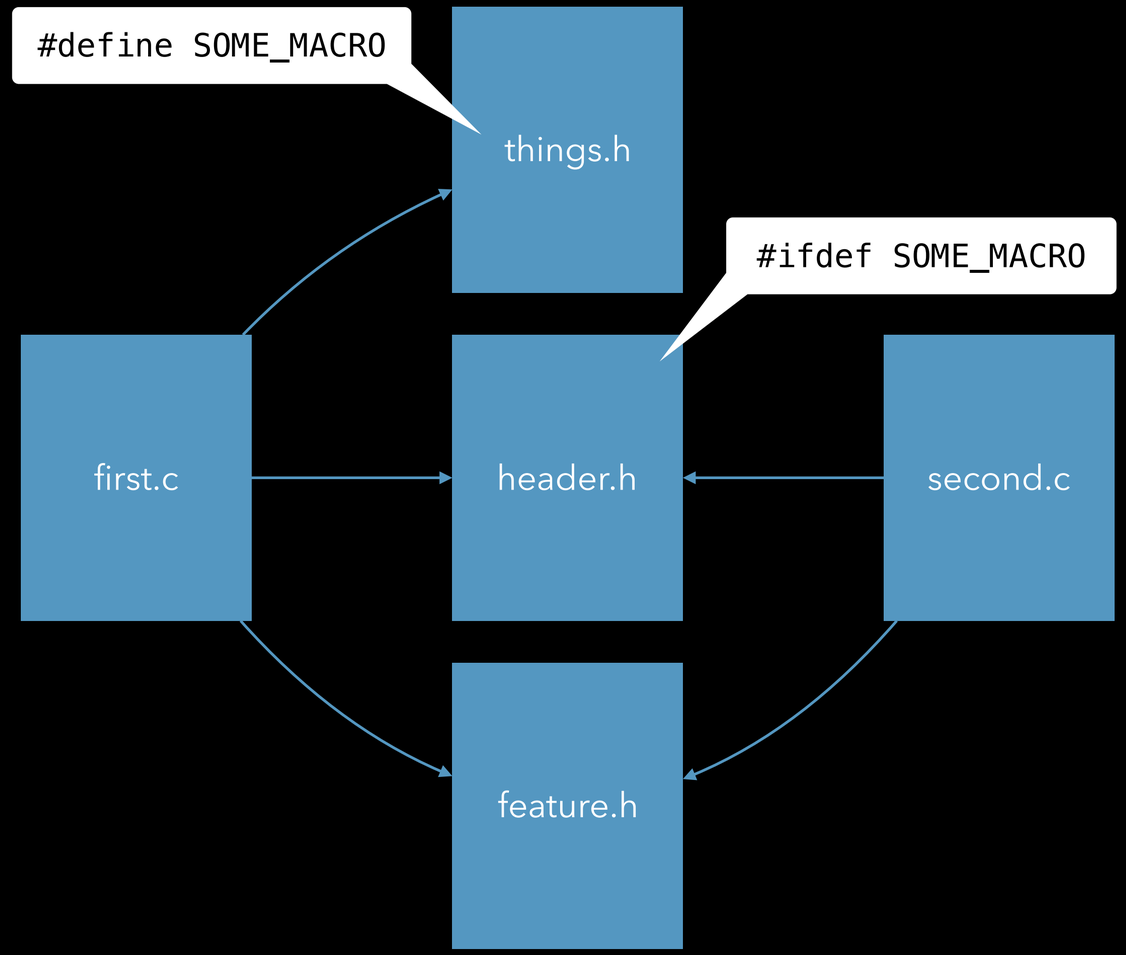

#define SOME_MACRO
things.h
#ifdef SOME_MACRO
first.c
header.h
second.c
feature.h


#define SOME_MACRO
things.h
#ifdef SOME_MACRO
first.c
header.h
second.c
feature.h

```
$ clang -o first.o -c first.c -index-store-path path/to/store
```

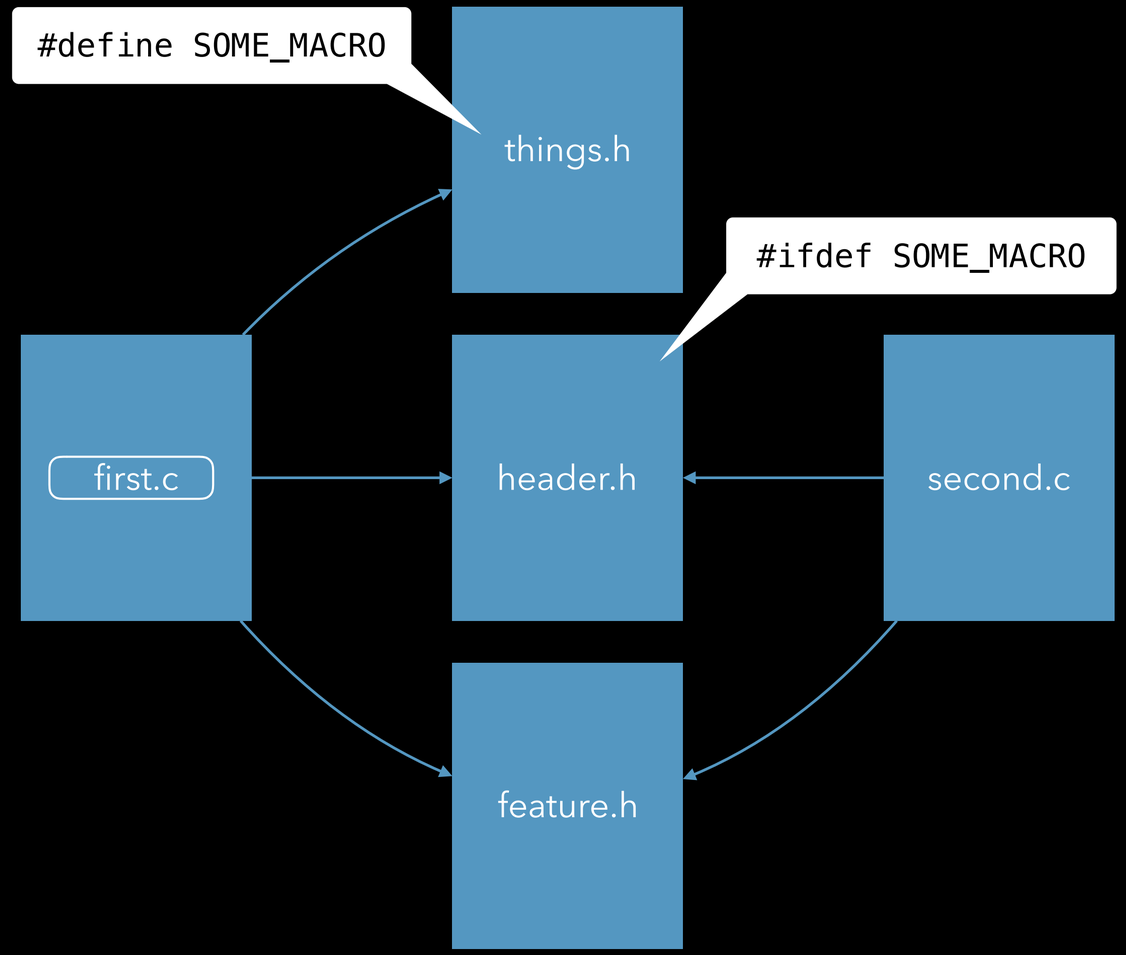


```
$ clang -o first.o -c first.c -index-store-path path/to/store
```



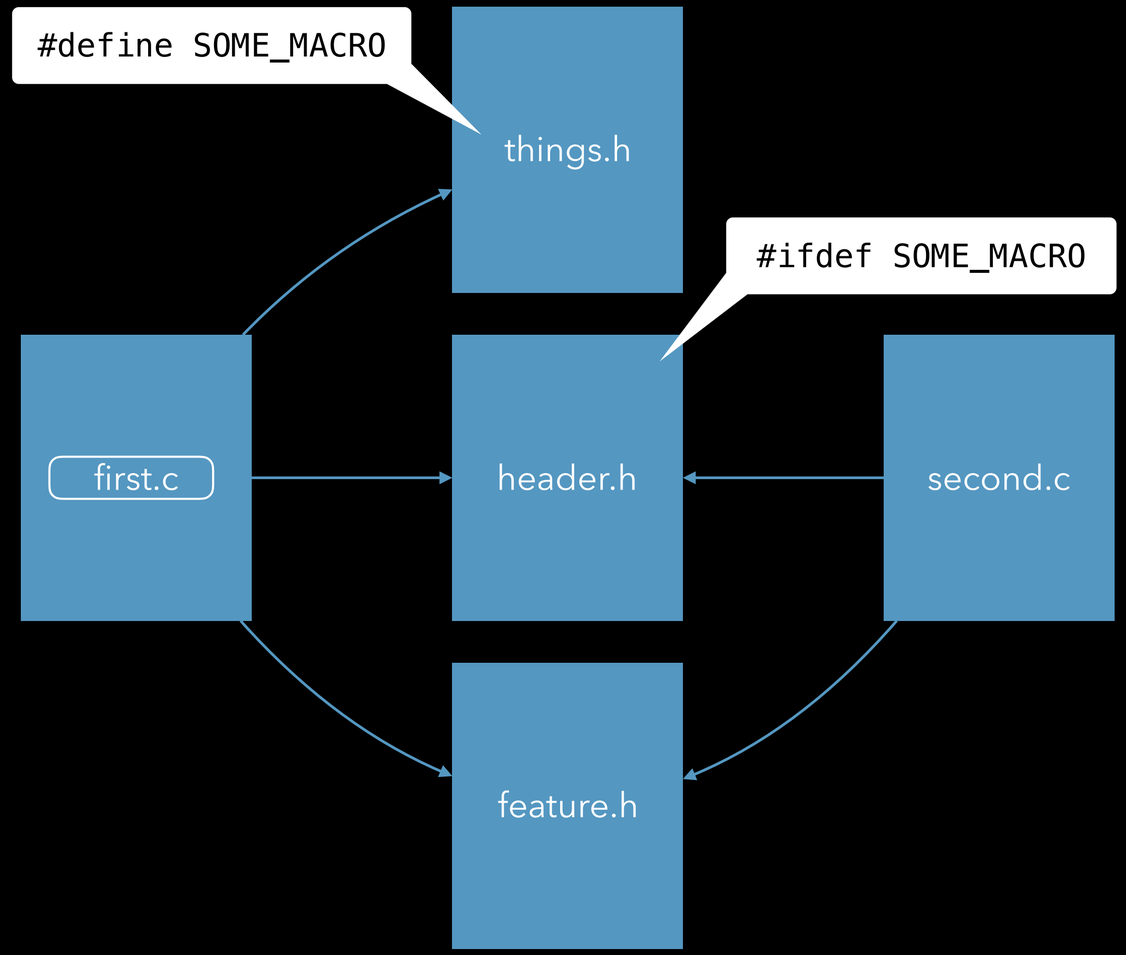
Unit
first.o-2J808UIU…

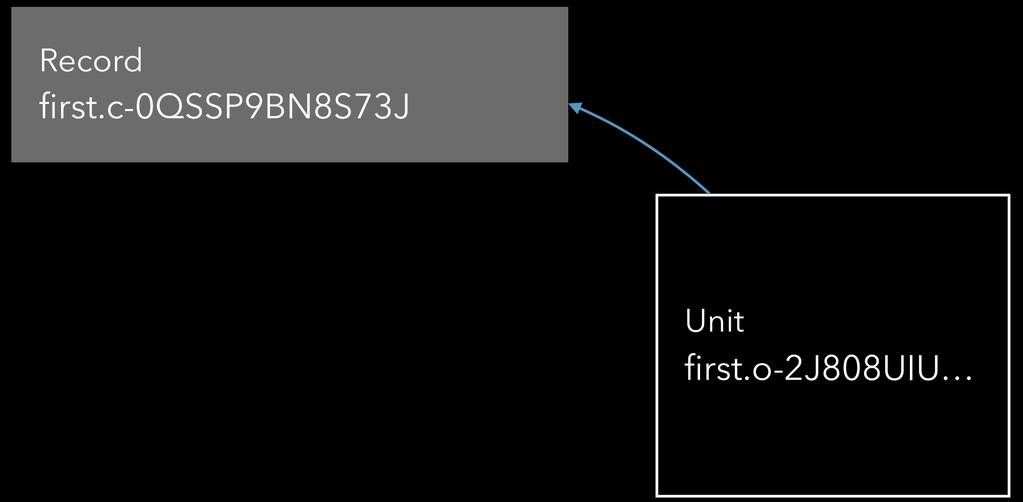
```
$ clang -o first.o -c first.c -index-store-path path/to/store
```



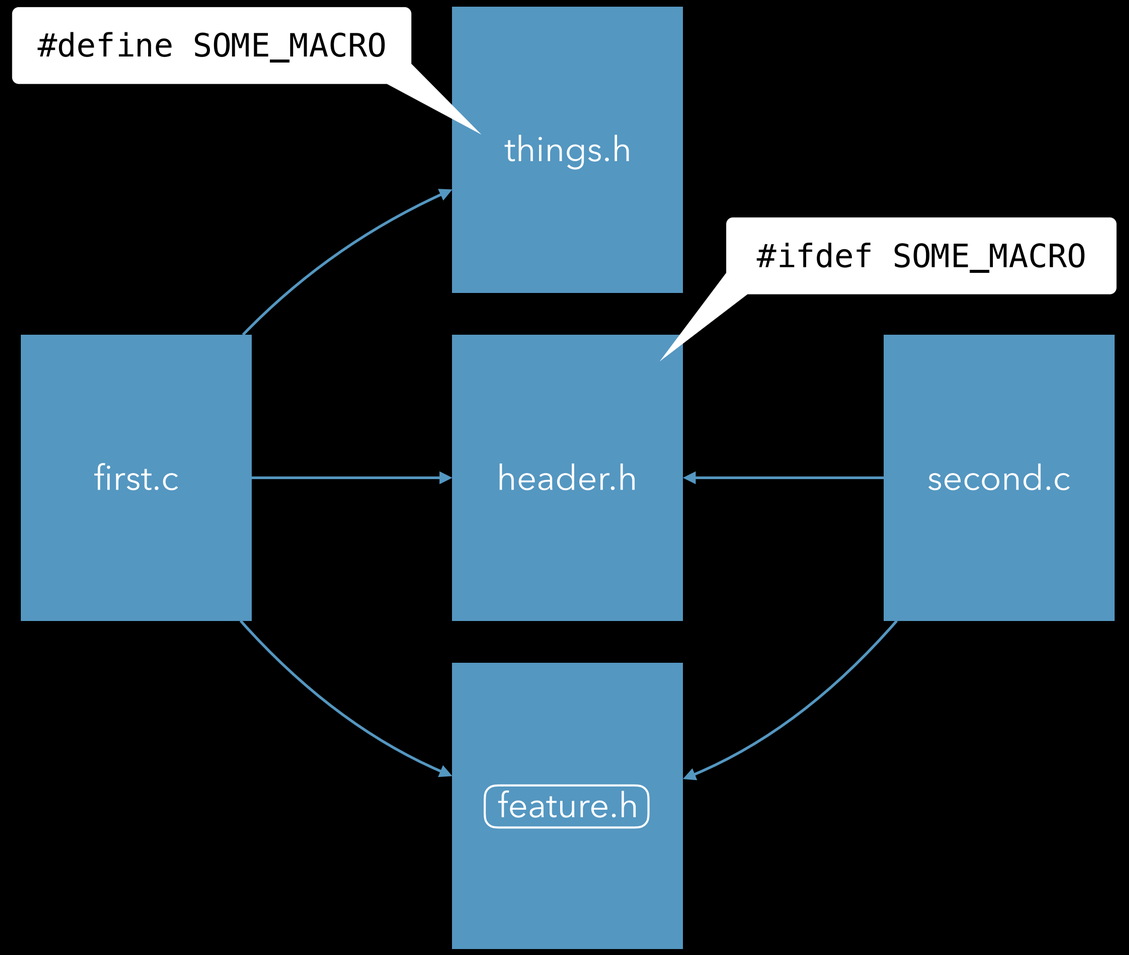
Unit
first.o-2J808UIU…

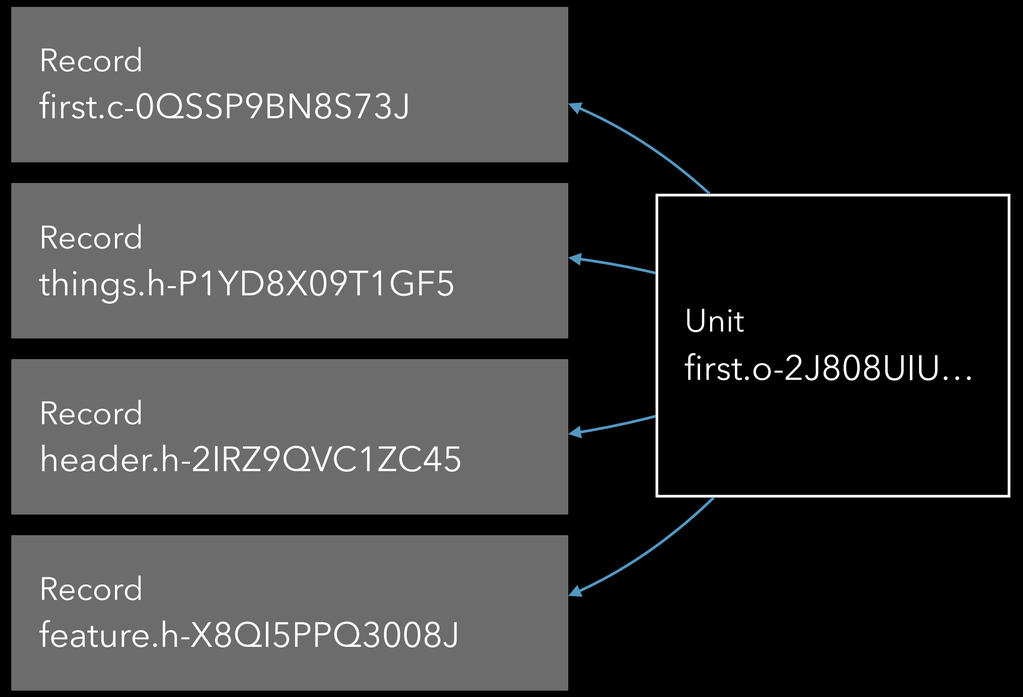
```
$ clang -o first.o -c first.c -index-store-path path/to/store
```

