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

Index Store

IDE

Indexer Service

clang
Indexing while building
Indexing while building

- Index Store
- Indexer Service
- clang
- `-index-store-path`

IDE
Indexing while building

- IDE
- Indexer Service
- clang
- Index Store
Indexing while building
Indexing while building

- Index Store
- Indexer Service
- clang

-ide-index-store-path
- -fsyntax-only
Indexing while building
Indexing while building

Diagram:
- Index Store
- IDE
- Indexer Service
- clang
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%
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

path/to/store

- v5
  - units
    - test.o-3V0GVCF3V9EDC4
  - records
    - 50
      - test.cpp-Z6570C9FQ50
    - 30
      - test.h-20EYSX9EG5E30
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

<table>
<thead>
<tr>
<th>path/to/store</th>
</tr>
</thead>
<tbody>
<tr>
<td>v5</td>
</tr>
<tr>
<td>units</td>
</tr>
<tr>
<td>test.o-3V0GVCB3V6EDC4</td>
</tr>
<tr>
<td>records</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>test.cpp-Z6570CKFQ050</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>test.h-2DEYXS09EGE30</td>
</tr>
</tbody>
</table>

Record
Symbols
Occurrences

| 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

<table>
<thead>
<tr>
<th>path/to/store</th>
</tr>
</thead>
<tbody>
<tr>
<td>v5</td>
</tr>
<tr>
<td>units</td>
</tr>
<tr>
<td>test.o-3V0GVCB3VEDC4</td>
</tr>
<tr>
<td>records</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>test.cpp-Z6570CKFQO50</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>test.h-2DEYSX09EGE30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbols</td>
</tr>
<tr>
<td>Occurrences</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
</tr>
<tr>
<td>IsSystem</td>
</tr>
<tr>
<td>IsModule</td>
</tr>
<tr>
<td>HasMain</td>
</tr>
<tr>
<td>MainPath</td>
</tr>
<tr>
<td>WorkingDir</td>
</tr>
<tr>
<td>OutputFile</td>
</tr>
<tr>
<td>Target</td>
</tr>
<tr>
<td>Dependencies</td>
</tr>
</tbody>
</table>
Generating the index data

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

<table>
<thead>
<tr>
<th>path/to/store</th>
</tr>
</thead>
<tbody>
<tr>
<td>units</td>
</tr>
<tr>
<td>test.o-3V0GVVCB3MEDC4</td>
</tr>
<tr>
<td>records</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>test.cpp-Z6570CKFO50</td>
</tr>
<tr>
<td>30</td>
</tr>
<tr>
<td>test.h-20EY8X09E6G30</td>
</tr>
</tbody>
</table>

- Record
  - Symbols
  - Occurrences

- Unit
- Provider
- IsSystem
- IsModule
- HasMain
- MainPath
- WorkingDir
- OutputFile
- Target
- Dependencies

Hash of symbolic data
#define SOME_MACRO
$ 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
$ clang -o first.o -c first.c -index-store-path path/to/store

#define SOME_MACRO
#ifdef SOME_MACRO

Unit
first.o-2J808UIU...

first.c
things.h
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
#define SOME_MACRO

ifdef SOME_MACRO

first.c

header.h

feature.h

second.c

Record first.c-0QSSP9BN8S73J
Record things.h-P1YD8X09T1GF5
Record header.h-2IRZ9QVC1ZC45
Record feature.h-X8QI5PPQ3008J

Unit first.o-2J808UIU...
```c
#define SOME_MACRO

#ifdef SOME_MACRO
```

```c
#include "things.h"

#include "header.h"

#include "feature.h"

#include "first.c"

#include "second.c"

#include "first.o-2J80UIU...
```

```c
Record
things.h-P1YD8X09T1GF5
```

```c
Record
header.h-2IRZ9QVC1ZC45
```

```c
Record
feature.h-X8Qj5PPQ3008J
```

```c
Unit
first.o-2J80UIU...
```
$ 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
define SOME_MACRO

first.c -> things.h

header.h -> things.h

#ifdef SOME_MACRO

second.c -> feature.h

feature.h -> header.h

DEFINE SOME_MACRO

#define SOME_MACRO

#ifndef SOME_MACRO

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

first.c

things.h

header.h

second.c

feature.h

first.c-0QSSP9BN8S73J

things.h-P1YD8X09T1GF5

header.h-2IRZ9QVC1ZC45

feature.h-X8QI5PPQ3008J

second.c-9RLNQAPAVVB59

Unit

first.o-2J808UIU...

