Code Completion in ClangRepl

—

Student: Yuquan (Fred) Fu
Mentor: Vassil Vassilev

Problem Recap

Avoid Tedious & Erroneous Typing

```
clang-repl> struct WhateverMeaningfulLoooooooooongName{ int field;};
clang-repl> Wh<tab>
```

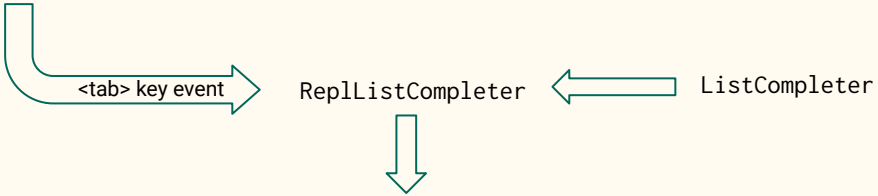
Semantic Code Completion

```
clang-repl> struct Apple{ int price;};
clang-repl> struct Banana{ int StoreID;};
clang-repl> void getApple(Apple &a) {};

clang-repl> Apple fruitIsApple(10);
clang-repl> Banana fruitIsBanana(42);
clang-repl> getApple(f<tab>
```

Implementations

```
clang-repl> struct WhateverMeaningfulLoooooooooongName{ int field;};
clang-repl> Wh<tab>
```




```
std::vector<llvm::LineEditor::Completion>
operator()(llvm::StringRef Buffer, size_t Pos) const;
```

"Wh" 2

LineEditor

How regular code completion works?

Ex: `clang++ -cc1 -fsyntax-only -code-completion-at=hello1.cpp:1:2 hello1.cpp:1:2`

```
CI = new CompilerInvocation()
   this->FrondendOpts.CodeCompletionAt.File = “hello_world.cpp”
   this->FrondendOpts.CodeCompletionAt.Col = 2
   this->FrondendOpts.CodeCompletionAt.Line = 1
   setUpTheRest()

Clang = new CompilerInstance(CI)
Act = new SyntaxOnlyAction()
Act->BeginSourceFile(Clang)
Act->Execute()
   Clang->createCodeCompletionConsumer()
          EnableCodeCompletion(Clang->FrondendOpts.CodeCompletionAt)
   Parsing()
      TriggerCodeCompletion
        DefaultCodeCompletionConsumer::ProcessCodeCompleteResults(CompletionResults)
```

Solution?

Copy & Paste w/ Modifications

Step 1

```
CI = new CompilerInvocation()
   this->FrondendOpts.CodeCompletionAt.File = “<<<input>>>”
   this->FrondendOpts.CodeCompletionAt.Col = Pos + 1
   this->FrondendOpts.CodeCompletionAt.Line = 1
   setUpTheRest()

Clang = new IncrementalCompilerInstance(CI)
Act = new IncrementalSyntaxOnlyAction()
Act->BeginSourceFile(Clang)
Act->Execute()
   Clang->setCodeCompletionConsumer(new ReplCodeCompletionConsumer)
   Super(SyntaxOnlyAction, Execute)
      Parsing()
      TriggerCodeCompletion
         ReplCodeCompletionConsumer::ProcessCodeCompleteResults(CompletionResults)
```

Step 2:

Do step 1 for each code completion

Proposed Better Solution for Step 1

```
AU = ASTUnit::LoadFromCompilerInvocationAction(new CompilerInvocation)
Act = new IncrementalSyntaxOnlyAction()
AU->codeComplete("<<<input>>>", 1, Pos + 1, new ReplCodeCompletionConsumer, Act)
```

Two Problems

Visibility of Defined Declarations

```
clang-repl> struct WhateverMeaningfulLoooooooooongName{ int field;};
clang-repl> <cursor>
```

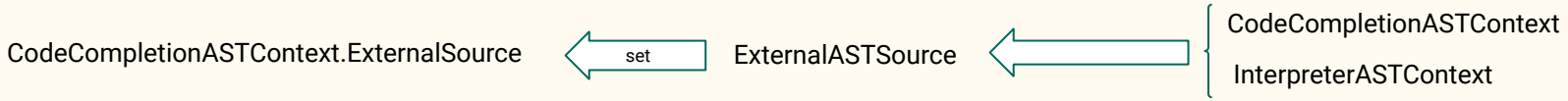
WHERE is the struct declaration I just defined????

1. Two compiler instances: CodeCompletionCI, InterpreterCI
2. Sema/CodeComplete* assume the input is a complete code (AST) context.

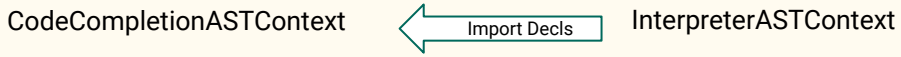
How can we do CodeCompletionCI.ASTContext ++ InterpreterCI.ASTContext??

Bridge Two ASTContexts

Step 1 – When IncrementalSyntaxOnlyAction::ExecuteAction starts



Step 2 – When `Sema/CodeComplete*` run, `ExternalASTSource::completeVisibleDeclsMap` is triggered



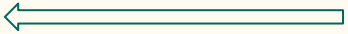
New Completion Contexts

```
clang-repl> int foo = 42;
clang-repl> 1 + f<tab>
```

```
int foo = 42;
1 + f<tab>
```

CodeCompletionContext:

CCC_TopLevelOrExpression



CCC_TopLevel

No Expression Statements

Yes Expression Statements

Complete Solution

Step 1:

```
AU = ASTUnit::LoadFromCompilerInvocationAction(new CompilerInvocation)
Act = new IncrementalSyntaxOnlyAction(InterpreterCI)
AU->codeComplete("<<<input>>>", 1, Pos + 1, new ReplCodeCompletionConsumer, Act)

    Act->BeginSourceFile(CodeCompletionCI)
    Act->Execute()
       CodeCompletionCI.ASTContext.ExternalContext = InterpreterCI.ASTContext
       Super(SyntaxOnlyAction, Execute)
         Parsing()
           If token::code_complete is at a top level and in a repl session
              triggerCodeCompletion(CCC_TopLevelOrExpression)
```

Step 2:

Do step 1 whenever code completion is triggered

Demo

```
➤ ./bin/clang-repl
clang-repl> struct WhateverNameYouChoseForMe{};
clang-repl>
```

Demo

```
➜ ./bin/clang-repl
clang-repl> int application = 42;
clang-repl> int apple = 84;
clang-repl>
```

Semantic Code Completion

Overview

1. Still work-in-progress

2. Improvement on `ReplCodeCompletionConsumer::ProcessCodeCompleteResults`

Worked Example

```
clang-repl> struct Apple{ int price;};
clang-repl> struct Banana{ int StoreID;};
clang-repl> void getApple(Apple &a) {};

clang-repl> Apple fruitIsApple(10);
clang-repl> Banana fruitIsBanana(42);
clang-repl> getApple(f<tab>
```

`ProcessCodeCompleteResults`

1. Find out the relevant function declaration.
2. Get the required parameter type of the function w.r.t the completion point. E.g. `FunDecl->getParamDecl(0)->getType()`
3. If a completion result is a declaration, compare its type with parameter type

Demo

```
capfredf@tumbleweed⚙️:~/c/l/build|sema-comp-attempt1
➤ ./bin/clang-repl
clang-repl> int fooo = 42;
clang-repl> char fuuuuuu = 'a';
clang-repl> void incInt(int &a){};
clang-repl> incInt(f
```

Q & A