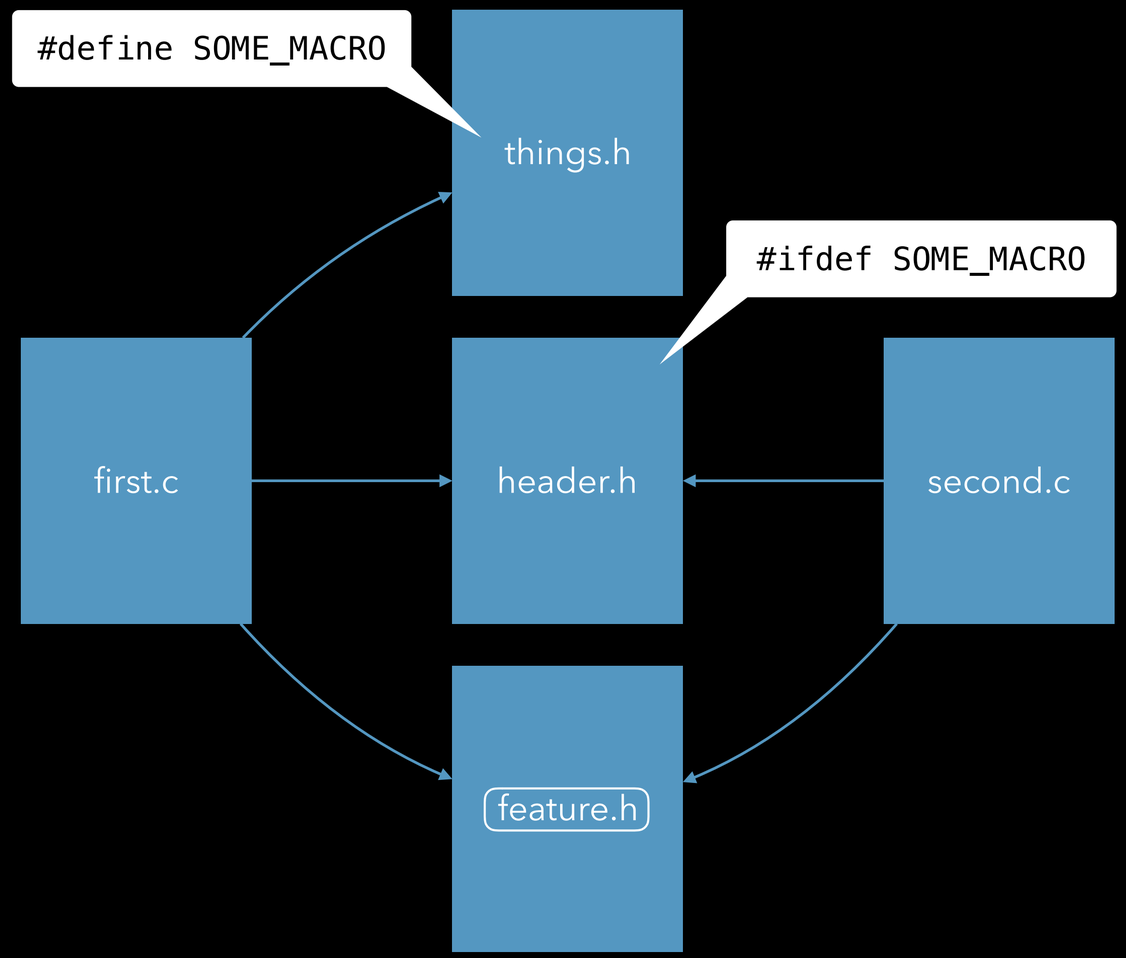


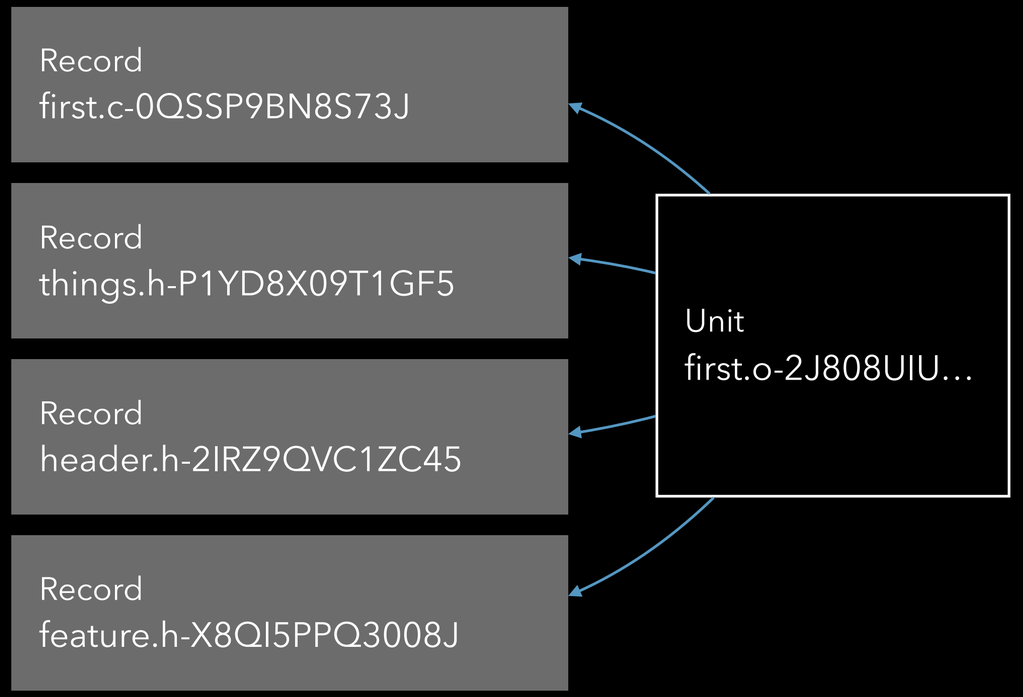


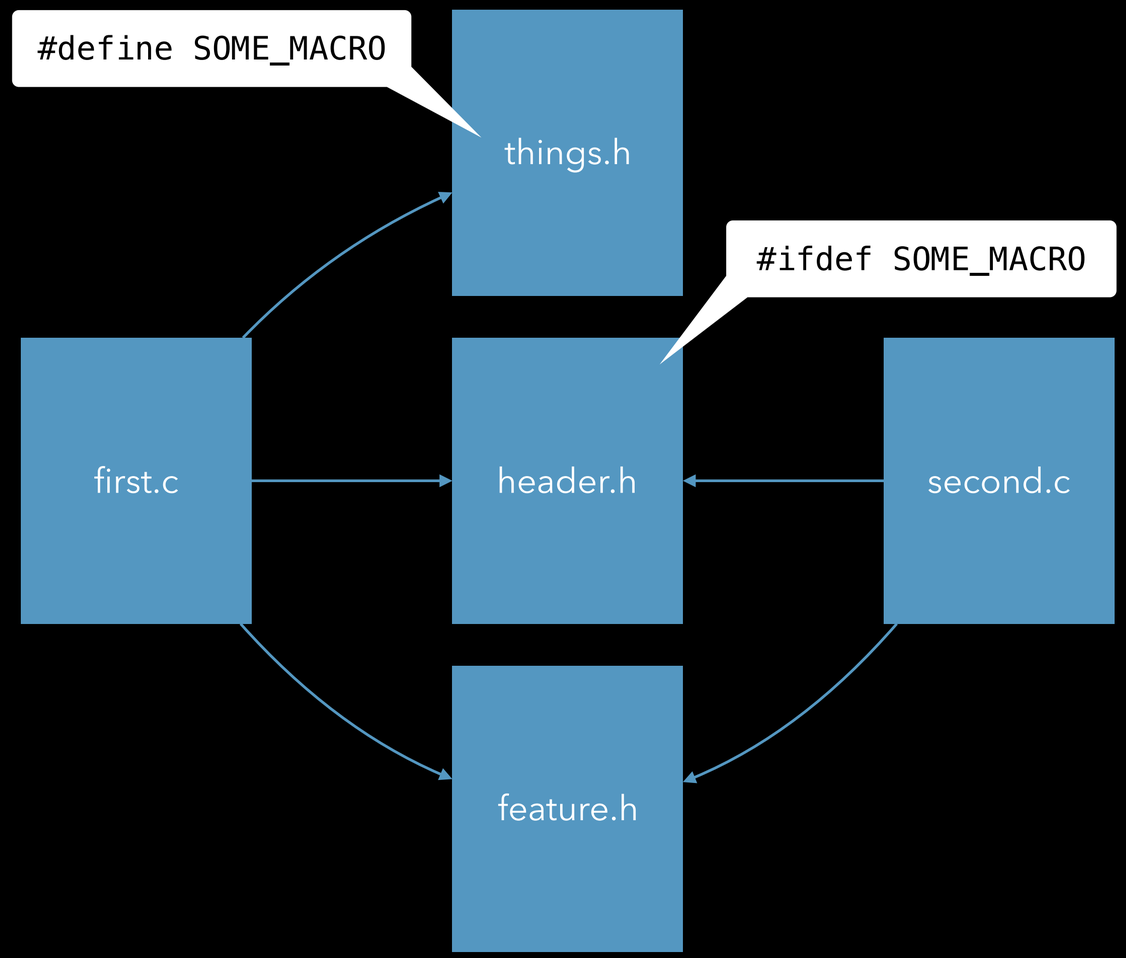
```
$ clang -o first.o -c first.c -index-store-path path/to/store
```

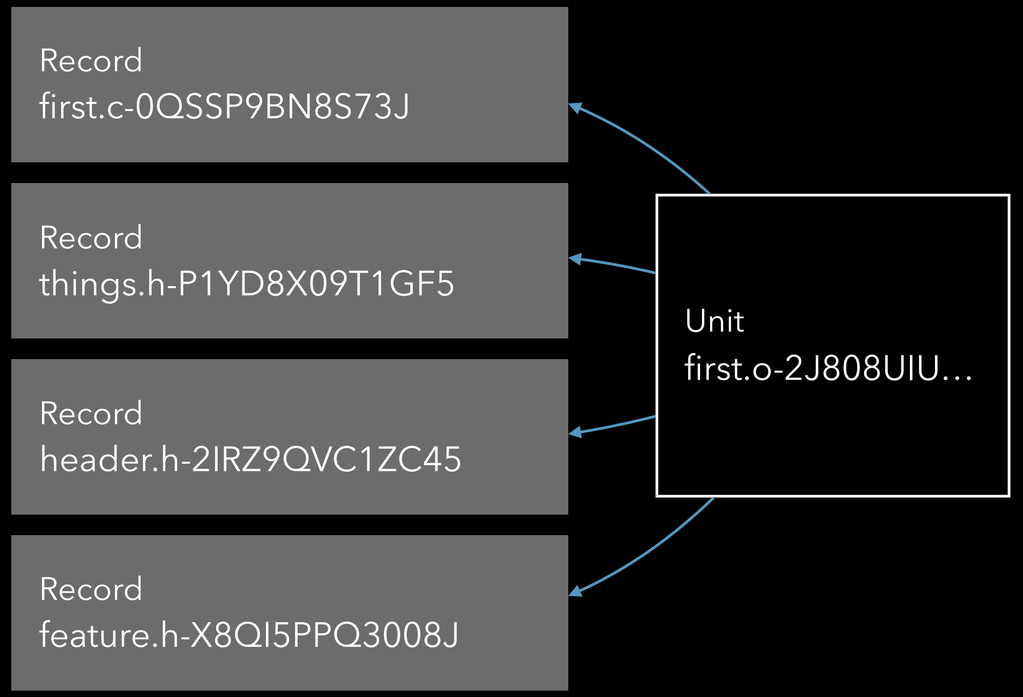




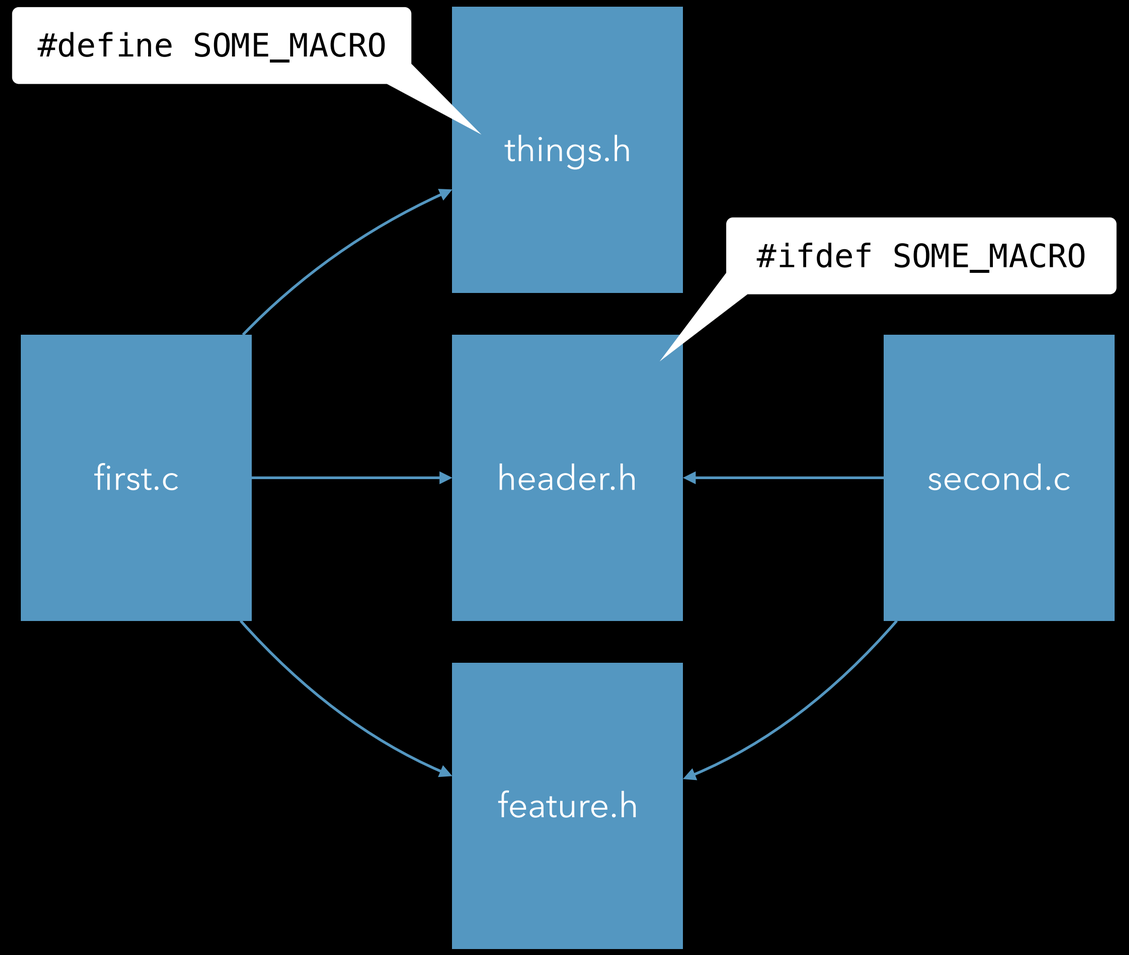

#define SOME_MACRO
things.h
#ifdef SOME_MACRO
first.c
header.h
second.c
feature.h

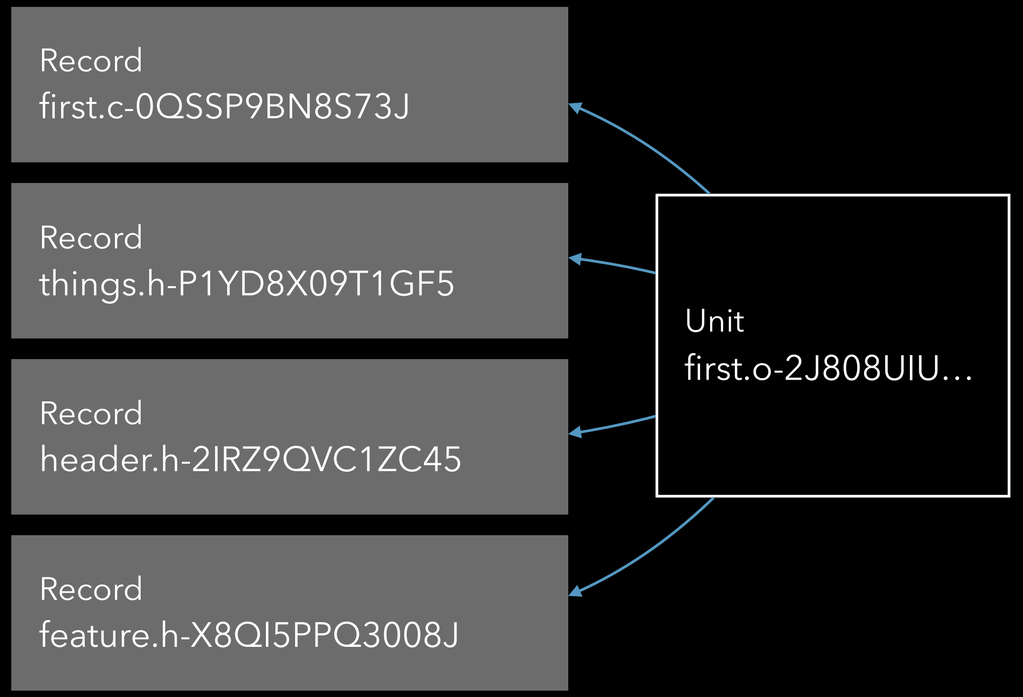

Record
first.c-0QSSP9BN8S73J
Record
things.h-P1YD8X09T1GF5
Record
header.h-2IRZ9QVC1ZC45
Record
feature.h-X8QI5PPQ3008J
Unit
first.o-2J808UIU…


#define SOME_MACRO
things.h
#ifdef SOME_MACRO
first.c
header.h
second.c
feature.h

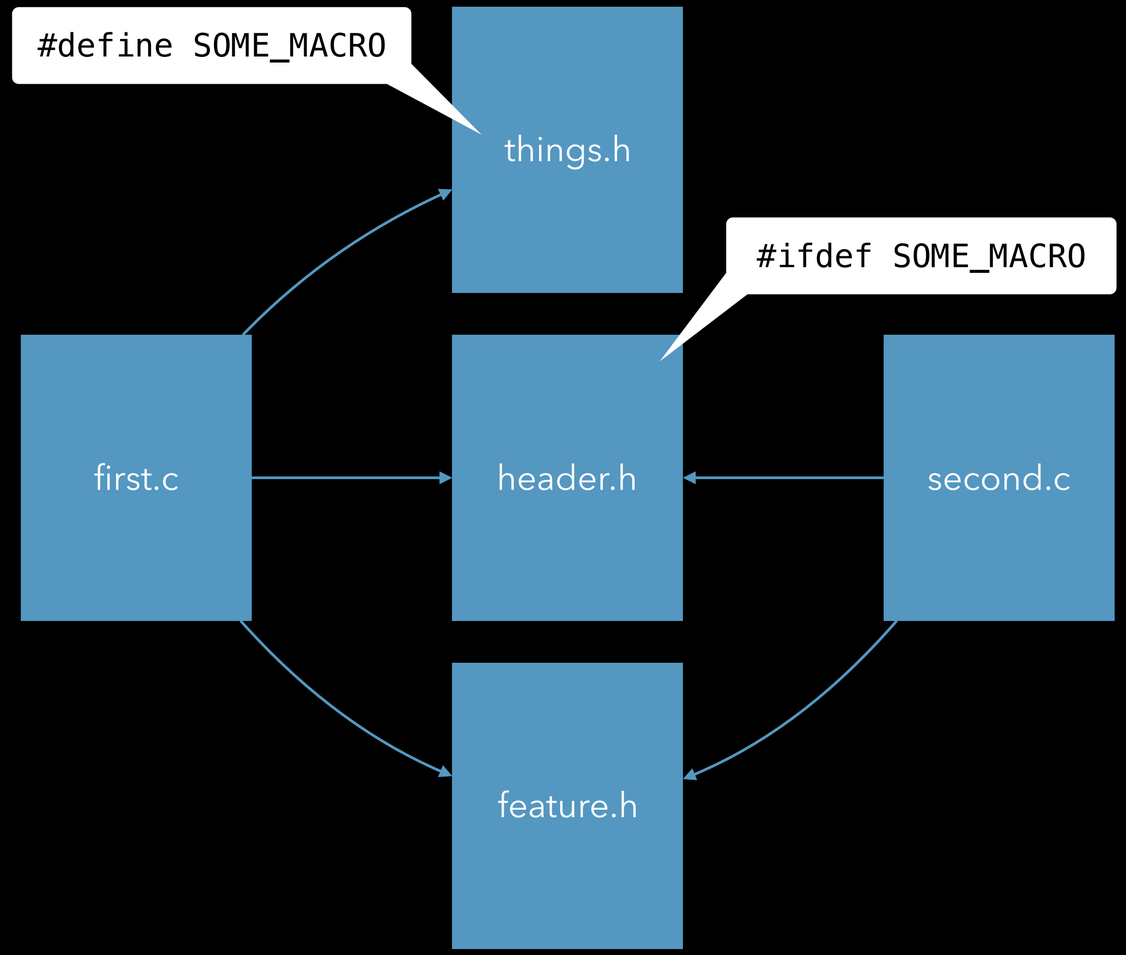

Record
first.c-0QSSP9BN8S73J
Record
things.h-P1YD8X09T1GF5
Record
header.h-2IRZ9QVC1ZC45
Record
feature.h-X8QI5PPQ3008J
Unit
first.o-2J808UIU…

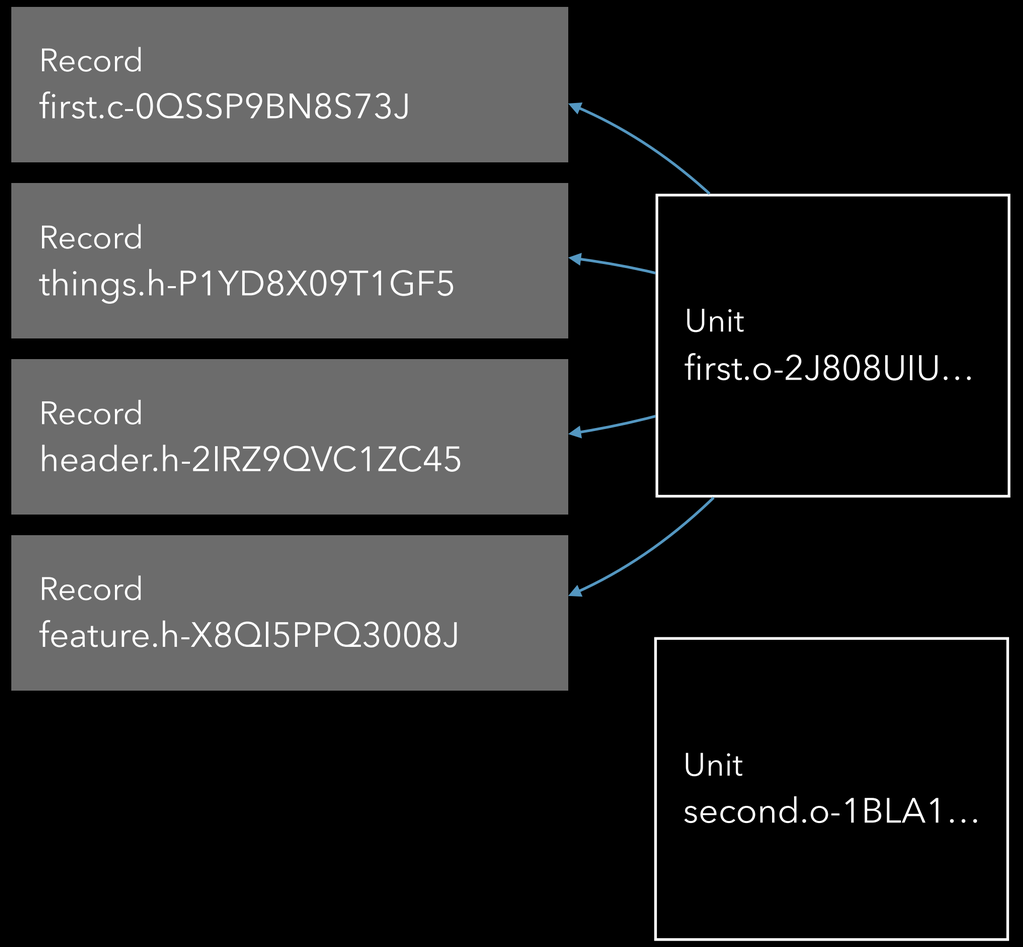
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



