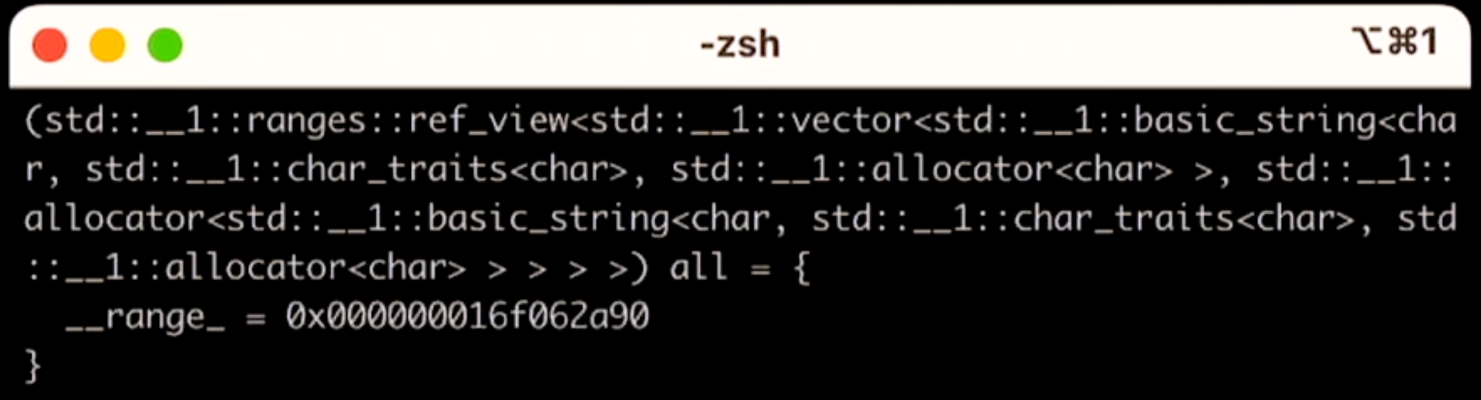
From std::ranges to simpler template names: A C++ debugging journey

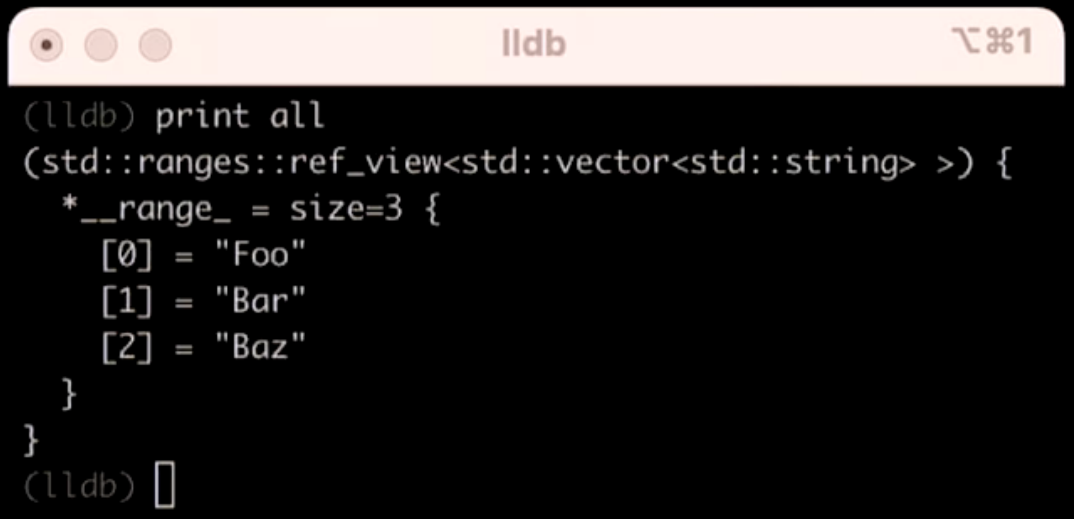
Michael Buch

EuroLLVM 2024 | Apple | 11th April 2024

LLDB Integration



```
(std::__1::ranges::ref_view<std::__1::vector<std::__1::basic_string<cha
r, std::__1::char_traits<char>, std::__1::allocator<char> >, std::__1::
allocator<std::__1::basic_string<char, std::__1::char_traits<char>, std
::__1::allocator<char> > > > >) all = {
  __range_ = 0x000000016f062a90
}
```



```
(lldb) print all
(std::ranges::ref_view<std::vector<std::string> >) {
  *__range_ = size=3 {
    [0] = "Foo"
    [1] = "Bar"
    [2] = "Baz"
  }
}
(lldb)
```