Unit

second.o-1BLA1...
$ 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

- Unit
- Provider
- IsSystem
- IsModule
- HasMain
- MainPath
- WorkingDir
- OutputFile
- Target
- Dependencies

- Record
- Symbols
- Occurrences
Index data model

- Unit
- Provider
- IsSystem
- IsModule
- HasMain
- MainPath
- WorkingDir
- OutputFile
- Target
- Dependencies
- Record
- Symbols
- Occurrences
Index data model

```cpp
enum class SymbolKind : uint8_t {
    Struct,
    Class,
    Union,
    TypeAlias,
    Function,
    Variable,
    Field,
    EnumConstant,
    InstanceMethod,
    Constructor,
    Destructor,
    ...
};
```
Index data model

enum class SymbolProperty : uint8_t {
    Generic                      = 1 << 0,
    TemplatePartialSpecialization = 1 << 1,
    TemplateSpecialization       = 1 << 2,
    UnitTest                     = 1 << 3,
    IBAAnnotated                 = 1 << 4,
    IBOutletCollection           = 1 << 5,
    GKInspectable                = 1 << 6,
    Local                        = 1 << 7,
};
Index data model

enum class SymbolRole : uint32_t {
  Declaration = 1 << 0,
  Definition  = 1 << 1,
  Reference   = 1 << 2,
  Read        = 1 << 3,
  Write       = 1 << 4,
  Call        = 1 << 5,
  Dynamic     = 1 << 6,
  AddressOf   = 1 << 7,
  Implicit    = 1 << 8,

  // Relation roles.
  RelationChildOf     = 1 << 9,
  RelationBaseOf      = 1 << 10,
  RelationOverrideOf  = 1 << 11,
  RelationReceivedBy  = 1 << 12,
  RelationCalledBy    = 1 << 13,
  RelationExtendedBy  = 1 << 14,
  RelationAccessorOf  = 1 << 15,
  RelationContainedBy = 1 << 16,
  RelationIBTypeOf    = 1 << 17,
  RelationSpecializationOf = 1 << 18,
};
Index data model

enum class SymbolRole : uint32_t {
    Declaration = 1 << 0,
    Definition  = 1 << 1,
    Reference   = 1 << 2,
    Read        = 1 << 3,
    Write       = 1 << 4,
    Call        = 1 << 5,
    Dynamic     = 1 << 6,
    AddressOf   = 1 << 7,
    Implicit    = 1 << 8,

    // Relation roles.
    RelationChildOf     = 1 << 9,
    RelationBaseOf      = 1 << 10,
    RelationOverrideOf  = 1 << 11,
    RelationReceivedBy  = 1 << 12,
    RelationCalledBy    = 1 << 13,
    RelationExtendedBy  = 1 << 14,
    RelationAccessorOf  = 1 << 15,
    RelationContainedBy = 1 << 16,
    RelationIBTypeOf    = 1 << 17,
    RelationSpecializationOf = 1 << 18,
};
Index data model

enum class SymbolRole : uint32_t {
  Declaration = 1 << 0,
  Definition  = 1 << 1,
  Reference   = 1 << 2,
  Read        = 1 << 3,
  Write       = 1 << 4,
  Call        = 1 << 5,
  Dynamic     = 1 << 6,
  AddressOf   = 1 << 7,
  Implicit    = 1 << 8,

  // Relation roles.
  RelationChildOf     = 1 << 9,
  RelationBaseOf      = 1 << 10,
  RelationOverrideOf  = 1 << 11,
  RelationReceivedBy  = 1 << 12,
  RelationCalledBy    = 1 << 13,
  RelationExtendedBy  = 1 << 14,
  RelationAccessorOf  = 1 << 15,
  RelationContainedBy = 1 << 16,
  RelationIBTypeOf    = 1 << 17,
  RelationSpecializationOf = 1 << 18,
};
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;
    }
};
```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;
        }
};
```
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;
    }
  
};
```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;
    }
  }
```
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; 
    }
  };
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;
        }
};
Producing index data
Producing index data
Consuming index data
Consuming index data

- Index Store
- Indexer Service
- IDE
- clang
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

• 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.
```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;
        }
};
```
Find subclasses of Polygon
Find subclasses of Polygon

```
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;
    }
};
```
Find subclasses of Polygon