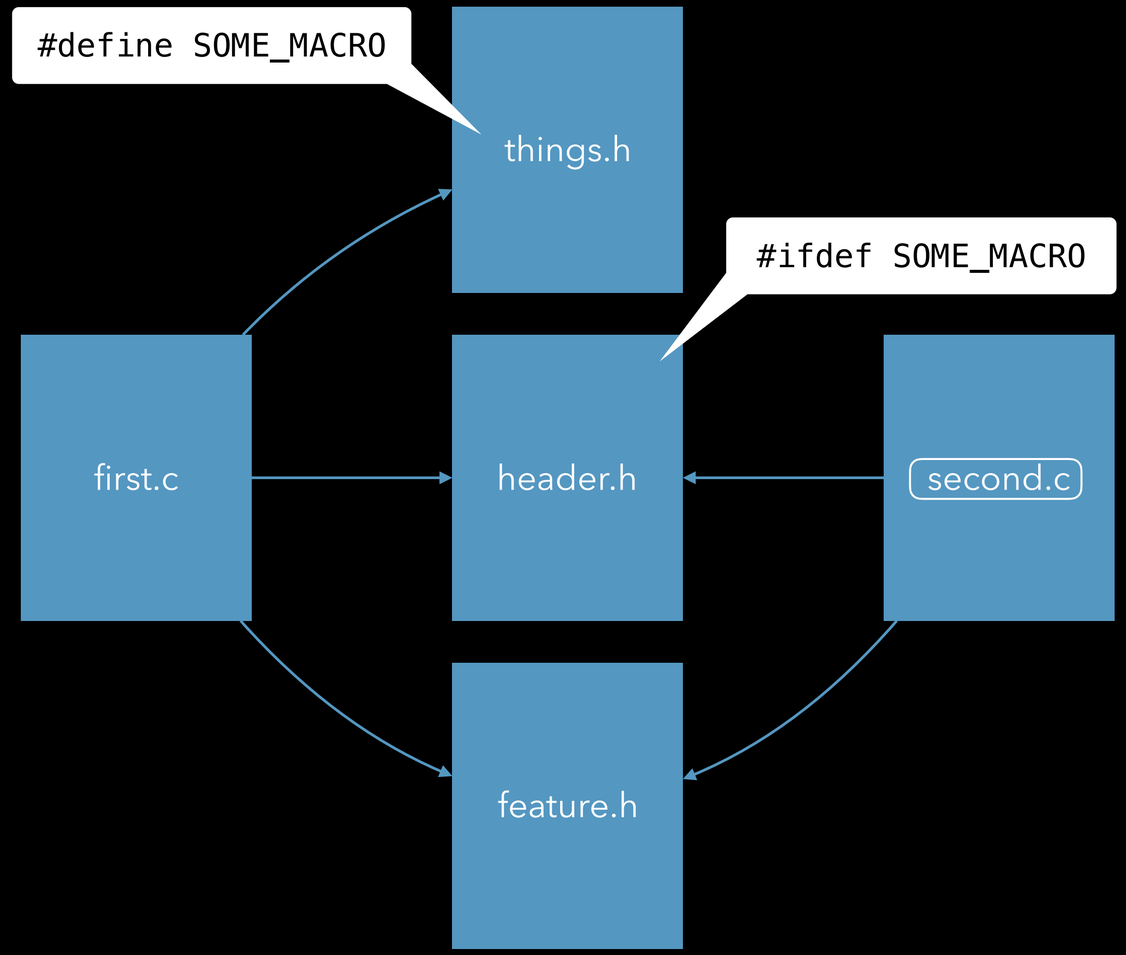


```
$ clang -o second.o -c second.c -index-store-path path/to/store
```

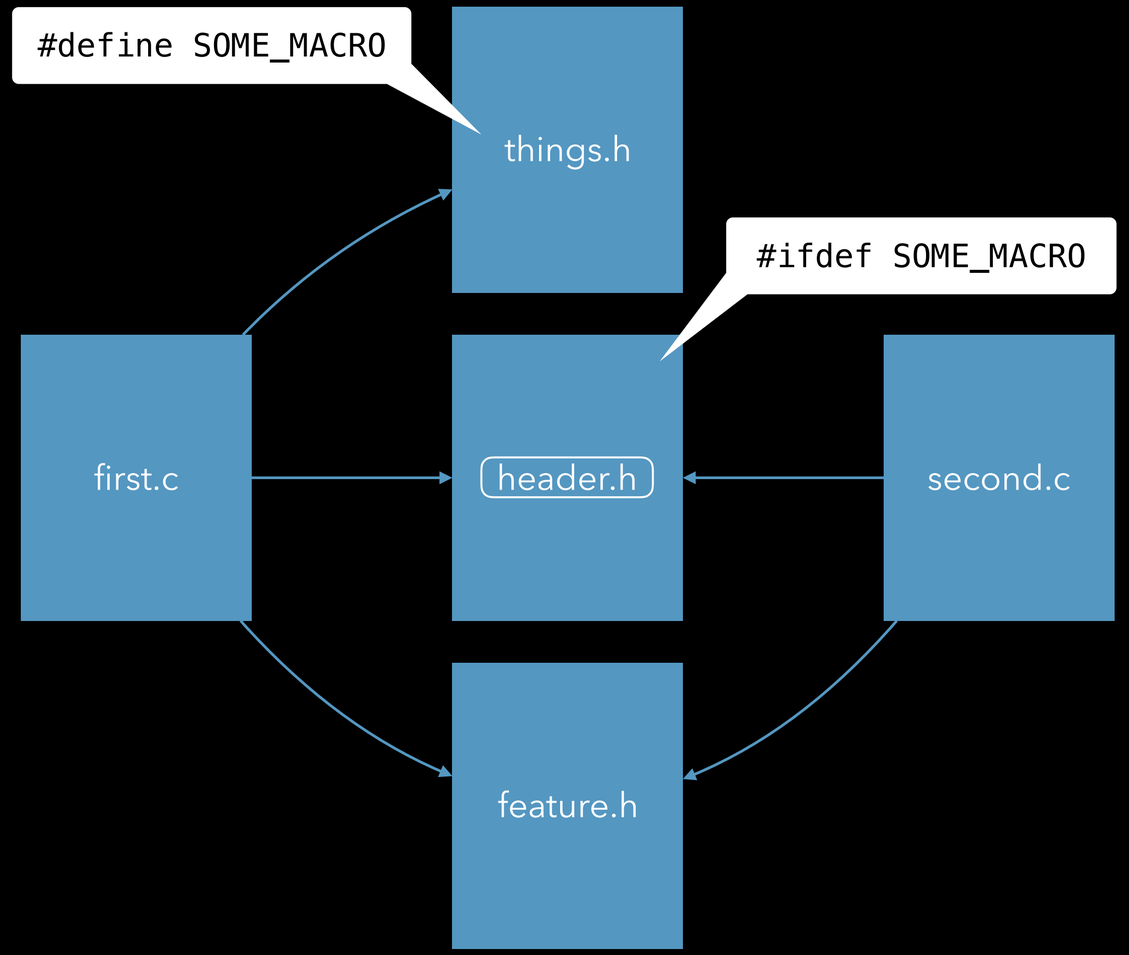




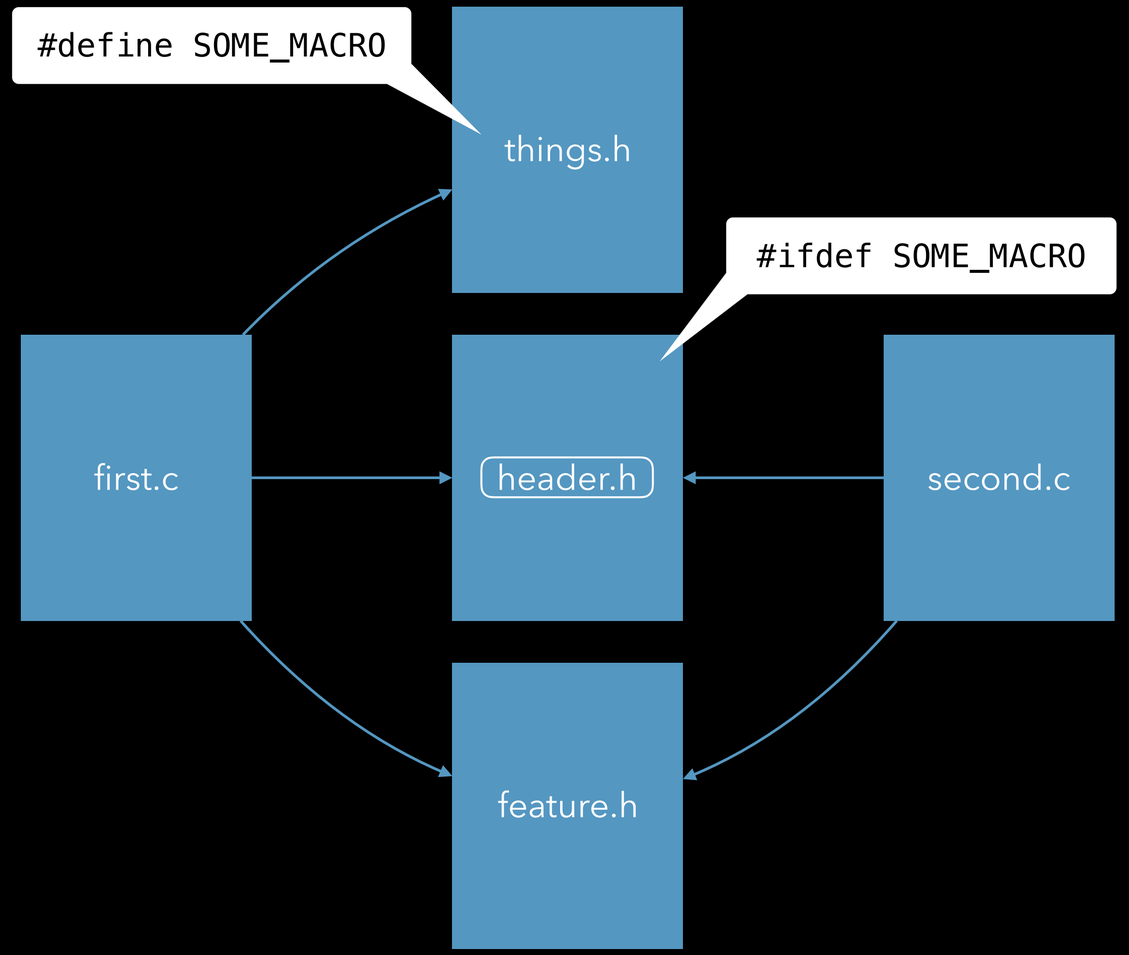
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



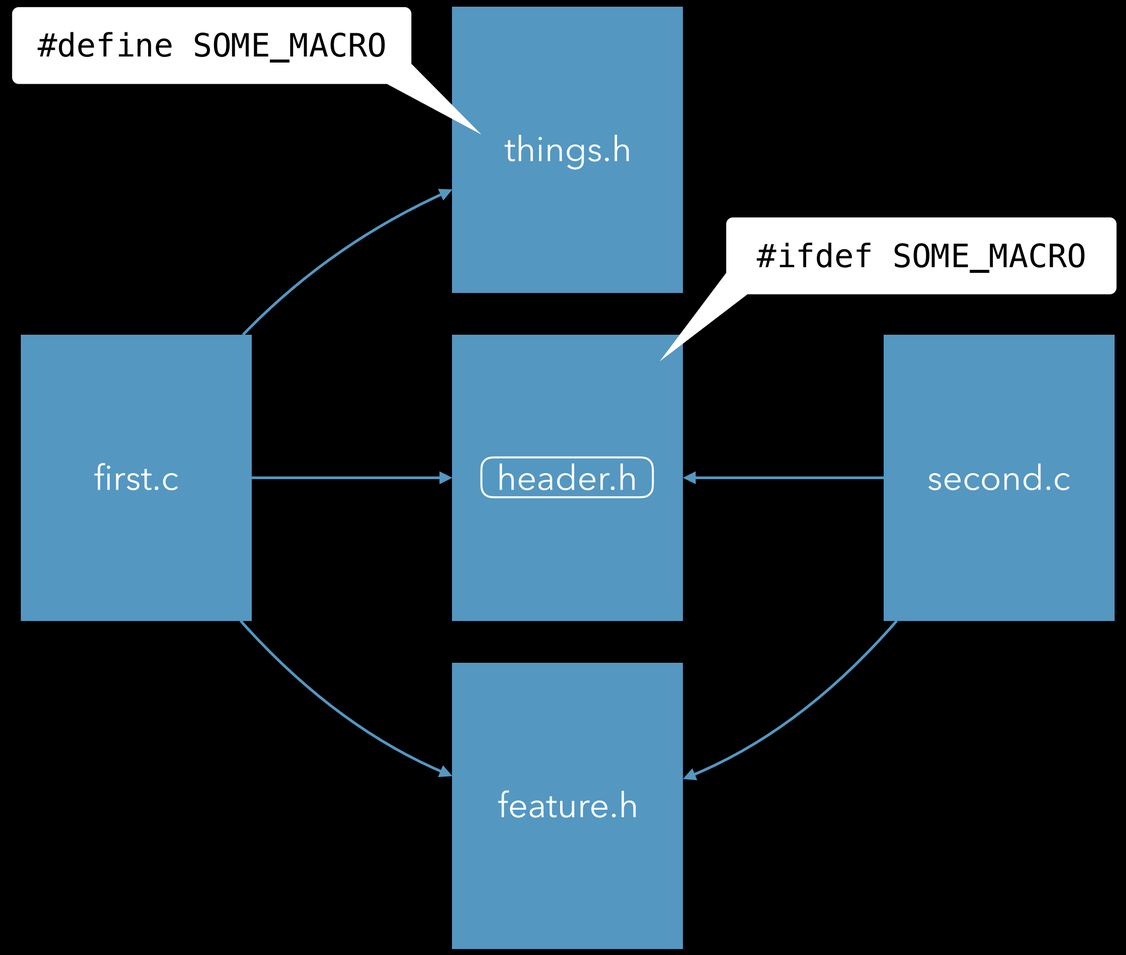
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



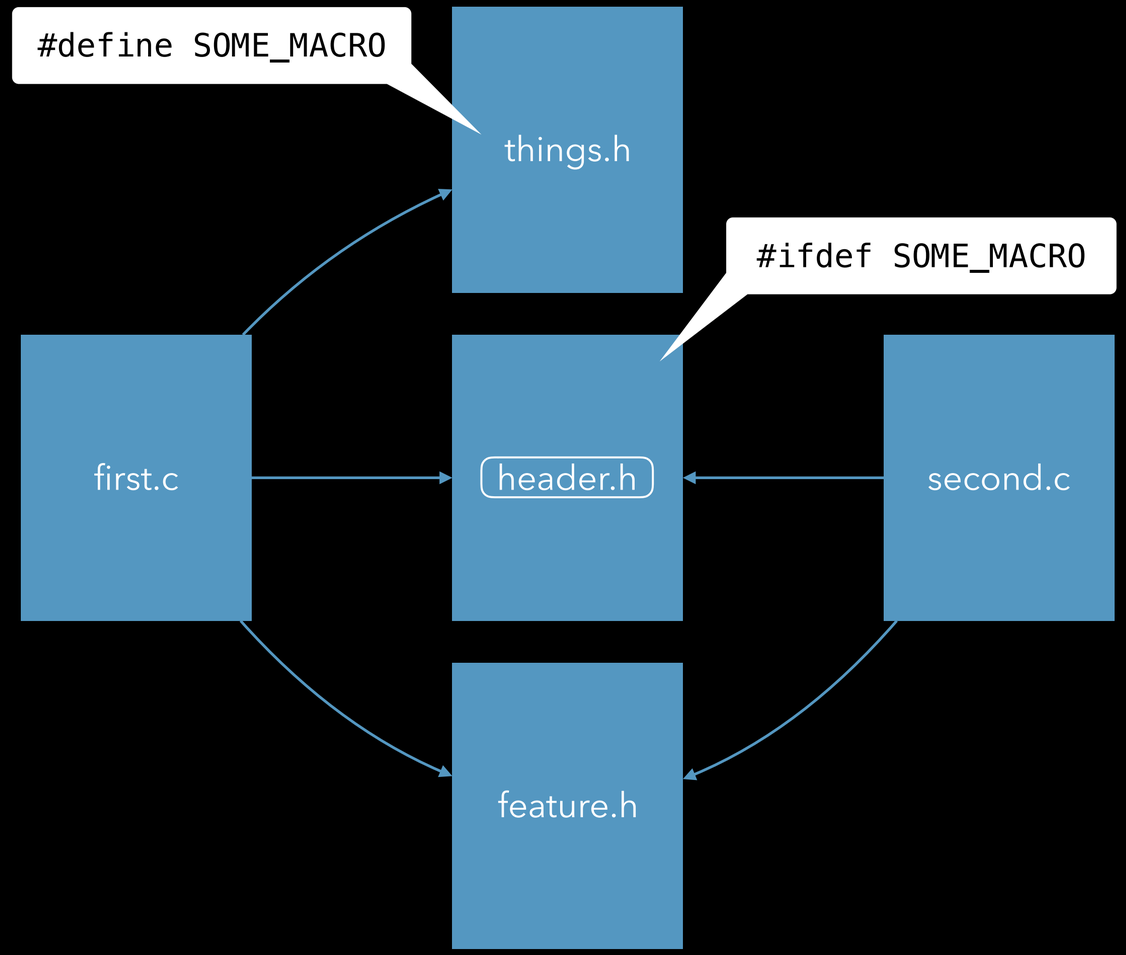
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



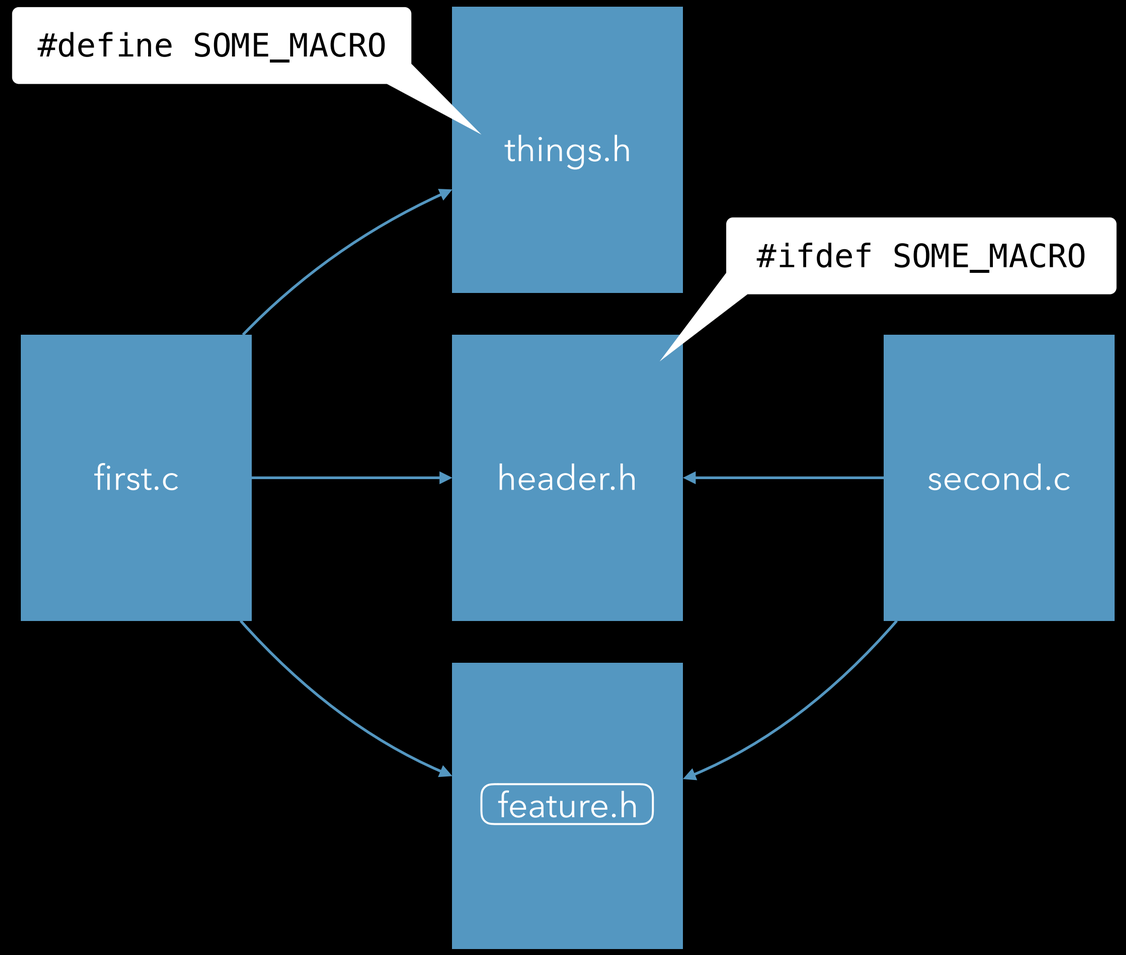
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



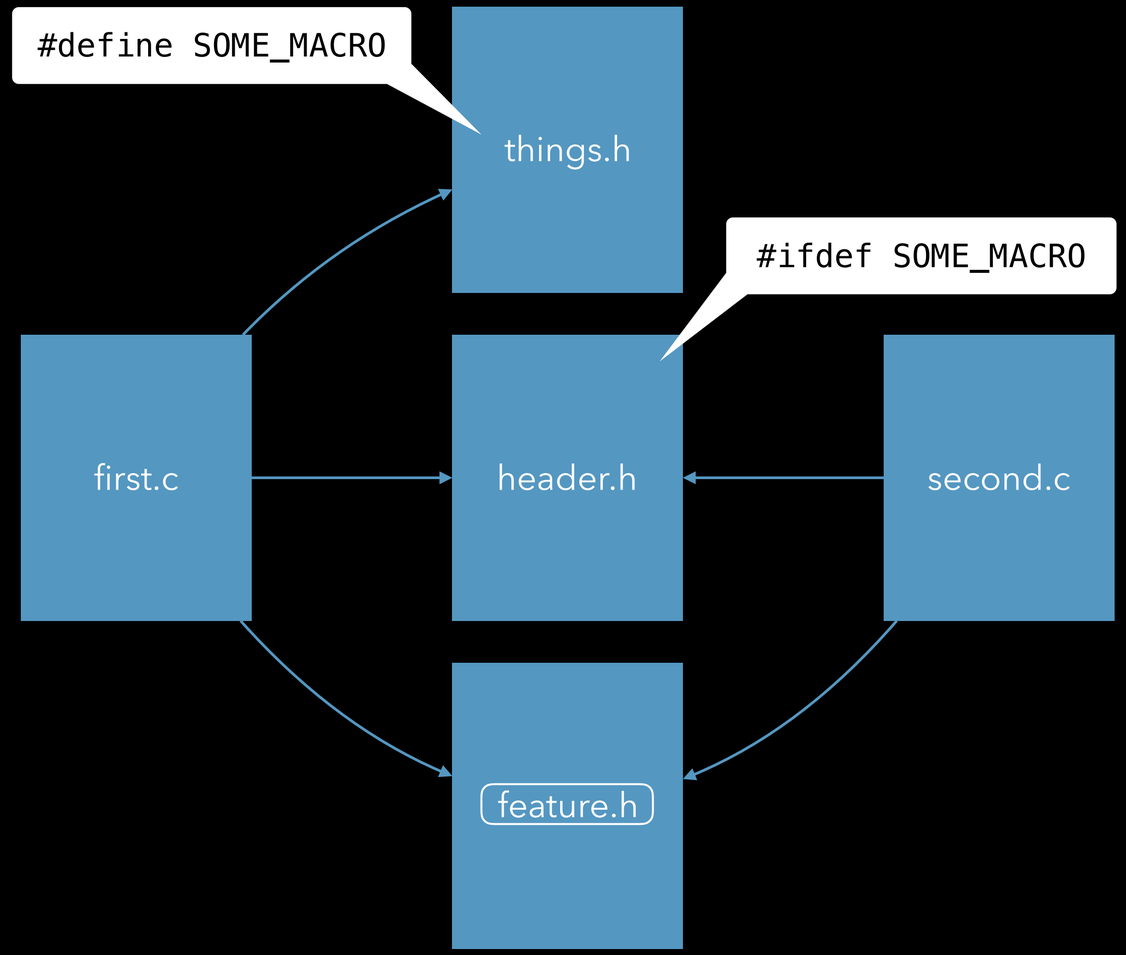
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



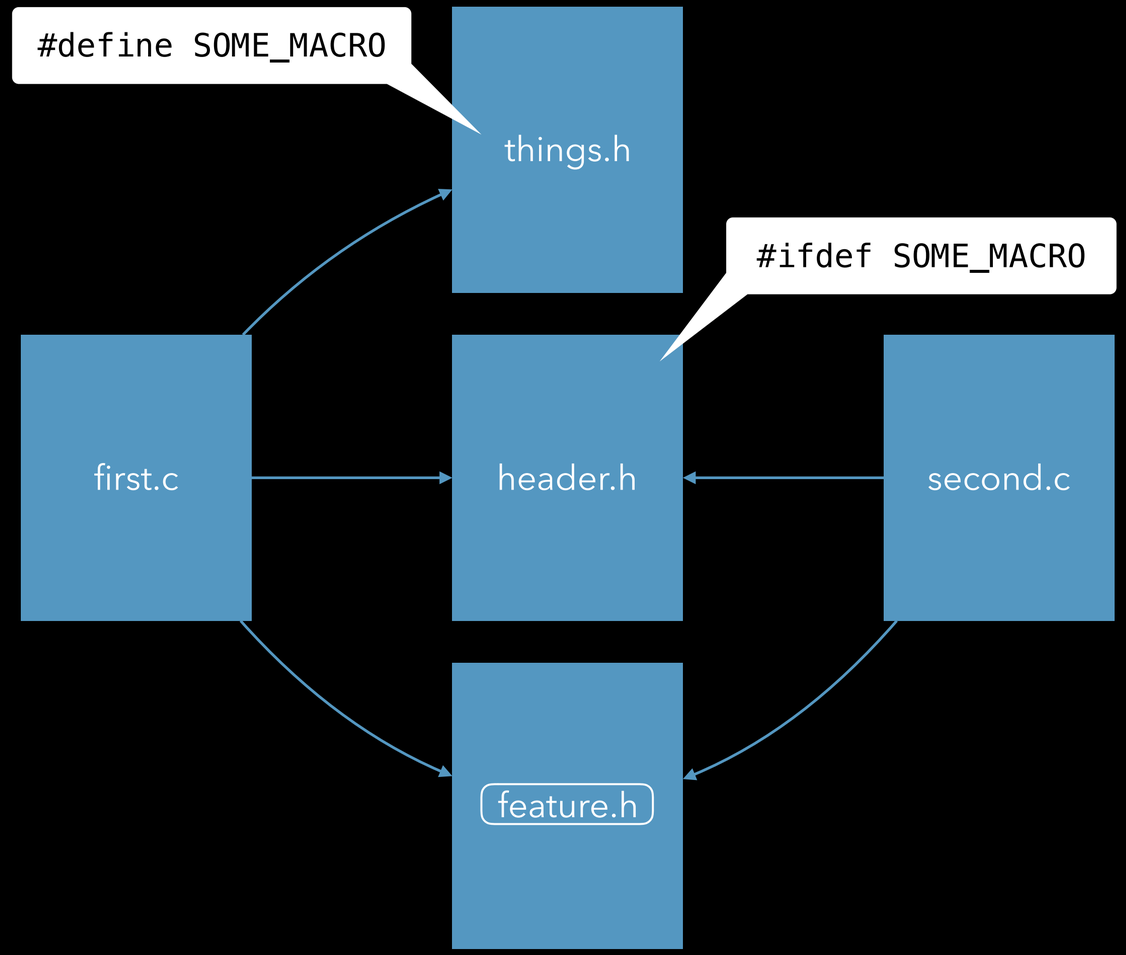
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



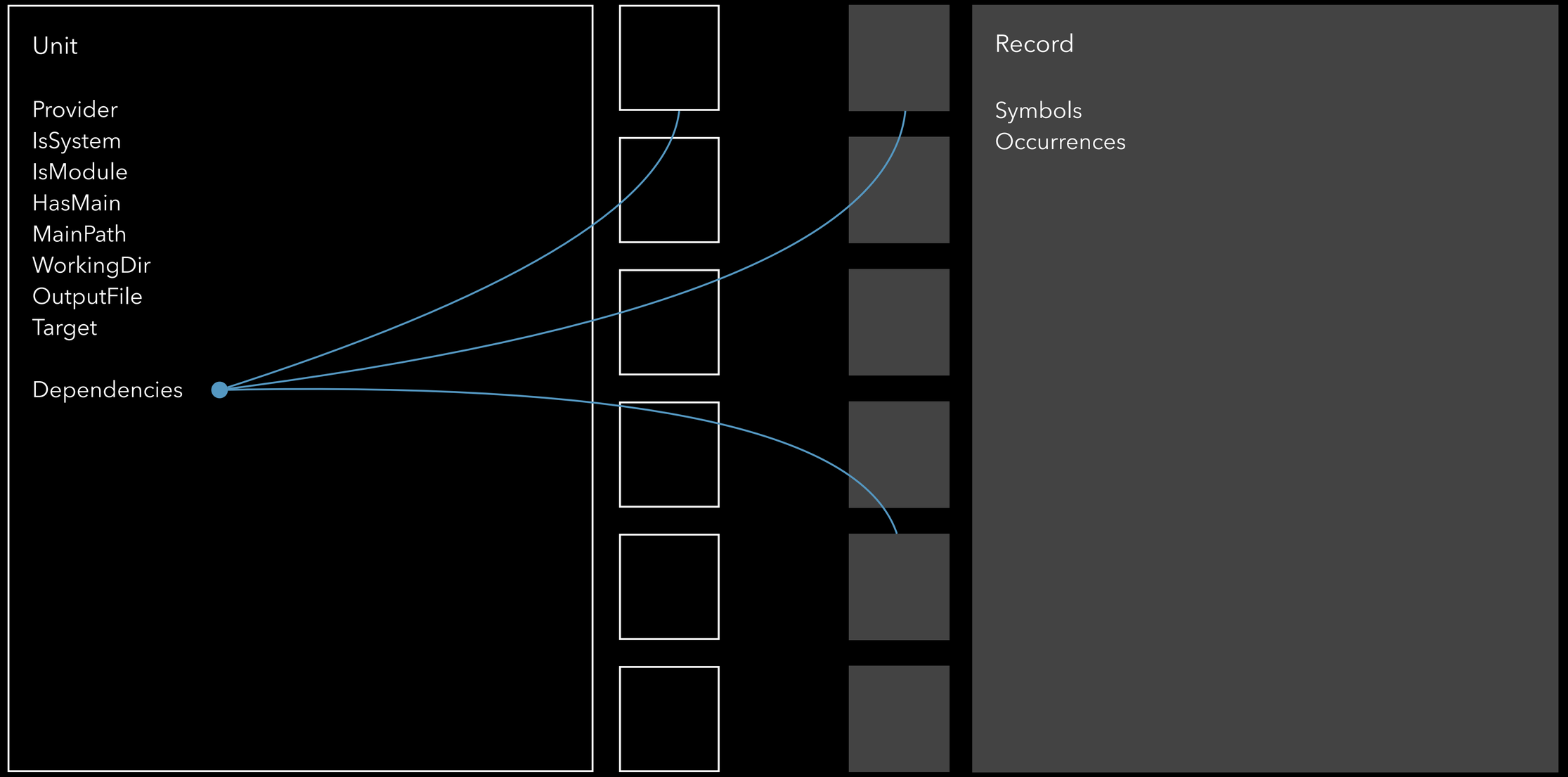
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



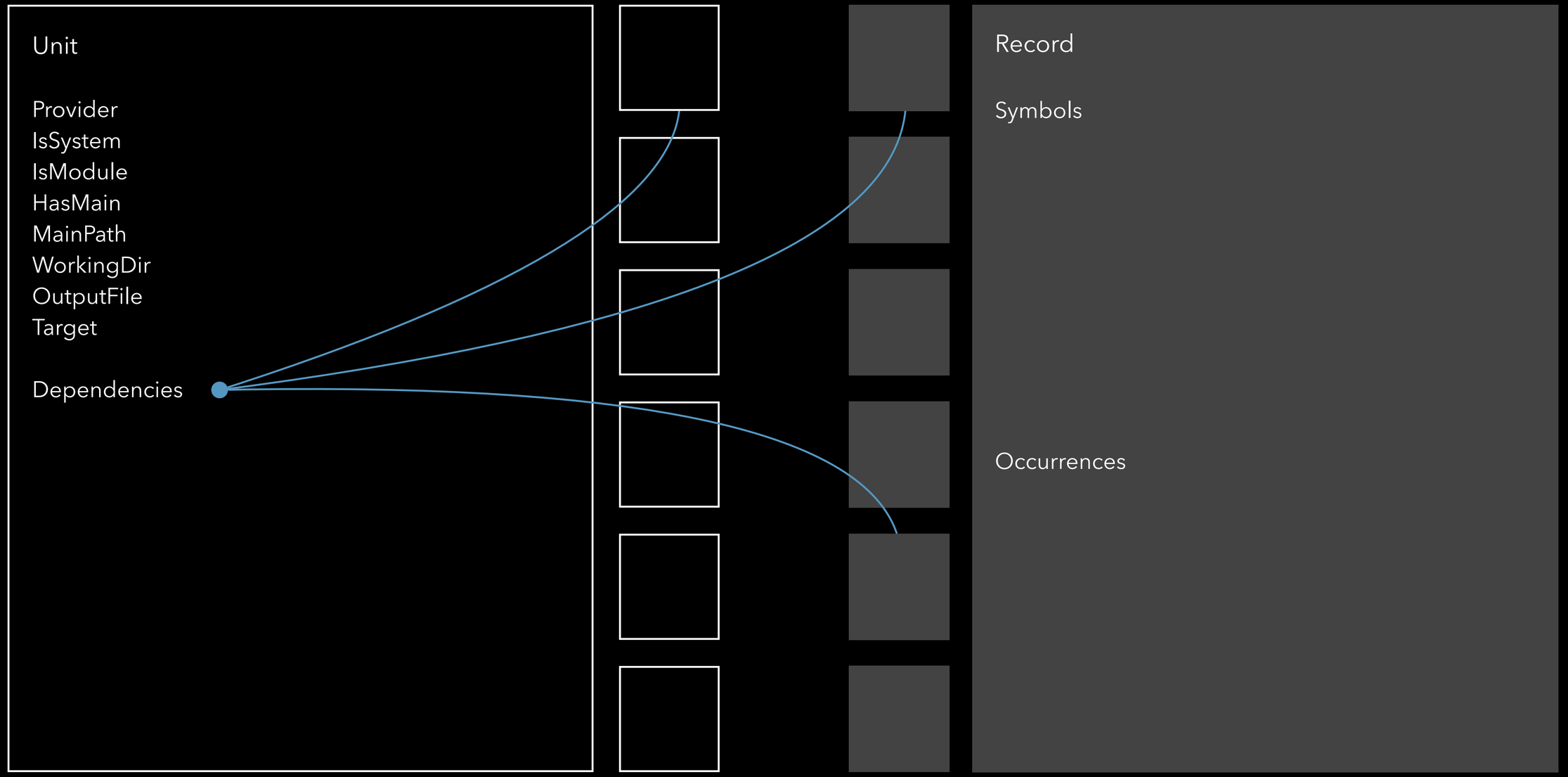
```
$ clang -o second.o -c second.c -index-store-path path/to/store
```