Symbol 1

**Polygon**

USR c:@S@Polygon
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

Symbol 2

**RegularPolygon**

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

Find occurrences with the RelationBaseOf role.

```
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;
    }
};
```
Find subclasses of Polygon
Find subclasses of Polygon

<table>
<thead>
<tr>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
<th>Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
<td>Record</td>
</tr>
</tbody>
</table>
Speeding things up in the indexer service
Speeding things up in the indexer service
Find subclasses of Polygon
Find subclasses of Polygon

USR

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

1. Lookup USR.
Find subclasses of Polygon

1. Lookup USR.
2. Find records with the RelationBaseOf role.
## Find subclasses of Polygon

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

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

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

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

<table>
<thead>
<tr>
<th>USR</th>
<th>Record + Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>c:@S@Polygon</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>test.cpp-2RLZQAEAVK8SU</td>
</tr>
<tr>
<td></td>
<td>Definition, Reference, RelationBaseOf</td>
</tr>
<tr>
<td></td>
<td>other.cpp-X8QI5PPQ303AO</td>
</tr>
<tr>
<td></td>
<td>Reference, RelationContainedBy</td>
</tr>
<tr>
<td>c:@S@RegularPolygon</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>test.cpp-2RLZQAEAVK8SU</td>
</tr>
<tr>
<td></td>
<td>Definition</td>
</tr>
<tr>
<td></td>
<td>chess.h-2RLZQAEAVK8SU</td>
</tr>
<tr>
<td></td>
<td>Reference, RelationBaseOf</td>
</tr>
</tbody>
</table>
### Find definitions of RegularPolygon

<table>
<thead>
<tr>
<th>USR</th>
<th>Record + Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>c:@S@Polygon</td>
<td></td>
</tr>
<tr>
<td>c:@S@RegularPolygon</td>
<td></td>
</tr>
<tr>
<td>test.cpp-2RLZQAEAVK8SU</td>
<td>Definition, Reference, RelationBaseOf</td>
</tr>
<tr>
<td>other.cpp-X8QI5PPQ303AO</td>
<td>Reference, RelationContainedBy</td>
</tr>
<tr>
<td>test.cpp-2RLZQAEAVK8SU</td>
<td>Definition</td>
</tr>
<tr>
<td>chess.h-2RLZQAEAVK8SU</td>
<td>Reference, RelationBaseOf</td>
</tr>
</tbody>
</table>

1. Lookup USR.
Find definitions of RegularPolygon

1. Lookup USR.
2. Find records with the Definition role.
Find definitions of RegularPolygon

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

1. Lookup USR.
2. Find records with the ________ role.
3. Retrieve those occurrences.
Find ________ of RegularPolygon

1. Lookup USR.
2. Find records with the ________ role.
3. Retrieve those occurrences of ________.
Other useful mappings
## Other useful mappings

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>USRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>c:@S@Shape, c:@T@Sh...</td>
</tr>
</tbody>
</table>

Search symbols by name
### Other useful mappings

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>USRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>c:@S@Shape, c:@T@Sh...</td>
</tr>
</tbody>
</table>

#### Search symbols by name

<table>
<thead>
<tr>
<th>Source file</th>
<th>Dependent units</th>
</tr>
</thead>
<tbody>
<tr>
<td>things.h</td>
<td>first.o-XXX, second.o-XXX</td>
</tr>
</tbody>
</table>

#### Units to re-index when header changes
Other useful mappings

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>USRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>c:@S@Shape, c:@T@Sh...</td>
</tr>
</tbody>
</table>

Search symbols by name

<table>
<thead>
<tr>
<th>Source file</th>
<th>Dependent units</th>
</tr>
</thead>
<tbody>
<tr>
<td>things.h</td>
<td>first.o-XXX, second.o-XXX</td>
</tr>
</tbody>
</table>

Units to re-index when header changes

<table>
<thead>
<tr>
<th>Record</th>
<th>Dependent units</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature.h-XXX</td>
<td>first.o-XXX</td>
</tr>
</tbody>
</table>

Remove data from unreferenced records
Indexing in the IDE

Index Store

Indexer Service

clang

IDE
General indexing infrastructure
General indexing infrastructure

Diagram showing the relationships between:
- **Index Store**
- **Core Indexer Service**
- **clang**
- **Vim**
- **Emacs**
- **Sublime**

Arrows indicate the flow of information or integration between these components.
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

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

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

```cpp
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