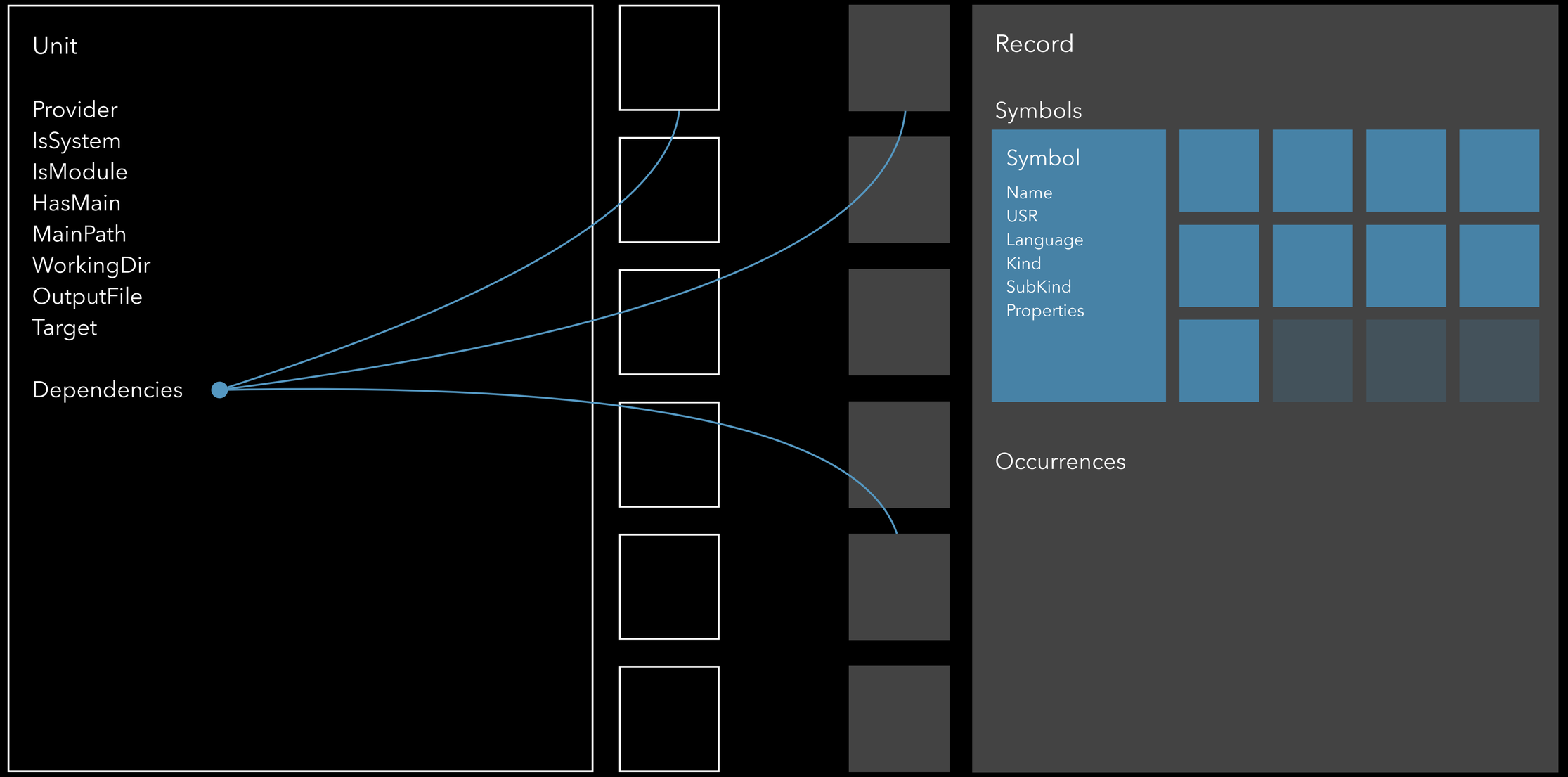
Index data model



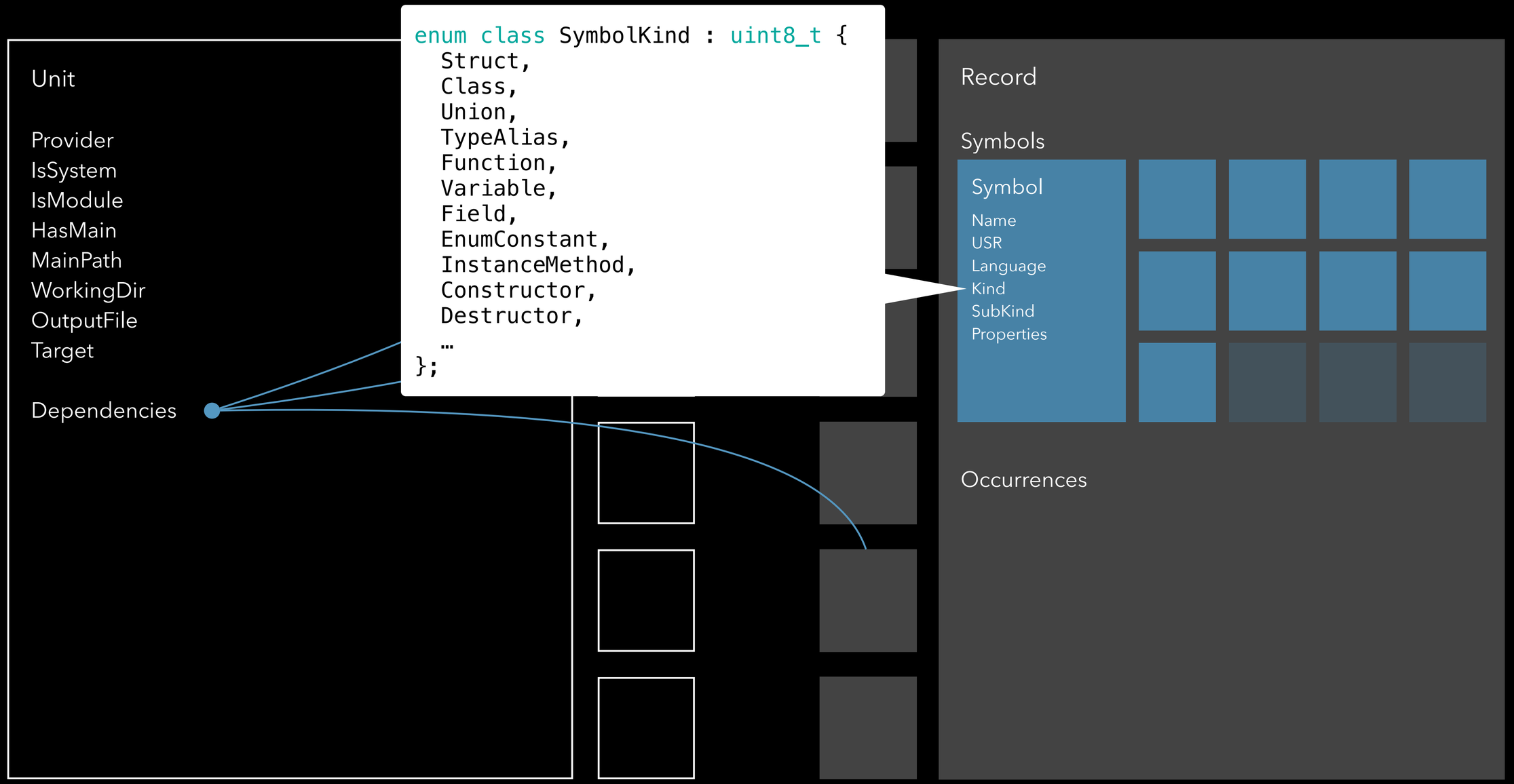
Index data model



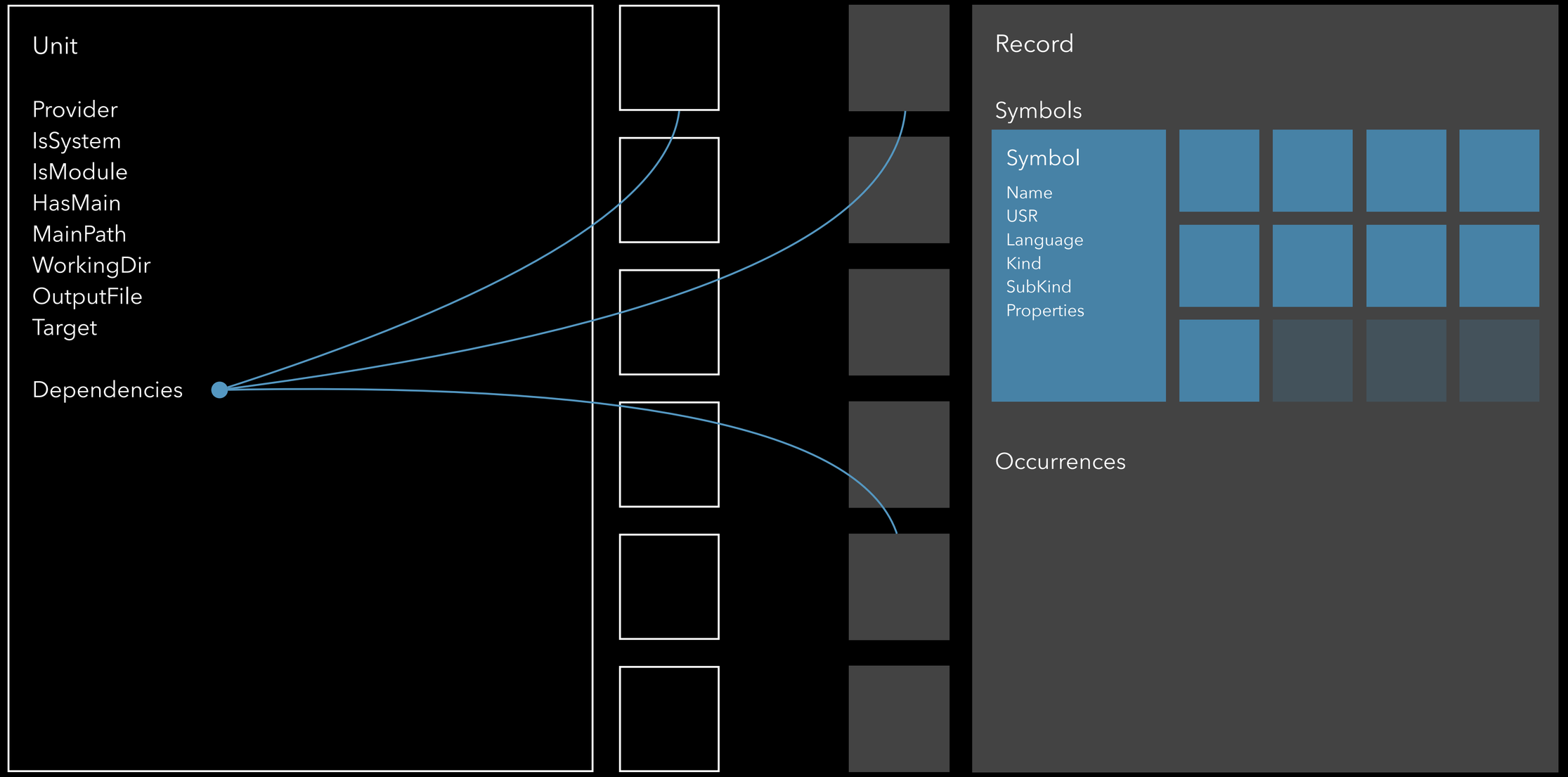
Index data model



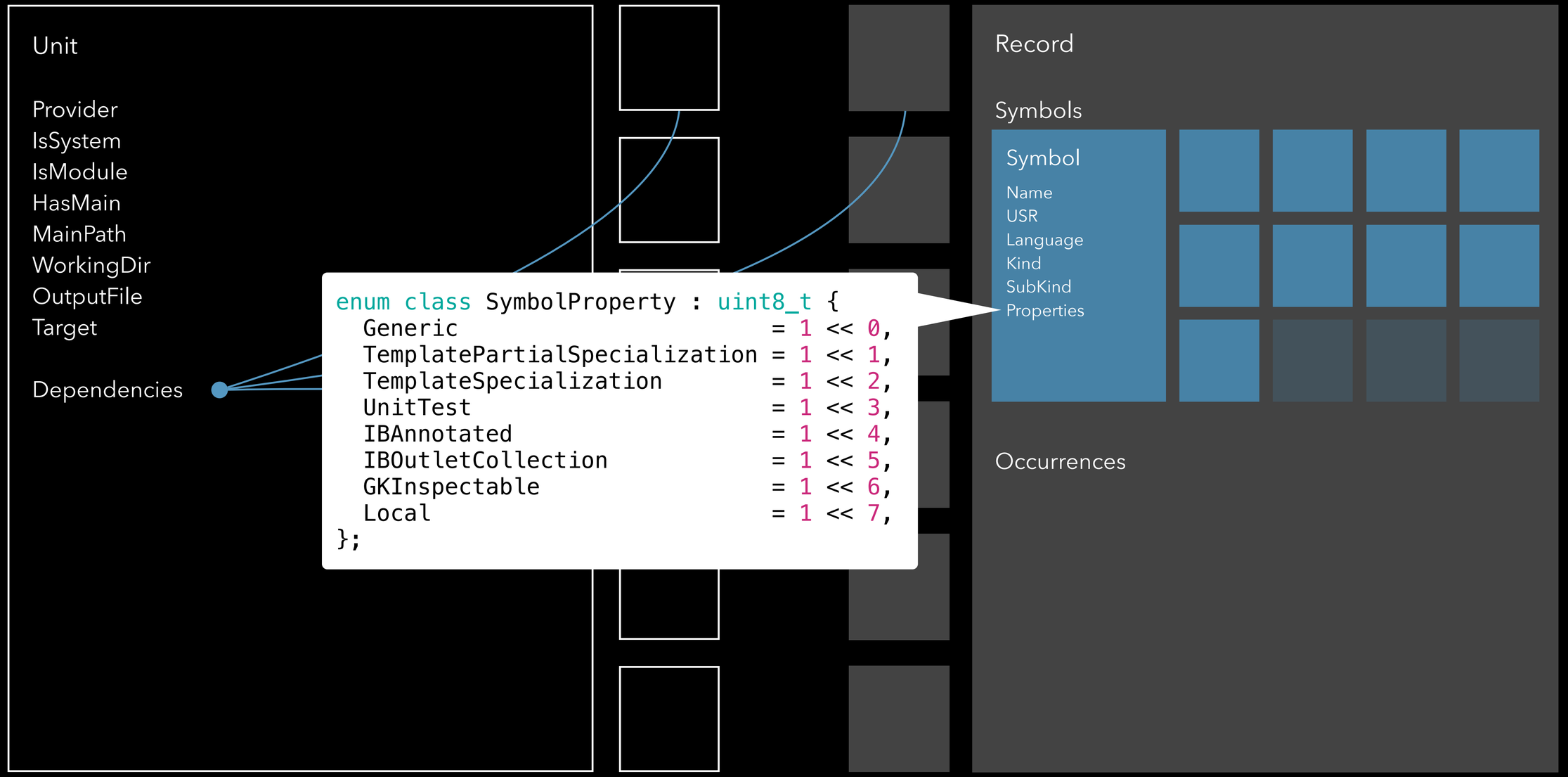
Index data model



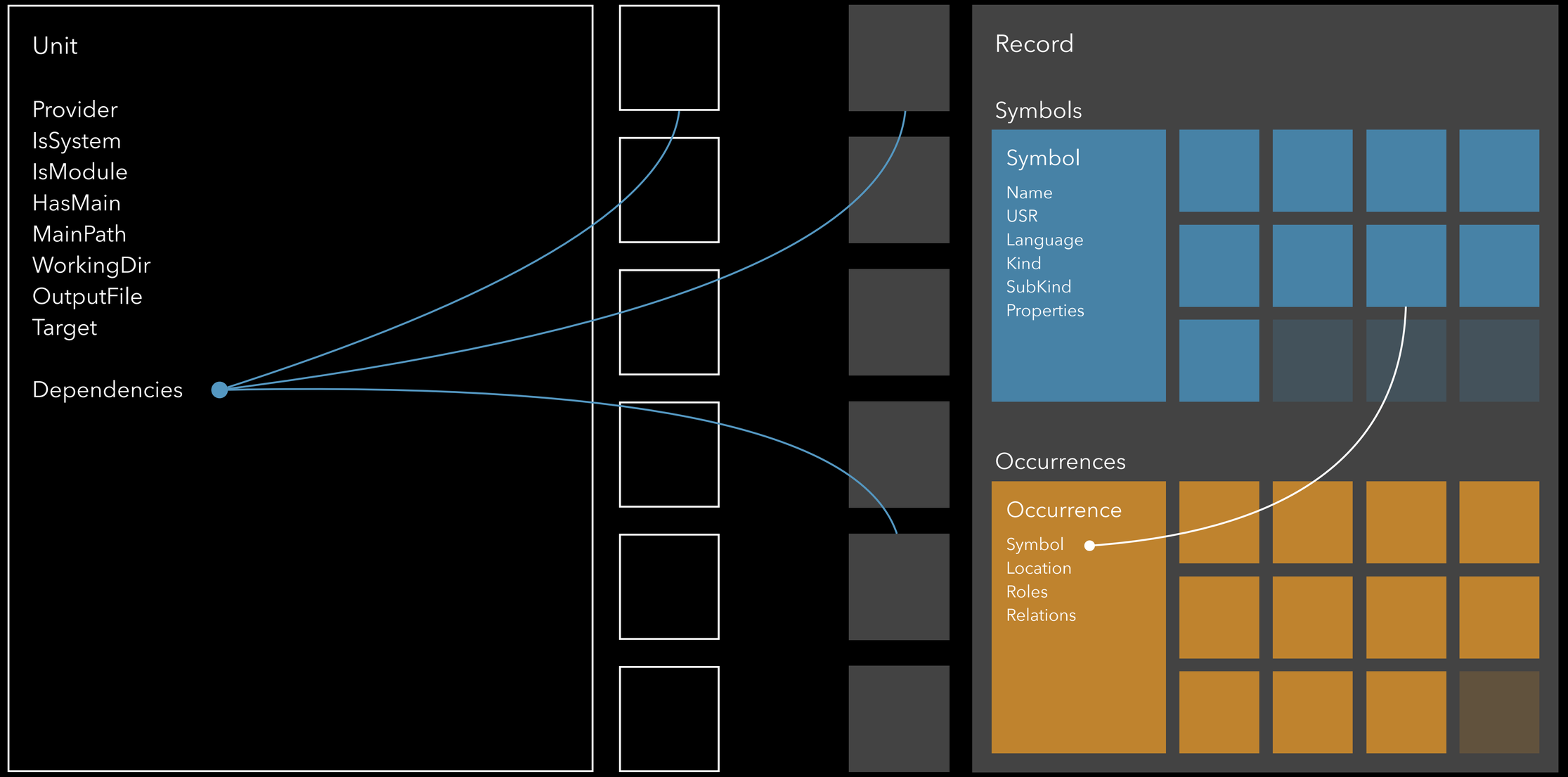
Index data model



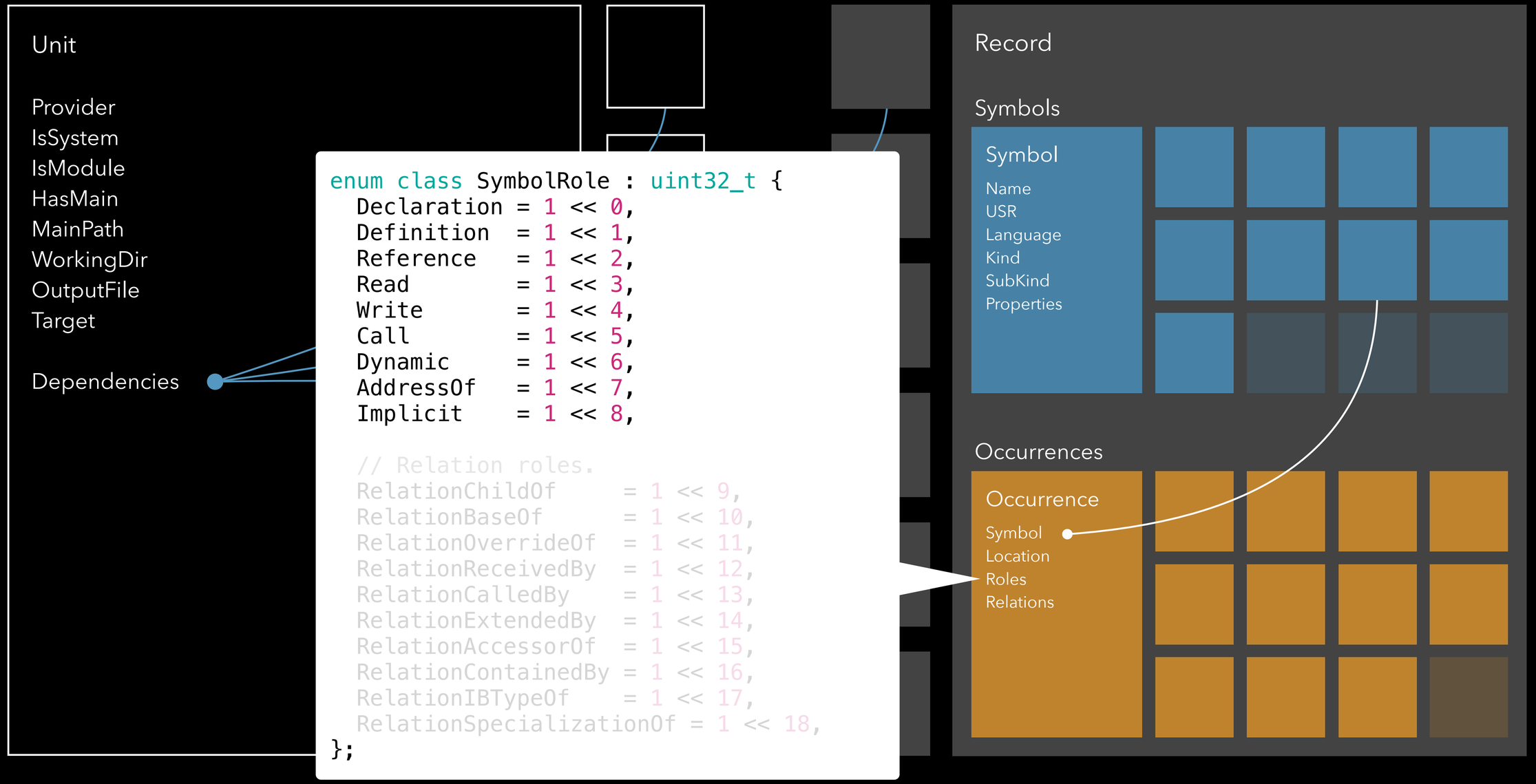
Index data model



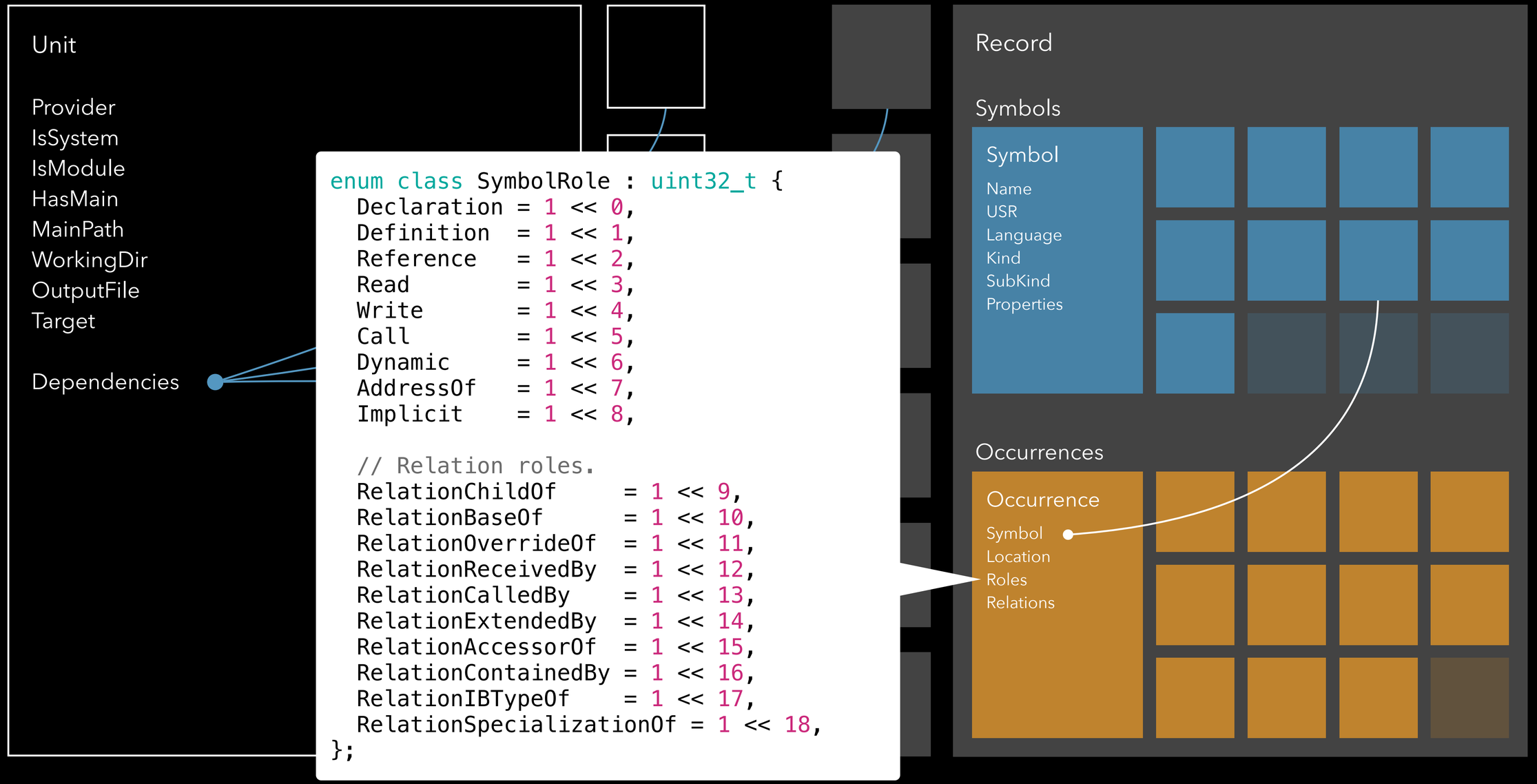
Index data model



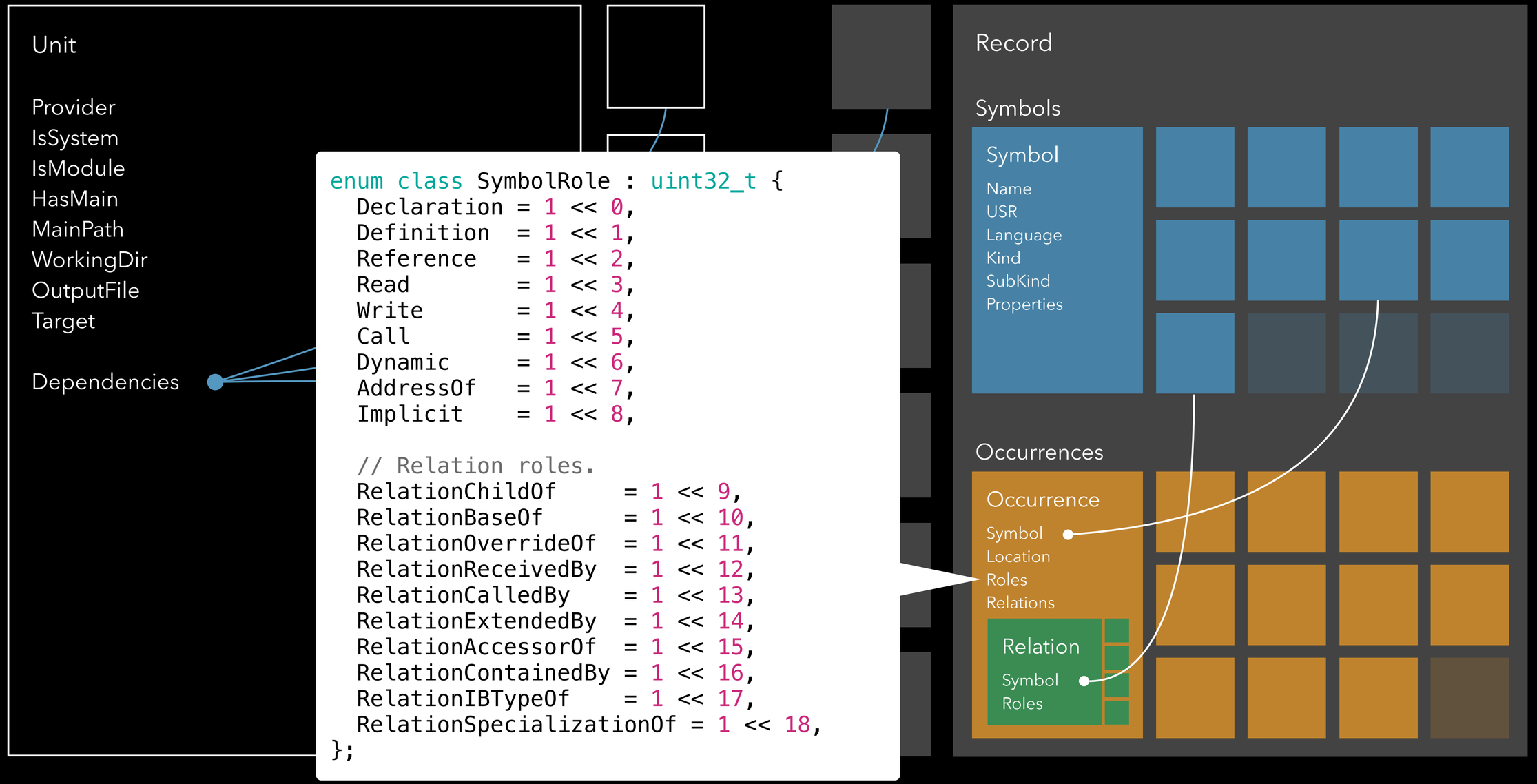
Index data model



Index data model



Index data model



```cpp
class Polygon {
protected:
  int NumberOfSides;

public:
  Polygon(int NumberOfSides)
  : NumberOfSides(NumberOfSides) {}

  int getSideCount() {
    return NumberOfSides;
  }
};

class RegularPolygon : public Polygon {
protected:
  int SideLength;

public:
  RegularPolygon(int NumberOfSides, int SideLength)
  : Polygon(NumberOfSides), SideLength(SideLength) {}

  int getPerimeter() {
    return SideLength * NumberOfSides;
  }
};
```

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

...

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

...

Occurrence

Symbol 1
Location 1:7
Roles Definition

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

...

Occurrence

Symbol 1
Location 1:7
Roles Definition

Occurrence

Symbol 2
Location 14:7
Roles Definition

...

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

...

Occurrence

Symbol 1
Location 1:7
Roles Definition

Occurrence

Symbol 1
Location 14:31
Roles Reference, RelationBaseOf
Relations

Occurrence

Symbol 2
Location 14:7
Roles Definition

...

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

...

Occurrence

Symbol 1
Location 1:7
Roles Definition

Occurrence

Symbol 1
Location 14:31
Roles Reference, RelationBaseOf
Relations

Relation

Roles RelationBaseOf
Symbol 2

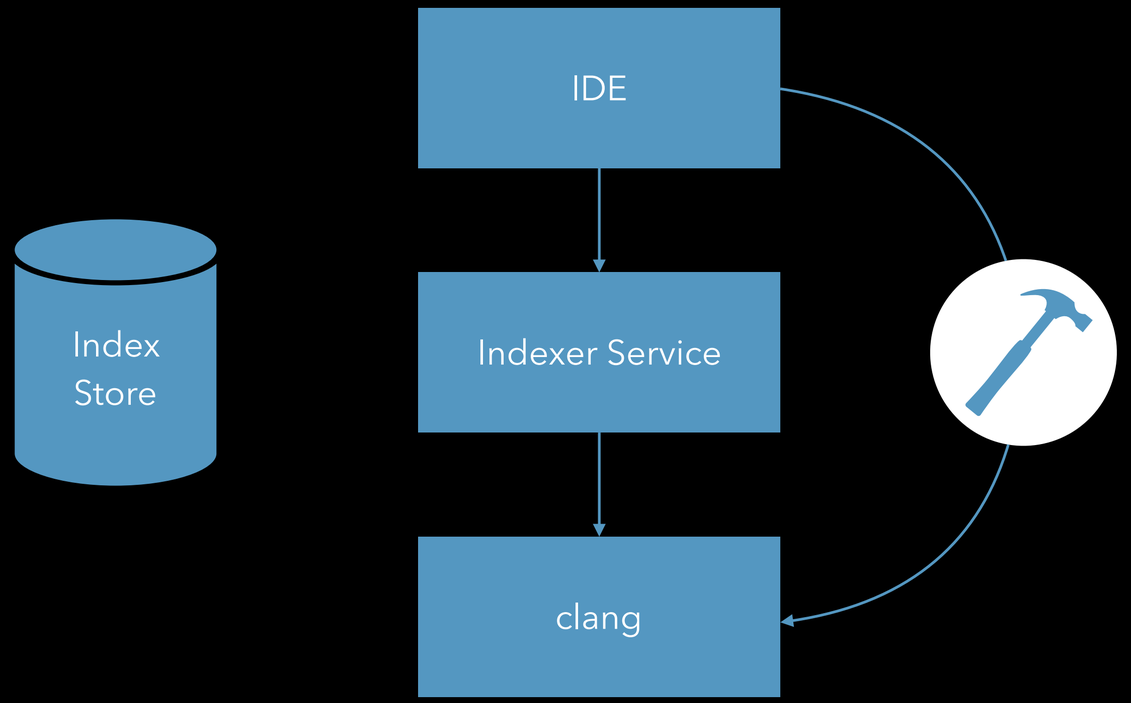
Occurrence

Symbol 2
Location 14:7
Roles Definition

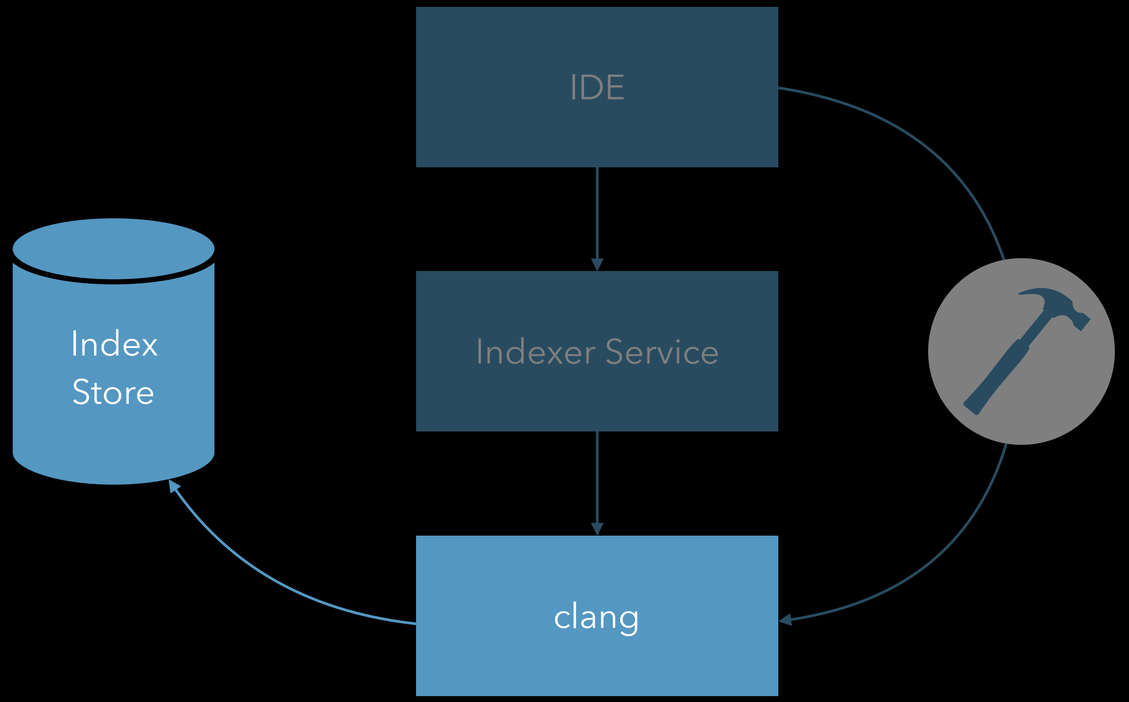
...

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

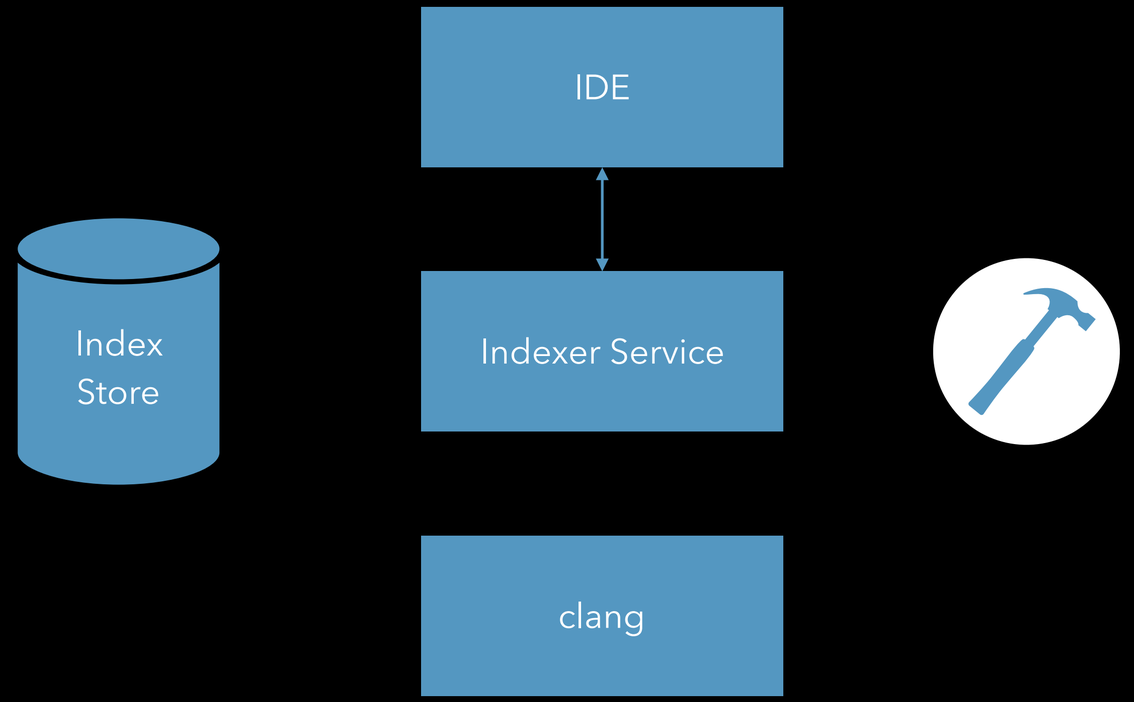
Producing index data



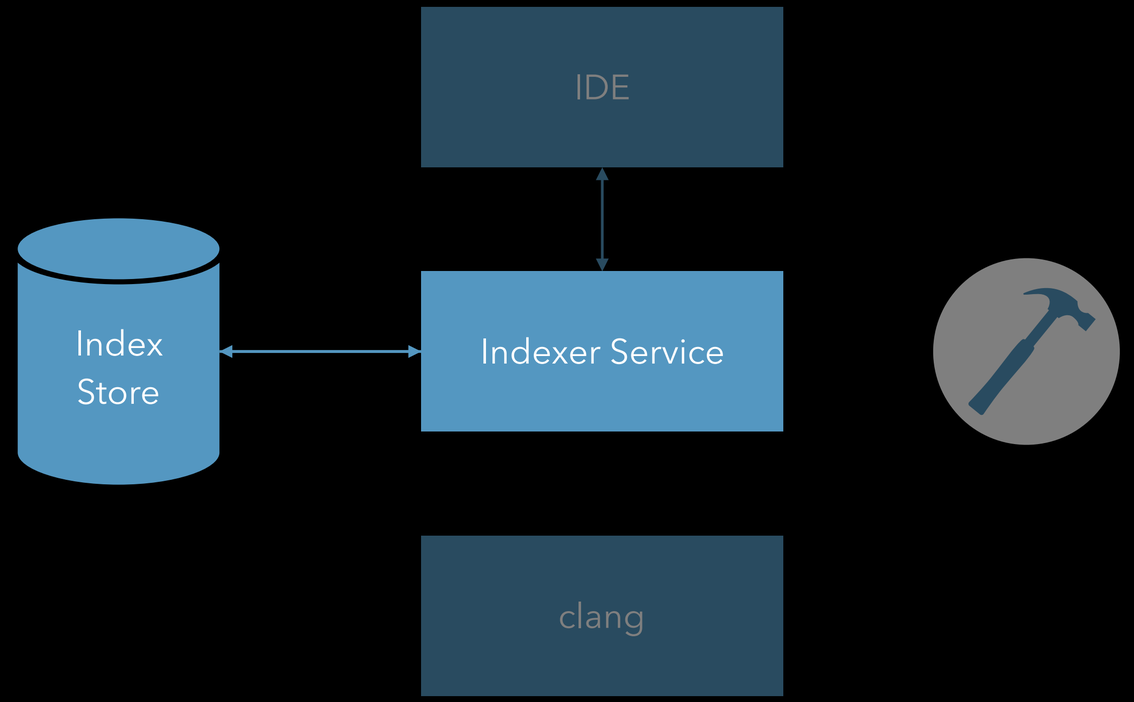
Producing index data



Consuming index data



Consuming index data



Reading the index store

Reading the index store

- New IndexStore library for clients to read and manage the store.

Reading the index store

- New IndexStore library for clients to read and manage the store.
- Located under tools/IndexStore.

Reading the index store

- New IndexStore library for clients to read and manage the store.
- Located under tools/IndexStore.
- Abstracts the directory structure and format (LLVM Bitstream).

Reading the index store

- New IndexStore library for clients to read and manage the store.
- Located under tools/IndexStore.
- Abstracts the directory structure and format (LLVM Bitstream).
- Notifies clients when units are updated, added, or removed.

Reading the index store

Not query!

- New IndexStore library for clients to read and manage the store.
- Located under tools/IndexStore.
- Abstracts the directory structure and format (LLVM Bitstream).
- Notifies clients when units are updated, added, or removed.

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

Occurrence

Symbol 1
Location 1:7
Roles Definition

Occurrence

Symbol 1
Location 14:31
Roles Reference, RelationBaseOf
Relations

Relation

Roles RelationBaseOf
Symbol 2

Occurrence

Symbol 2
Location 14:7
Roles Definition

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Find subclasses of Polygon

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

Occurrence

Symbol 1
Location 1:7
Roles Definition

Occurrence

Symbol 1
Location 14:31
Roles Reference, RelationBaseOf
Relations

Relation

Roles RelationBaseOf
Symbol 2

Occurrence

Symbol 2
Location 14:7
Roles Definition

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9    int getSideCount() {
10     return NumberOfSides;
11   }
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Find subclasses of Polygon

Symbol 1
Polygon

USR c:@S@Polygon
Language C++
Kind class

Symbol 2
RegularPolygon

USR c:@S@RegularPolygon
Language C++
Kind class

Occurrence

Symbol 1
Location 1:7
Roles Definition

Occurrence

Symbol 1
Location 14:31
Roles Reference, RelationBaseOf
Relations

Relation

Roles RelationBaseOf
Symbol 2

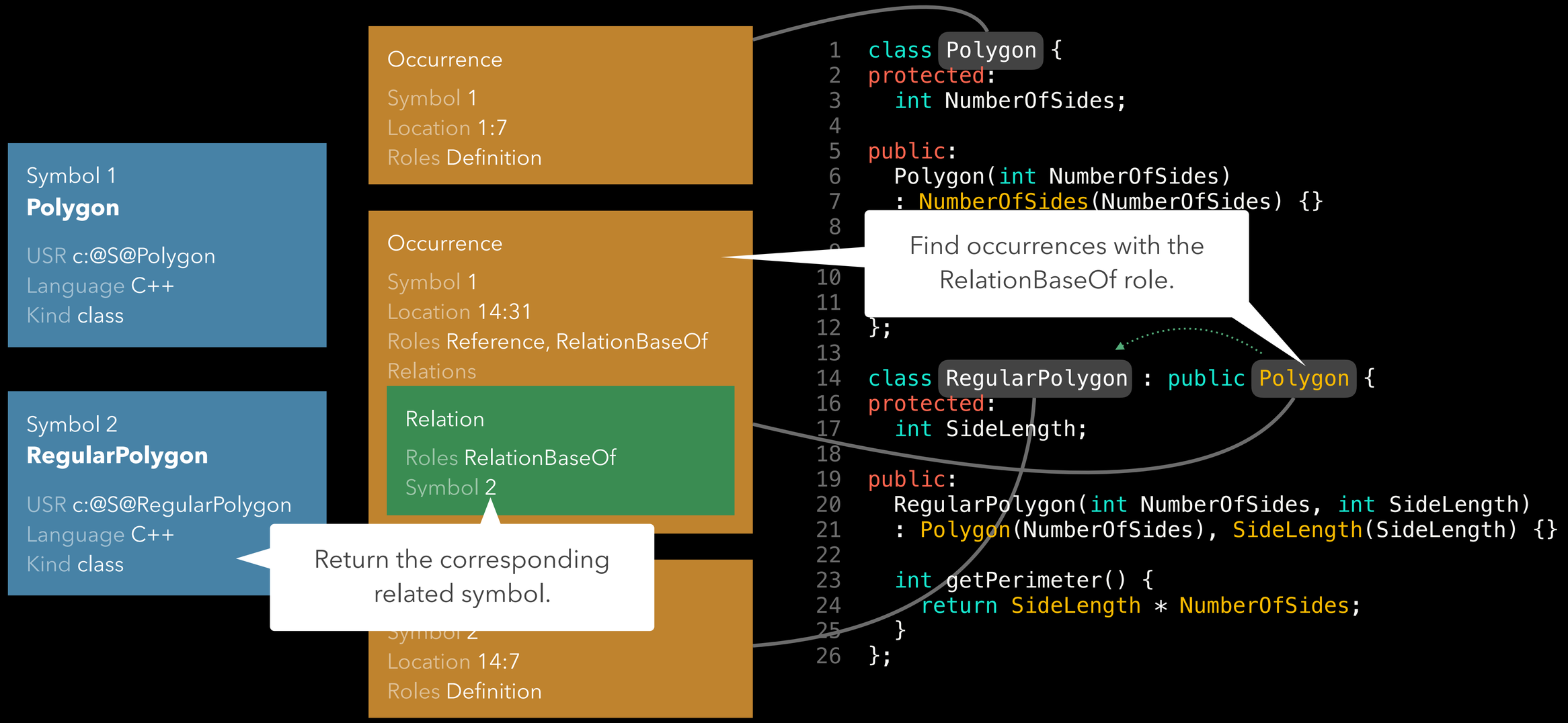
Occurrence

Symbol 2
Location 14:7
Roles Definition

```
1  class Polygon {
2  protected:
3    int NumberOfSides;
4
5  public:
6    Polygon(int NumberOfSides)
7    : NumberOfSides(NumberOfSides) {}
8
9
10
11
12 };
13
14 class RegularPolygon : public Polygon {
16 protected:
17   int SideLength;
18
19 public:
20   RegularPolygon(int NumberOfSides, int SideLength)
21   : Polygon(NumberOfSides), SideLength(SideLength) {}
22
23   int getPerimeter() {
24     return SideLength * NumberOfSides;
25   }
26 };
```

Find occurrences with the RelationBaseOf role.

Find subclasses of Polygon



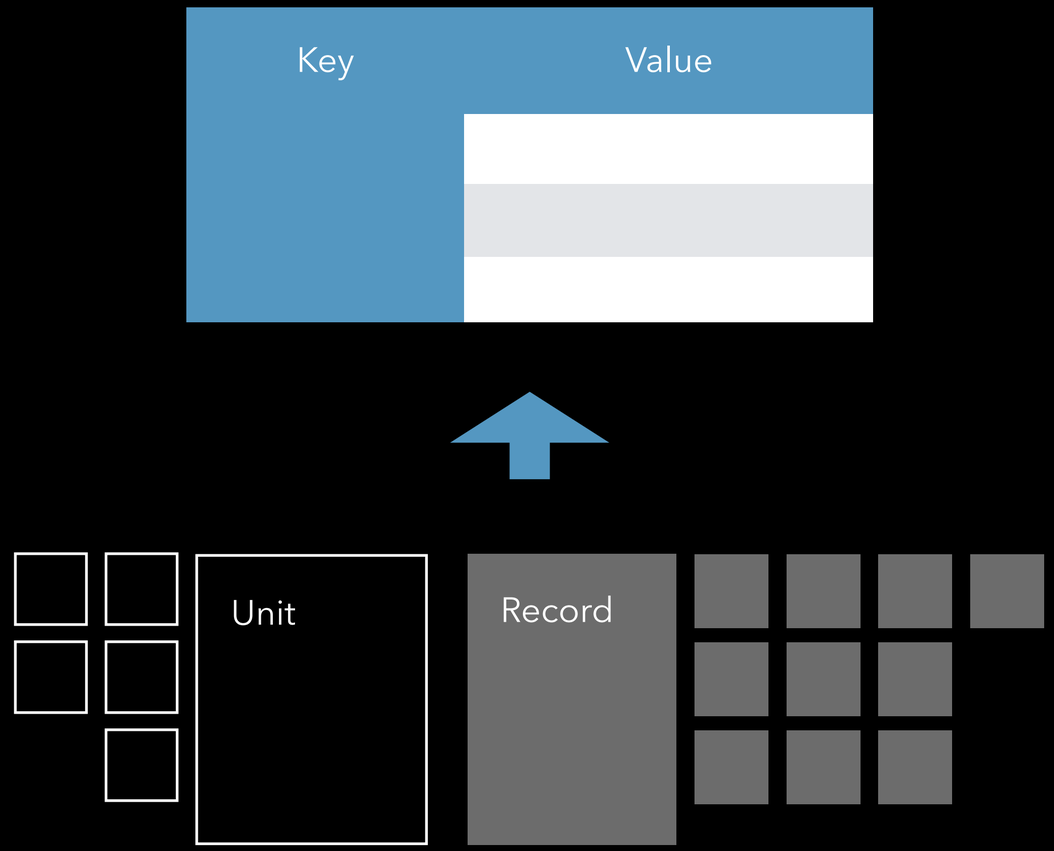
Find subclasses of Polygon

Find subclasses of Polygon

Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record
Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record
Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record
Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record	Record

Speeding things up in the indexer service

Speeding things up in the indexer service



Find subclasses of Polygon

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the RelationBaseOf role.

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the RelationBaseOf role.
3. Retrieve those occurrences and their related symbols.

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the RelationBaseOf role.
3. Retrieve those occurrences and their related symbols.
4. Find the canonical occurrences of those symbols.

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the RelationBaseOf role.
3. Retrieve those occurrences and their related symbols.
4. Find the canonical occurrences of those symbols.

i.e. Find their definitions

Find subclasses of Polygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the RelationBaseOf role.
3. Retrieve those occurrences and their related symbols.
4. Find the canonical occurrences of those symbols.

Find definitions of RegularPolygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

Find definitions of RegularPolygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.

Find definitions of RegularPolygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the Definition role.

Find definitions of RegularPolygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the Definition role.
3. Retrieve those occurrences.

Find _____ of RegularPolygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the _____ role.
3. Retrieve those occurrences.

Find _____ of RegularPolygon

USR	Record + Roles
c:@S@**Polygon**	test.cpp-2RLZQAEAVK8SU Definition, Reference, RelationBaseOf other.cpp-X8QI5PPQ303AO Reference, RelationContainedBy
c:@S@**RegularPolygon**	test.cpp-2RLZQAEAVK8SU Definition chess.h-2RLZQAEAVK8SU Reference, RelationBaseOf

1. Lookup USR.
2. Find records with the _____ role.
3. Retrieve occurre

Reference, Declaration, Read, Write, Call, etc.