```c
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

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

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

```c
r.width * r.height
```
Refactoring
Let’s start with an example

```cpp
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

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

float computeAreaRatio(const Rectangle &r, const Rectangle &r2) {
    return (r.width * r.height) / (r2.width * r2.height);
};
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
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”
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);
}
Breakdown of extract

Extract

Extract method

Extract function

```c++
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);
}
```
Breakdown of extract

Extract method

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

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

Extract function

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

Extract method

Extract function

```c++
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);
}
```
```cpp
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);
}
```
Breakdown of extract

**Extract method**

```c++
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);
}
```

**Extract function**

- `class RefactoringAction`
- `class ExtractRefactoringAction`
- `class RefactoringActionRule`
- `class ExtractFunction`
Breakdown of extract

Extract method

Extract function

```c++
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);
}
```
class RefactoringAction

class RefactoringActionRule
Refactoring action -vs- refactoring action rule

```python
class RefactoringAction

class RefactoringActionRule
```
Refactoring action -vs- refactoring action rule

```java
class RefactoringAction

• High-level description of a refactoring
```

```java
class RefactoringActionRule
```
Refactoring action -vs- refactoring action rule

```python
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

```cpp
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection, 
                    Optional<std::string> Name)
```
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection, Optional<std::string> Name)
```
Implementing “extract function”

1. Create the operation that performs the refactoring

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

1.1. Pick the right base class
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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;
};
```

1.1. Pick the right base class

• Extraction is a local source transformation
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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;
};
```

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

```cpp
class ExtractFunction final : public SourceChangeRefactoringRule {
public:
    ExtractFunction(CodeRangeASTSelection Selection, Optional<std::string> Name)
```
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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”

1. Create the operation that performs the refactoring

```cpp
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”

1. Create the operation that performs the refactoring

```cpp
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;
};
```

1.2. Create constructor that receives the required inputs
Implementing “extract function”

1. Create the operation that performs the refactoring

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

eXpected<AtomicChanges>
screateSourceReplacements(RefactoringRuleContext &Context) { return
```
Implementing “extract function”

1. Create the operation that performs the refactoring

```java
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;
};
```

• Extraction moves consecutive statements of code
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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;
};
```

• Extraction moves consecutive statements of code

⇒ `class CodeRangeASTSelection`
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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;
};
```

- 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

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

    Expected<AtomicChanges> createSourceReplacements()
    { Result;
        std::string FnName = Name ? *Name : "extracted";

        // Create the new function using selection
        // ... Replacement selection with a call to the new function ...

        return Result;
    }

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

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

Initiation stage
Initiation stage

- ExtractFunction operation constructed by library after initiation
Initiation stage

- ExtractFunction operation constructed by library after initiation
- Inputs provided by operation’s initiation requirements
Initiation stage

- ExtractFunction operation constructed by library after initiation
- Inputs provided by operation’s initiation requirements
  - Requirement creates one input value for constructor when satisfied
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;
};

Initiation stage

- 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
Built-in initiation requirements

```cpp
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;
};
```
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;
};

class CodeRangeSelectionRequirement implements 
Expected<CodeRangeASTSelection> evaluate
Builtin initiation requirements

```cpp
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;
};
```
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;
};
Built-in initiation requirements

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;
};

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>
    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;
};

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 "…"; }
};
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”

1. Create the operation that performs the refactoring
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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;
};
```

1.2. Create constructor that receives the required inputs
Implementing “extract function”

1. Create the operation that performs the refactoring

```cpp
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”

1. Create the operation that performs the refactoring

```cpp
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

```cpp
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

```cpp
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

```cpp
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

```cpp
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

```cpp
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

```cpp
REFACTORING_ACTION(Extract)
```

4. Define the action factory function

```cpp
std::unique_ptr<RefactoringAction> createExtractAction() {
    return llvm::make_unique<ExtractRefactoringAction>();
}
```
Implementing “extract function”

3. Add an entry in the refactoring action registry

```cpp
REFACTORING_ACTION(Extract)
```

4. Define the action factory function

```cpp
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

```python
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

```cpp
REFACTORYING_EDITOR_COMMAND(ExtractFunction, "Extract Function")
```

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

```cpp
EditorCommand::ExtractFunction().bind(
    createRefactoringActionRule<ExtractFunction>(
```
Implementing “extract function”

IDE Integration

1. Add an entry in the editor command registry

   ```cpp
   REFACTORING_EDITOR_COMMAND(ExtractFunction, "Extract Function")
   ```

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

   ```cpp
   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
 Currently upstreaming…
Status
Currently upstreaming...

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

Clients

- clingd
  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