Other useful mappings

Other useful mappings

Symbol Name	USRs
Shape	c:@S@Shape, c:@T@Sh…

Search symbols by name

Other useful mappings

Symbol Name	USRs
Shape	c:@S@Shape, c:@T@Sh…

Search symbols by name

Source file	Dependent units
things.h	first.o-XXX, second.o-XXX

Units to re-index when header changes

Other useful mappings

Symbol Name	USRs
Shape	c:@S@Shape, c:@T@Sh…

Search symbols by name

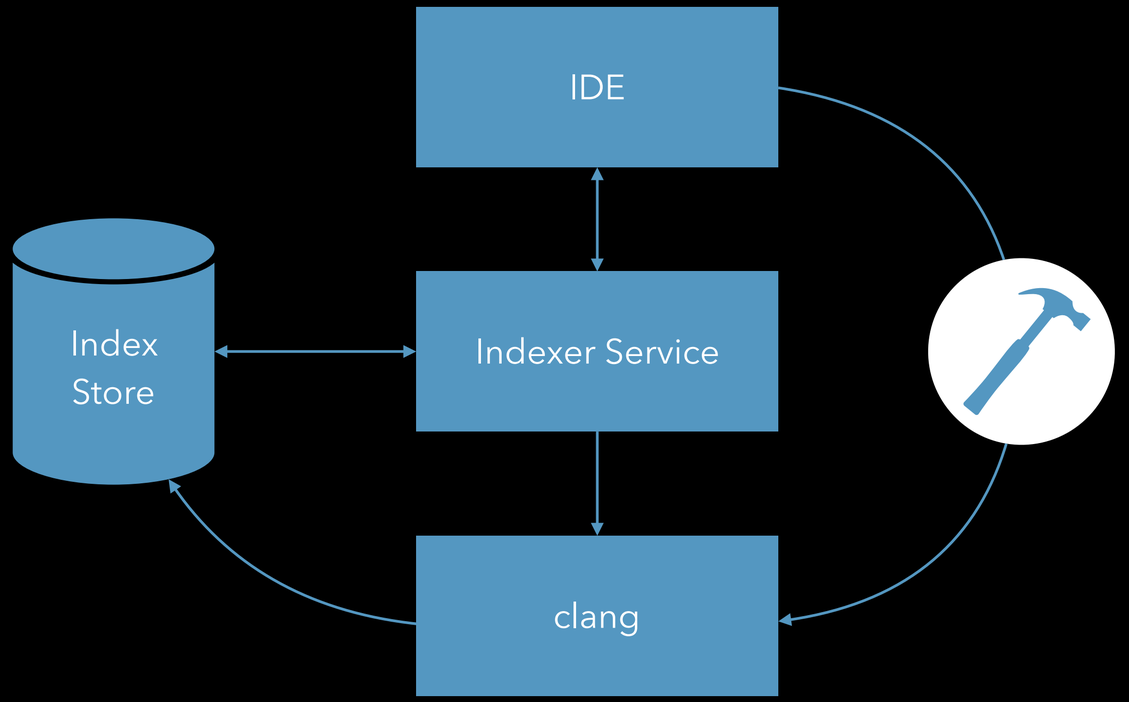
Record	Dependent units
feature.h-XXX	first.o-XXX

Remove data from unreferenced records

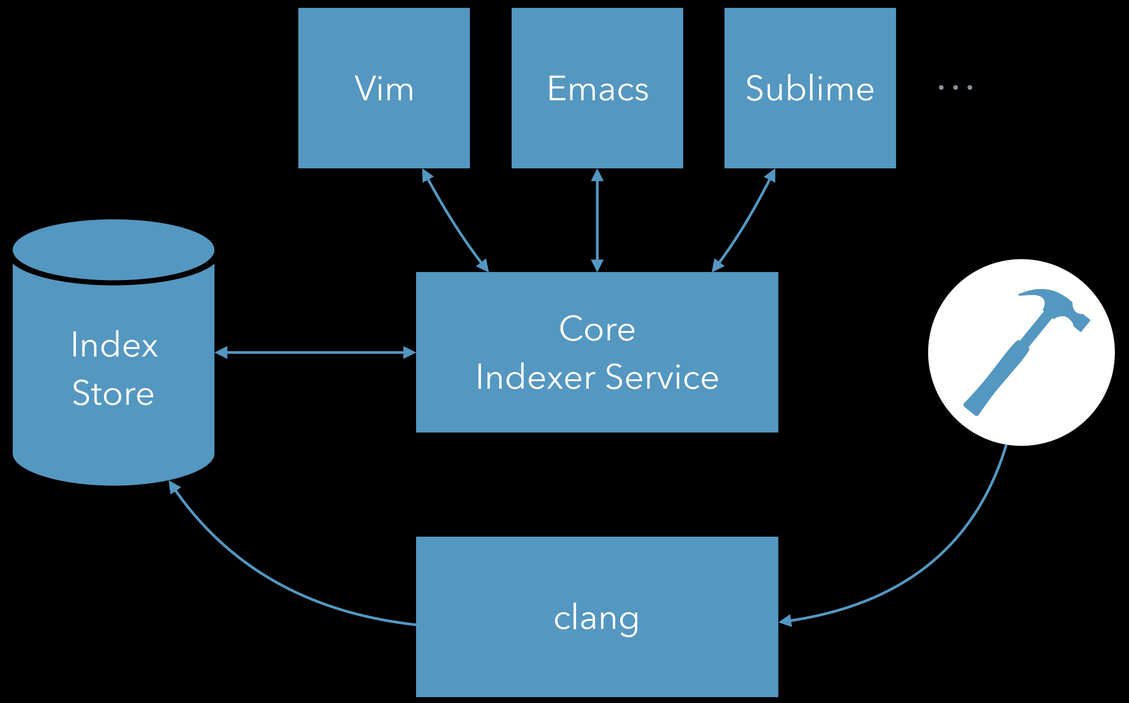
Source file	Dependent units
things.h	first.o-XXX, second.o-XXX

Units to re-index when header changes

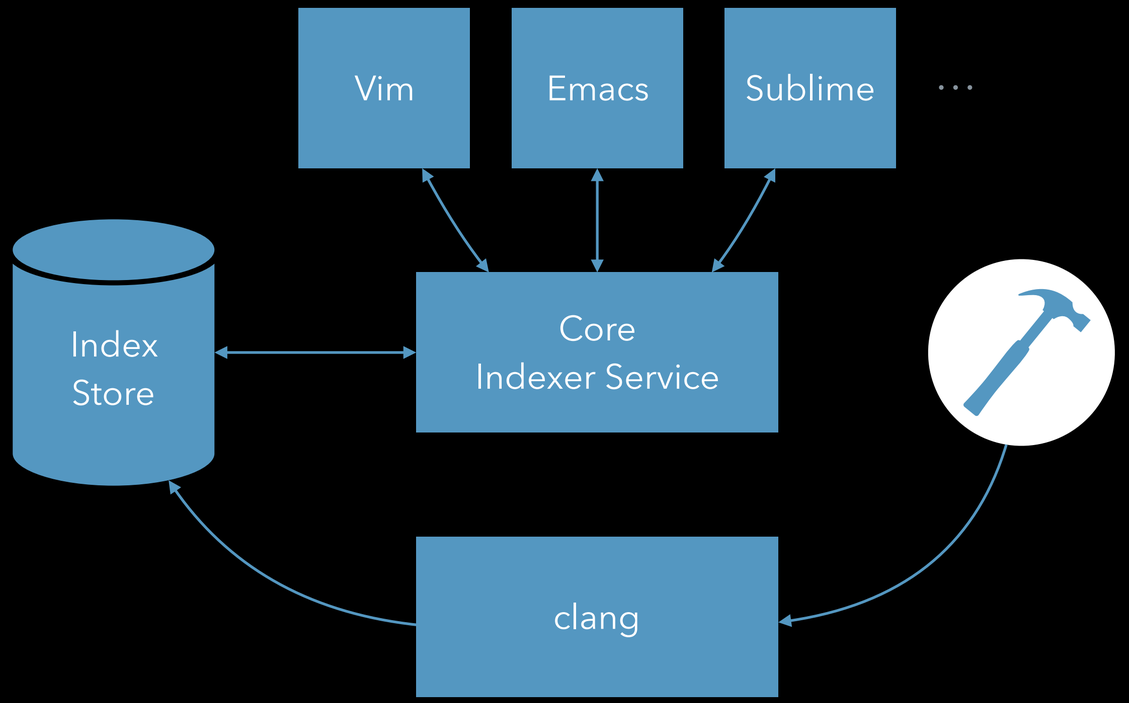
Indexing in the IDE



General indexing infrastructure

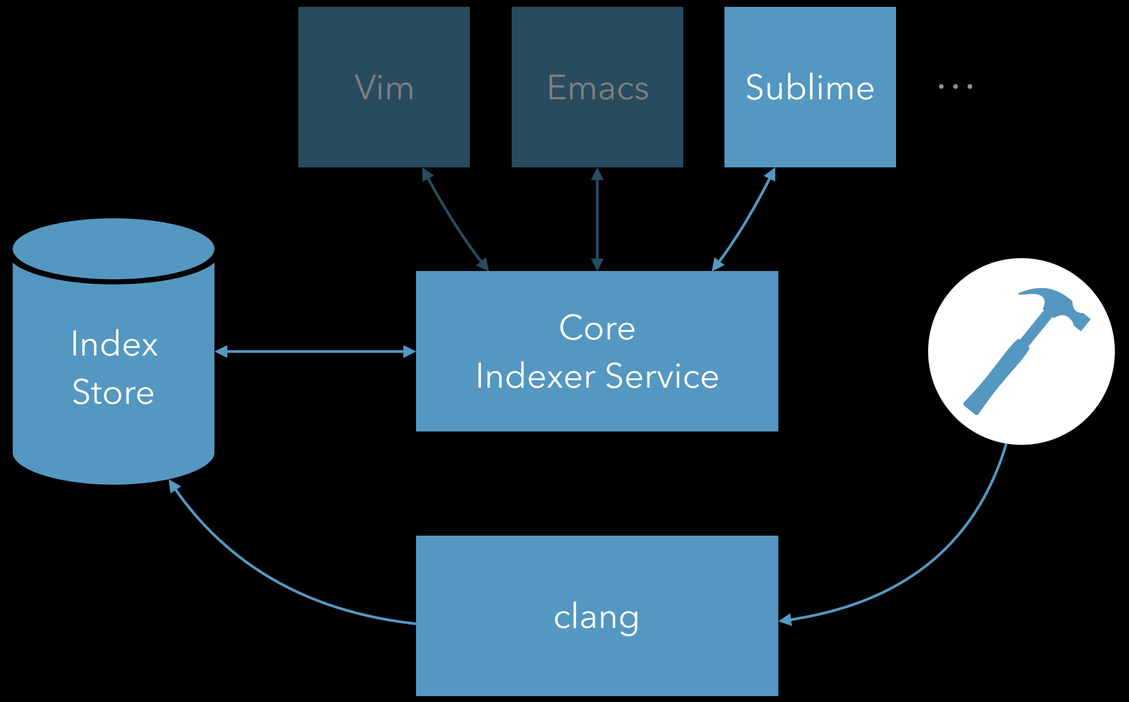


General indexing infrastructure



General indexing infrastructure

Demo!



Current status

Current status

- Clang -index-store-path and IndexStore library upstream in progress.

Current status

- Clang -index-store-path and IndexStore library upstream in progress.
- Try it out now in our swift-clang repo at:

https://github.com/apple/swift-clang

Current status

- Clang -index-store-path and IndexStore library upstream in progress.
- Try it out now in our swift-clang repo at:

  https://github.com/apple/swift-clang
- Planning to upstream basic indexer service along with editor plugins for vim, emacs, and sublime.

Current status

- Clang -index-store-path and IndexStore library upstream in progress.
- Try it out now in our swift-clang repo at:

  https://github.com/apple/swift-clang
- Planning to upstream basic indexer service along with editor plugins for vim, emacs, and sublime.
- To follow along and for more info check out the *RFC: Adding index-while-building support to Clang* thread on the cfe-dev mailing list.

Refactoring

Refactoring agenda

New refactoring engine

Improved IDE support

clang-refactor

Implementing “extract function”

Refactoring library components

Status

Ideas for future contributors

Refactoring

Refactoring

Let’s start with an example

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
```

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};




float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
```

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};




float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
```

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};


          r.width * r.height


float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return                          / (r2.width * r2.height);
};
```

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};


          r.width * r.height


float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return          / (r2.width * r2.height);
};
```

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return area(r) / (r2.width * r2.height);
};
```

Refactoring

Let’s start with an example

```
struct Rectangle {
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return area(r) / (r2.width * r2.height);
};
```

An “extract function” refactoring

Refactoring

User requirements

Refactoring

User requirements

- One user lives in an IDE
  - ➡ Clang must provide “extract function” to the IDE

Refactoring

User requirements

- One user lives in an IDE
  - ➡ Clang must provide “extract function” to the IDE
- Another user swears by command-line tools
  - ➡ Clang must provide a refactoring tool that can “extract”

New refactoring engine

New refactoring engine

Overview

New refactoring engine

Overview

- Refactoring library in `libTooling`

New refactoring engine

Overview

- Refactoring library in `libTooling`
- New refactoring actions

New refactoring engine

Overview

- Refactoring library in `libTooling`
- New refactoring actions
- New reusable refactoring components

New refactoring engine

Overview

- Refactoring library in `libTooling`
- New refactoring actions
- New reusable refactoring components
- Handles AST selection and simplifies editor bindings

New refactoring engine

Overview

- Refactoring library in `libTooling`
- New refactoring actions
- New reusable refactoring components
- Handles AST selection and simplifies editor bindings

➡ Simple IDE integration for “extract function”

New refactoring engine

New command-line tool

New refactoring engine

New command-line tool

- `clang-refactor`

New refactoring engine

New command-line tool

- `clang-refactor`
- Supports all new refactorings

New refactoring engine

New command-line tool

- `clang-refactor`
- Supports all new refactorings
- Local source transformations work

New refactoring engine

New command-line tool

- `clang-refactor`
- Supports all new refactorings
- Local source transformations work

➡ Automatic support for “extract function”

Using clang-refactor

```
struct Rectangle {
    float x, y, width, height;
};

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
```

Using clang-refactor

```
struct Rectangle {
    float x, y, width, height;
};

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
```

```
clang-refactor extract -selection=test.cpp:6:13-6:31 -name=area test.cpp
```

Using clang-refactor

```
struct Rectangle {
    float x, y, width, height;
};

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
```

```
clang-refactor extract -selection=test.cpp:6:13-6:31 -name=area test.cpp
```

Each action gets its own subcommand

New refactoring engine

Simplified IDE integration process

New refactoring engine

Simplified IDE integration process

- Existing refactorings: `clang-rename`, `clang-reorder-fields`, ...

New refactoring engine

Simplified IDE integration process

- Existing refactorings: `clang-rename`, `clang-reorder-fields`, …
  - No `libclang` / `clangd` integration

New refactoring engine

Simplified IDE integration process

- Existing refactorings: `clang-rename`, `clang-reorder-fields`, …
  - No `libclang` / `clangd` integration
- New refactorings implemented in `libTooling`

New refactoring engine

Simplified IDE integration process

- Existing refactorings: `clang-rename`, `clang-reorder-fields`, …
  - No `libclang` / `clangd` integration
- New refactorings implemented in `libTooling`
- Refactoring library manages integration with editor services

New refactoring engine

Simplified IDE integration process

- Existing refactorings: `clang-rename`, `clang-reorder-fields`, …
  - No `libclang` / `clangd` integration
- New refactorings implemented in `libTooling`
- Refactoring library manages integration with editor services

➡ Easy to integrate with `libclang` / `clangd`

New refactoring engine

clang-refactor *-vs-* existing tools

New refactoring engine

clang-refactor *-vs-* existing tools

- `clang-refactor` *vs* `clang-rename`, `clang-reorder-fields`, ...
  - How many tools are needed?

New refactoring engine

clang-refactor *-vs-* existing tools

- `clang-refactor` *vs* `clang-rename`, `clang-reorder-fields`, ...
  - How many tools are needed?
- Single tool enables:

New refactoring engine

clang-refactor *-vs-* existing tools

- `clang-refactor` *vs* `clang-rename`, `clang-reorder-fields`, ...
  - How many tools are needed?
- Single tool enables:
  - Common command-line interface with action-specific options

New refactoring engine

clang-refactor *-vs-* existing tools

- `clang-refactor` *vs* `clang-rename`, `clang-reorder-fields`, ...
  - How many tools are needed?
- Single tool enables:
  - Common command-line interface with action-specific options
  - One set of editor plugins

New refactoring engine

clang-refactor *-vs-* existing tools

- `clang-refactor` *vs* `clang-rename`, `clang-reorder-fields`, ...
  - How many tools are needed?
- Single tool enables:
  - Common command-line interface with action-specific options
  - One set of editor plugins
  - No duplication of indexing infrastructure integration efforts

New refactoring engine

Refactoring stages

New refactoring engine

Refactoring stages

- Before: monolithic `ASTConsumer` design

New refactoring engine

Refactoring stages

- Before: monolithic `ASTConsumer` design
- Refactoring library splits refactoring into stages

New refactoring engine

Refactoring stages

- Before: monolithic `ASTConsumer` design
- Refactoring library splits refactoring into stages
  - Initiation verifies options, handles AST selection, matches AST

New refactoring engine

Refactoring stages

- Before: monolithic `ASTConsumer` design
- Refactoring library splits refactoring into stages
  - Initiation verifies options, handles AST selection, matches AST
  - Refactoring creates source replacements

New refactoring engine

Refactoring stages

- Before: monolithic `ASTConsumer` design
- Refactoring library splits refactoring into stages
  - Initiation verifies options, handles AST selection, matches AST
  - Refactoring creates source replacements

➡ Editor clients use quick initiation to find available actions for selection

New refactoring engine

Refactoring stages

- Before: monolithic `ASTConsumer` design
- Refactoring library splits refactoring into stages
  - Initiation verifies options, handles AST selection, matches AST
  - Refactoring creates source replacements

➡ Editor clients use quick initiation to find available actions for selection

➡ `clang-refactor` examines initiation to create command-line interface

Implementing “extract function”

Breakdown of extract

Extract

Extract method

Extract function

```
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const R
    return area(r) / (r2.width * r2.height);
}
```

Breakdown of extract

Extract

Extract method

Extract function

```
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const R
    return area(r) / (r2.width * r2.height);
}
```

Breakdown of extract

Extract

`class ExtractRefactoringAction`

Extract method

Extract function

```
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const R
    return area(r) / (r2.width * r2.height);
}
```

Breakdown of extract

Extract

Extract method

Extract function

```
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const R
    return area(r) / (r2.width * r2.height);
}
```

`class ExtractRefactoringAction`

`class ExtractFunction`

Breakdown of extract

Extract

Extract method

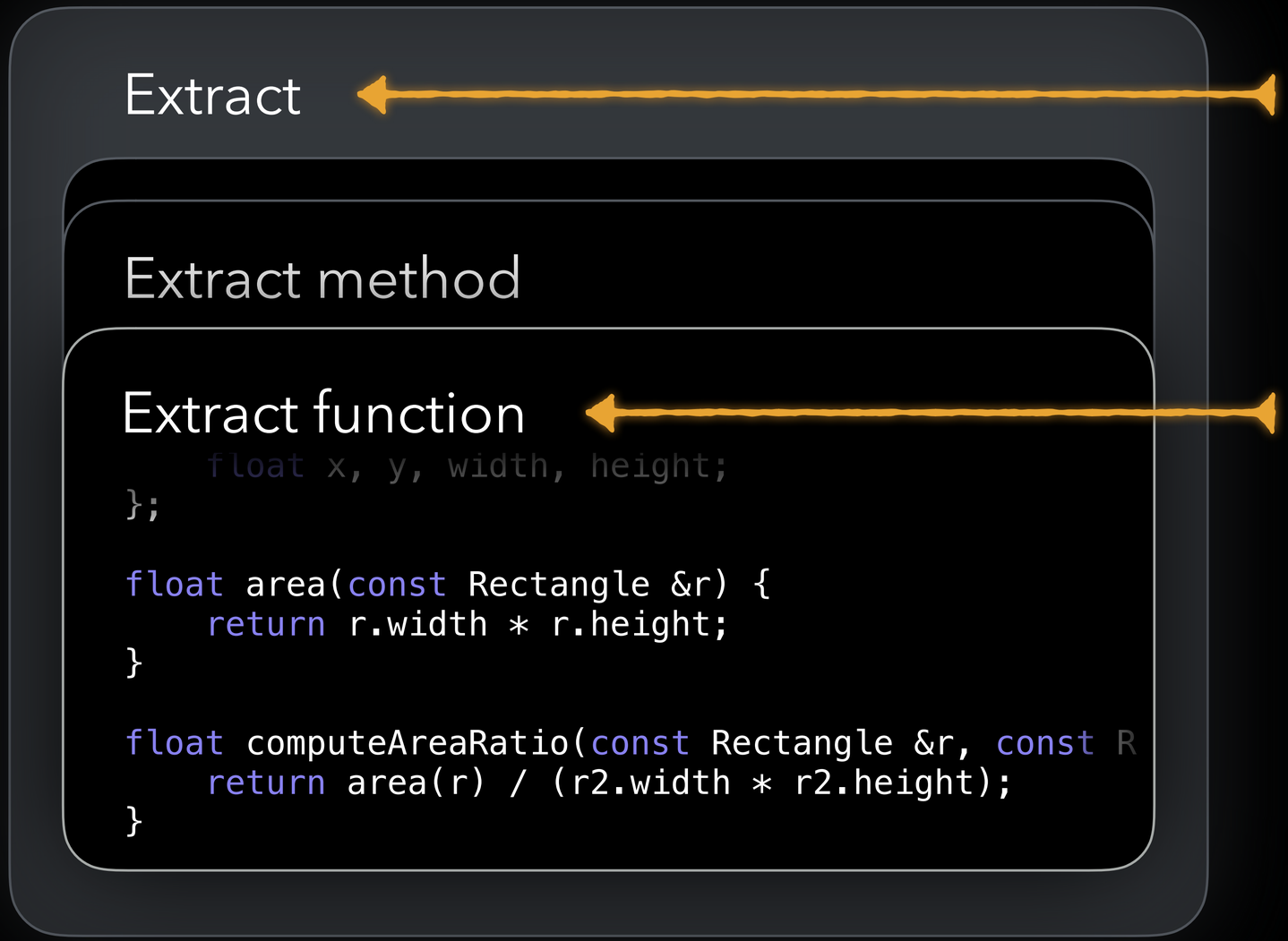
Extract function

```
    float x, y, width, height;
};

float area(const Rectangle &r) {
    return r.width * r.height;
}

float computeAreaRatio(const Rectangle &r, const R
    return area(r) / (r2.width * r2.height);
}
```

`class ExtractRefactoringAction`

`class ExtractFunction`

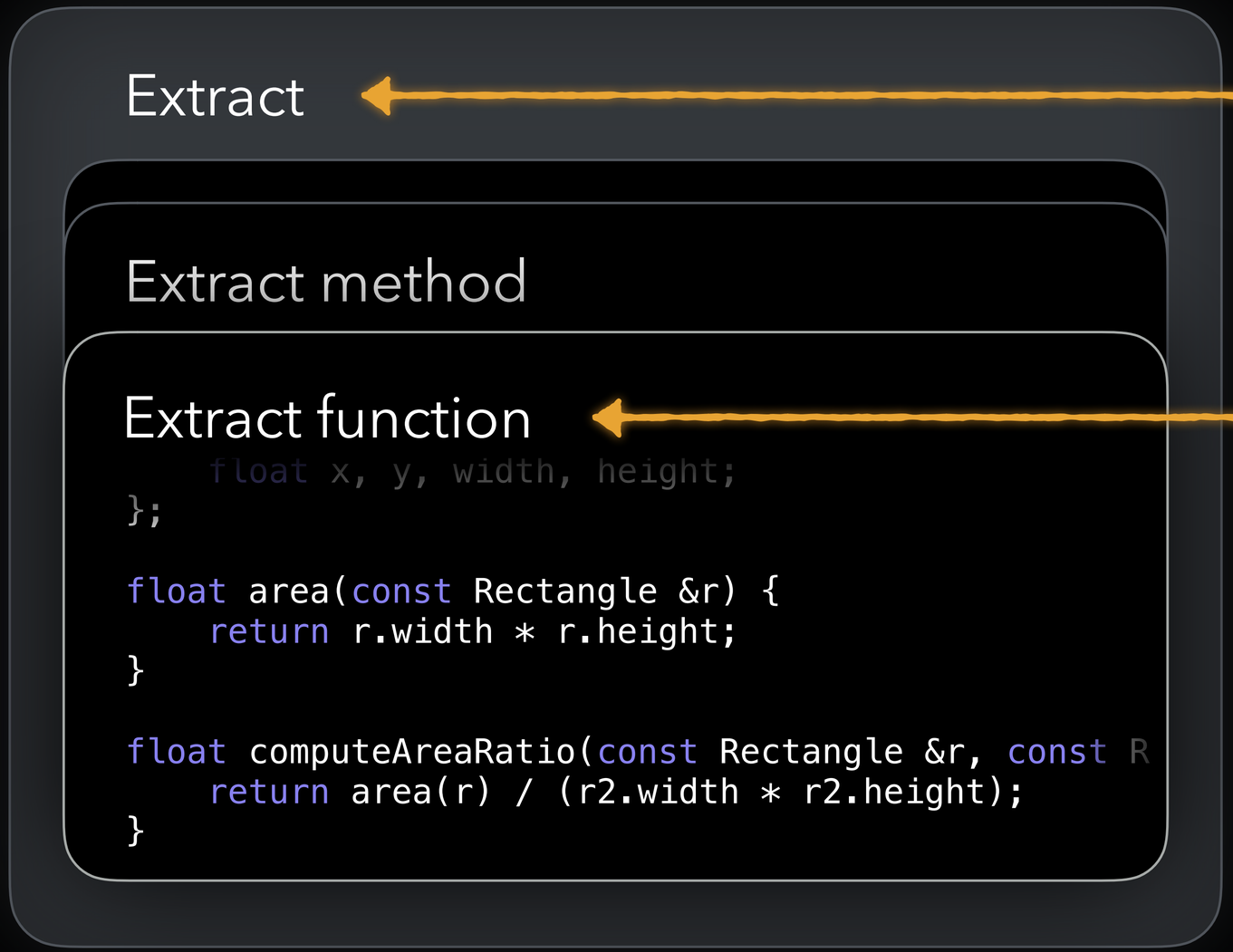
Breakdown of extract




```
class ExtractRefactoringAction
```

```
class ExtractFunction
```

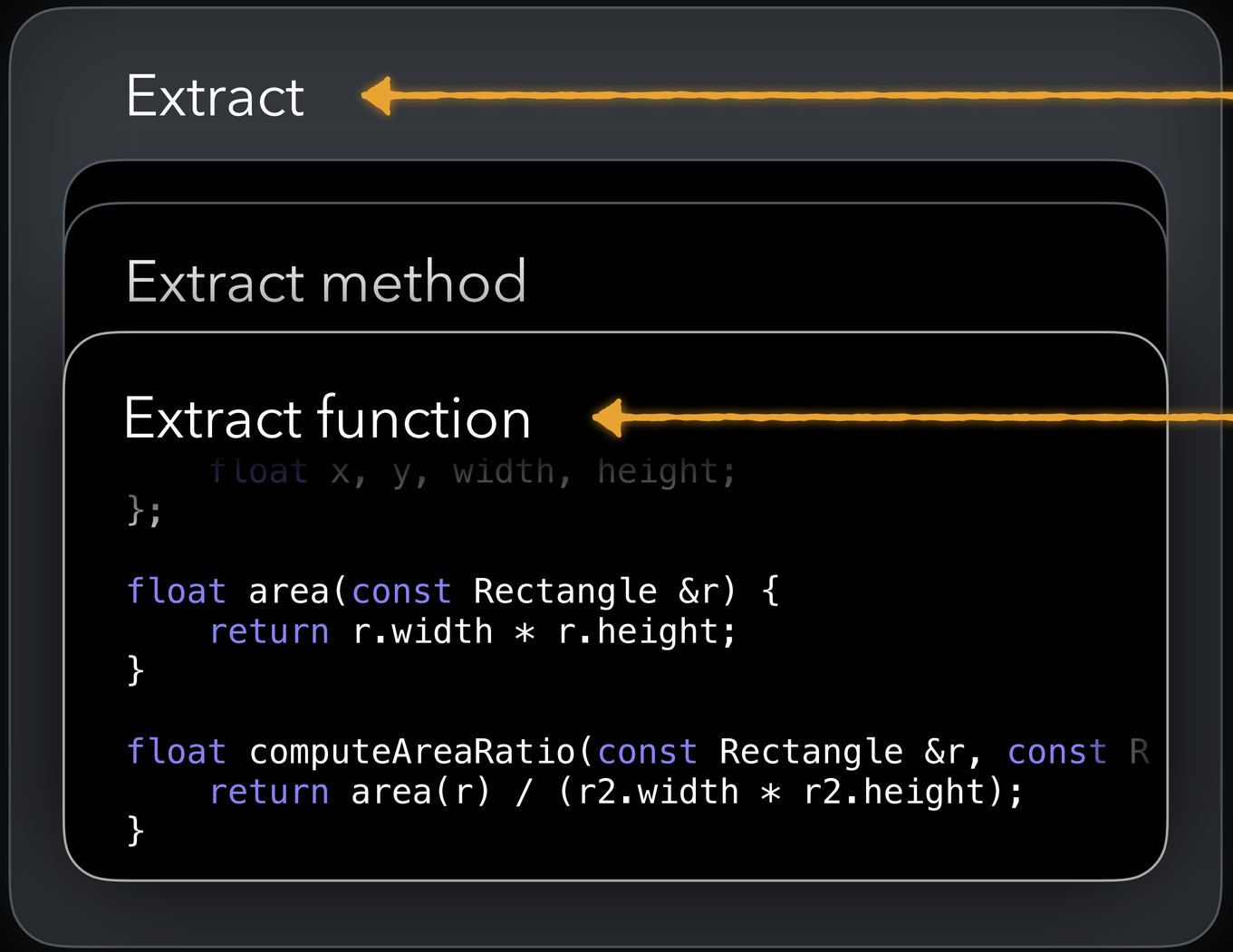
Breakdown of extract




```
class RefactoringAction
class ExtractRefactoringAction
```

```
class RefactoringActionRule
class ExtractFunction
```

Breakdown of extract




```
class RefactoringAction
class ExtractRefactoringAction
```

creates

```
class RefactoringActionRule
class ExtractFunction
```

```
class RefactoringAction
```

```
class RefactoringActionRule
```

Refactoring action *-vs-* refactoring action rule

```
class RefactoringAction
```

```
class RefactoringActionRule
```

Refactoring action *-vs-* refactoring action rule

`class RefactoringAction`

- High-level description of a refactoring

`class RefactoringActionRule`

Refactoring action *-vs-* refactoring action rule

`class RefactoringAction`

- High-level description of a refactoring
- Defines clang-refactor subcommand name

`class RefactoringActionRule`

Refactoring action *-vs-* refactoring action rule

`class RefactoringAction`

- High-level description of a refactoring
- Defines clang-refactor subcommand name

`class RefactoringActionRule`

- Description of a low-level operation

Refactoring action *-vs-* refactoring action rule

`class RefactoringAction`

- High-level description of a refactoring
- Defines clang-refactor subcommand name

`class RefactoringActionRule`

- Description of a low-level operation
- Library manages the initiation stage

Refactoring action *-vs-* refactoring action rule

`class RefactoringAction`

- High-level description of a refactoring
- Defines clang-refactor subcommand name

`class RefactoringActionRule`

- Description of a low-level operation
- Library manages the initiation stage
- Operations like `ExtractFunction` implement the refactoring stage

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
```

1.1. Pick the right base class

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
```

1.1. Pick the right base class

- Extraction is a local source transformation

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
```

1.1. Pick the right base class

- Extraction is a local source transformation

➡ `class SourceChangeRefactoringRule`

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

1.2. Create constructor that receives the required inputs

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- Extraction moves consecutive statements of code

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- Extraction moves consecutive statements of code

  ➡ `class CodeRangeASTSelection`

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- Extraction moves consecutive statements of code

  ➡ `class CodeRangeASTSelection`

- Name of the extracted function will not be provided by the IDE

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- Extraction moves consecutive statements of code
  - ➡ `class CodeRangeASTSelection`
- Name of the extracted function will not be provided by the IDE
  - ➡ `Optional<std::string>`

Initiation stage

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Initiation stage

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- `ExtractFunction` operation constructed by library after initiation

Initiation stage

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- `ExtractFunction` operation constructed by library after initiation
- Inputs provided by operation's initiation requirements

Initiation stage

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- `ExtractFunction` operation constructed by library after initiation
- Inputs provided by operation's initiation requirements

➡ Requirement creates one input value for constructor when satisfied

Initiation stage

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

- `ExtractFunction` operation constructed by library after initiation
- Inputs provided by operation's initiation requirements

➡ Requirement creates one input value for constructor when satisfied

➡ Otherwise refactoring fails or is unavailable

Builtin initiation requirements

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Builtin initiation requirements

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

```
class CodeRangeSelectionRequirement

implements

Expected<CodeRangeASTSelection> evaluate
```

Builtin initiation requirements

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Builtin initiation requirements

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

```
class OptionRequirement<NameOption>
```

implements

```
Expected<Optional<std::string>> evaluate
```

Builtin initiation requirements

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

```
class OptionRequirement<NameOption>
```

requires definition such as

Builtin initiation requirements

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

```
class OptionRequirement<NameOption>
```

requires definition such as

```
class NameOption : public OptionalRefactoringOption<std::string> {
public:
    StringRef getName() const override { return "name"; }
    StringRef getDescription() const override { return "…"; }
};
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
```

1.2. Create constructor that receives the required inputs

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
    createSourceReplacements(RefactoringRuleContext &Context) {
        AtomicChanges Result;
        std::string FnName = Name ? *Name : “extracted”;
        // … Create the new function using selection …
        // … Replace selection with a call to the new function …
        return Result;
    }

private:
```

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
    createSourceReplacements(RefactoringRuleContext &Context) {
        AtomicChanges Result;
        std::string FnName = Name ? *Name : “extracted”;
        // … Create the new function using selection …
        // … Replace selection with a call to the new function …
        return Result;
    }

private:
```

1.3. Implement the refactoring function

Implementing “extract function”

1. Create the operation that performs the refactoring

```
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection,
                    Optional<std::string> Name)
    : Selection(Selection), Name(Name) {}

    Expected<AtomicChanges>
    createSourceReplacements(RefactoringRuleContext &Context) {
        AtomicChanges Result;
        std::string FnName = Name ? *Name : “extracted”;
        // … Create the new function using selection …
        // … Replace selection with a call to the new function …
        return Result;
    }

private:
    CodeRangeASTSelection Selection;
    Optional<std::string> Name;
};
```

Implementing “extract function”

2. Create the refactoring action

```
class ExtractRefactoringAction final : public RefactoringAction {
public:
    StringRef getCommand() const override { return "extract"; }

    RefactoringActionRules createActionRules() const override {
```

Implementing “extract function”

2. Create the refactoring action

```
class ExtractRefactoringAction final : public RefactoringAction {
public:
    StringRef getCommand() const override { return "extract"; }

    RefactoringActionRules createActionRules() const override {
      RefactoringActionRules Rules;
      Rules.push_back(
        createRefactoringActionRule<ExtractFunction>(
          CodeRangeSelectionRequirement(),
          OptionRequirement<NameOption>()));
      return Rules;
    }
};
```

Implementing “extract function”

2. Create the refactoring action

```
class ExtractRefactoringAction final : public RefactoringAction {
public:
    StringRef getCommand() const override { return "extract"; }

    RefactoringActionRules createActionRules() const override {
        RefactoringActionRules Rules;
        Rules.push_back(
          createRefactoringActionRule<ExtractFunction>(
            CodeRangeSelectionRequirement(),
            OptionRequirement<NameOption>()));
        return Rules;
    }
};
```

2.1. Construct the refactoring action rules

Implementing “extract function”

2. Create the refactoring action

```
class ExtractRefactoringAction final : public RefactoringAction {
public:
    StringRef getCommand() const override { return "extract"; }

    RefactoringActionRules createActionRules() const override {
        RefactoringActionRules Rules;
        Rules.push_back(
          createRefactoringActionRule<ExtractFunction>(
            CodeRangeSelectionRequirement(),
            OptionRequirement<NameOption>()));
        return Rules;
    }
};
```

Implementing “extract function”

Implementing “extract function”

3. Add an entry in the refactoring action registry

```
REFACTORING_ACTION(Extract)
```

Implementing “extract function”

3. Add an entry in the refactoring action registry

```
REFACTORING_ACTION(Extract)
```

4. Define the action factory function

```
std::unique_ptr<RefactoringAction> createExtractAction() {
  return llvm::make_unique<ExtractRefactoringAction>();
}
```

Implementing “extract function”

3. Add an entry in the refactoring action registry

```
REFACTORING_ACTION(Extract)
```

4. Define the action factory function

```
std::unique_ptr<RefactoringAction> createExtractAction() {
  return llvm::make_unique<ExtractRefactoringAction>();
}
```

➡ `clang-refactor` supports “extract”

Implementing “extract function”

Implementing “extract function”

IDE Integration

Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry

Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry

```
REFACTORING_EDITOR_COMMAND(ExtractFunction, ”Extract Function”)
```

Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry

```
REFACTORING_EDITOR_COMMAND(ExtractFunction, ”Extract Function”)
```

2. Bind the action-specific rule to the editor command

Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry

```
REFACTORING_EDITOR_COMMAND(ExtractFunction, ”Extract Function”)
```

2. Bind the action-specific rule to the editor command

```
EditorCommand::ExtractFunction().bind(
    createRefactoringActionRule<ExtractFunction>(
```

Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry

```
REFACTORING_EDITOR_COMMAND(ExtractFunction, ”Extract Function”)
```

2. Bind the action-specific rule to the editor command

```
EditorCommand::ExtractFunction().bind(
    createRefactoringActionRule<ExtractFunction>(
       CodeRangeSelectionRequirement(),
       OptionRequirement<NameOption>()))
))
```

Implementing “extract function”

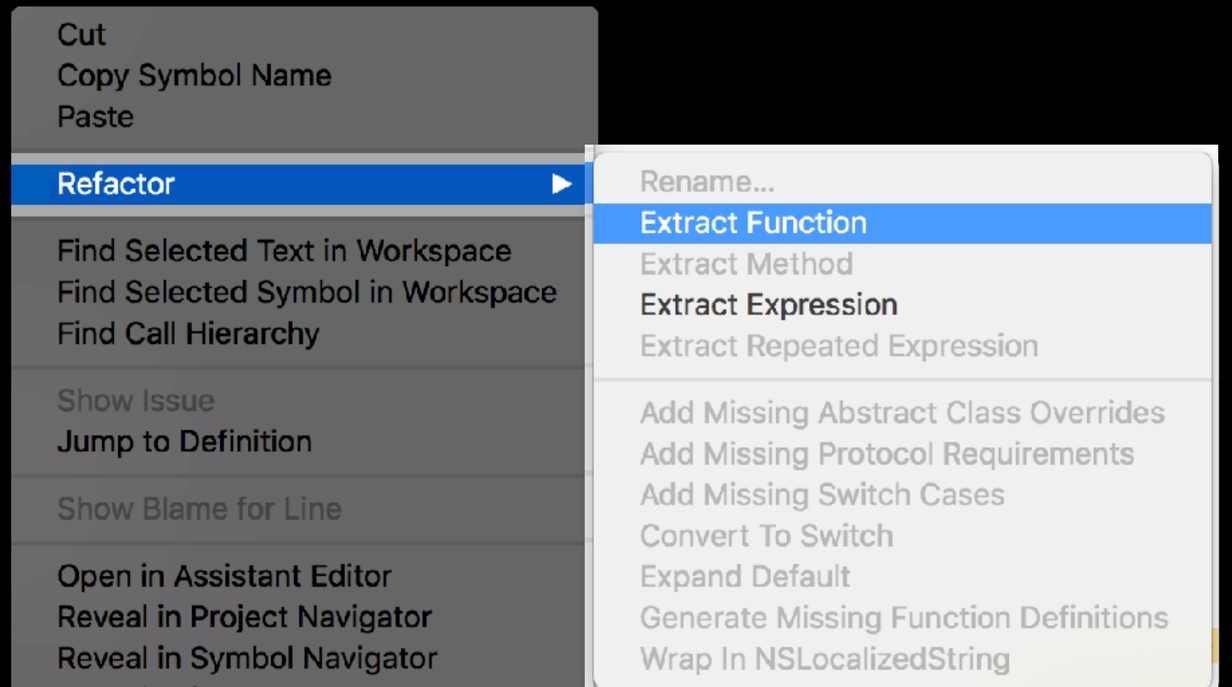
IDE Integration

1. Add an entry in the editor command registry
2. Bind the action-specific rule to the editor command

Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry
2. Bind the action-specific rule to the editor command



Implementing other refactorings

Implementing other refactorings

- Library allows actions to

Implementing other refactorings

- Library allows actions to
  - Subclass builtin requirements for custom initiation logic

Implementing other refactorings

- Library allows actions to
  - Subclass builtin requirements for custom initiation logic
  - Use custom diagnostics to propagate errors to `clang-refactor`

Implementing other refactorings

- Library allows actions to
  - Subclass builtin requirements for custom initiation logic
  - Use custom diagnostics to propagate errors to `clang-refactor`
  - Future components will help with AST matching and cross-TU operations

Implementing other refactorings

- Library allows actions to
  - Subclass builtin requirements for custom initiation logic
  - Use custom diagnostics to propagate errors to `clang-refactor`
  - Future components will help with AST matching and cross-TU operations
- Comprehensive guide available

➡ https://clang.llvm.org/docs/RefactoringEngine.html

Status

Status

Currently upstreaming…

Status

Currently upstreaming…

- Generate Missing Function Definitions
- Rename
- Extract Function
- Extract Variable
- Add Missing Protocol Requirements
- Convert to Switch
- Add Abstract Class Overrides
- Add Missing Switch Cases

Status

Clients

`clangd`
In review; “rename” later

`clang-refactor`
Mostly works

`libclang`
Coming soon…

Status

Status

- Core components committed or in review

Status

- Core components committed or in review
- Migration of clang-rename ongoing

Status

- Core components committed or in review
- Migration of clang-rename ongoing
- Soon: cross-TU actions using refactoring continuations

Status

Available when

Status

Available when

- “extract” undergoing review

Status

Available when

- “extract” undergoing review
- Editor plugins: vim & emacs

Status

Available when

- “extract” undergoing review
- Editor plugins: vim & emacs
- clangd support

Ideas for improvement

Ideas for improvement

- Integration with Clang’s indexing infrastructure

Ideas for improvement

- Integration with Clang’s indexing infrastructure
- Verification of semantic correctness

Ideas for improvement

- Integration with Clang’s indexing infrastructure
- Verification of semantic correctness
- Distributed global refactoring with clang-refactor

Contributing to the new engine

Contributing to the new engine

- Local refactoring tutorial in review: D39027

Contributing to the new engine

- Local refactoring tutorial in review: D39027
- Ideas: bugzilla keyword “beginner” + term “[refactoring]: idea”

Contributing to the new engine

- Local refactoring tutorial in review: D39027
- Ideas: bugzilla keyword “beginner” + term “[refactoring]: idea”
- Migration of clang-reorder-fields and other tools?

Q & A