2023 LLVM DEVELOPERS' MEETING

Design and Implementation of C++20 Ranges in libc++

Konstantin Varlamov

https://youtu.be/g9p1oo8bDJA?t=1119

```cpp
std::map<int, std::vector<std::string>> map{
  {1, {“foo"}},
  {2, {"bar"}}};

auto elems = std::views::values(map);
```

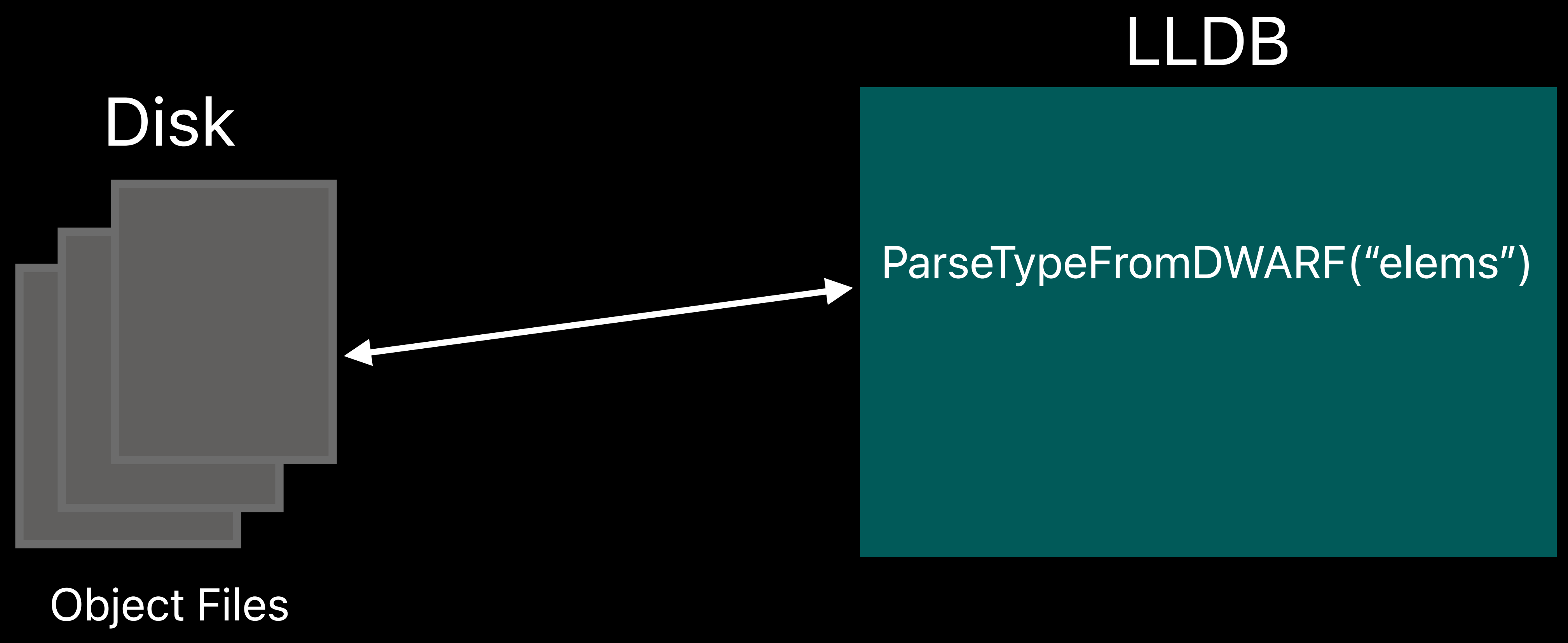
```
(lldb) p elems
```

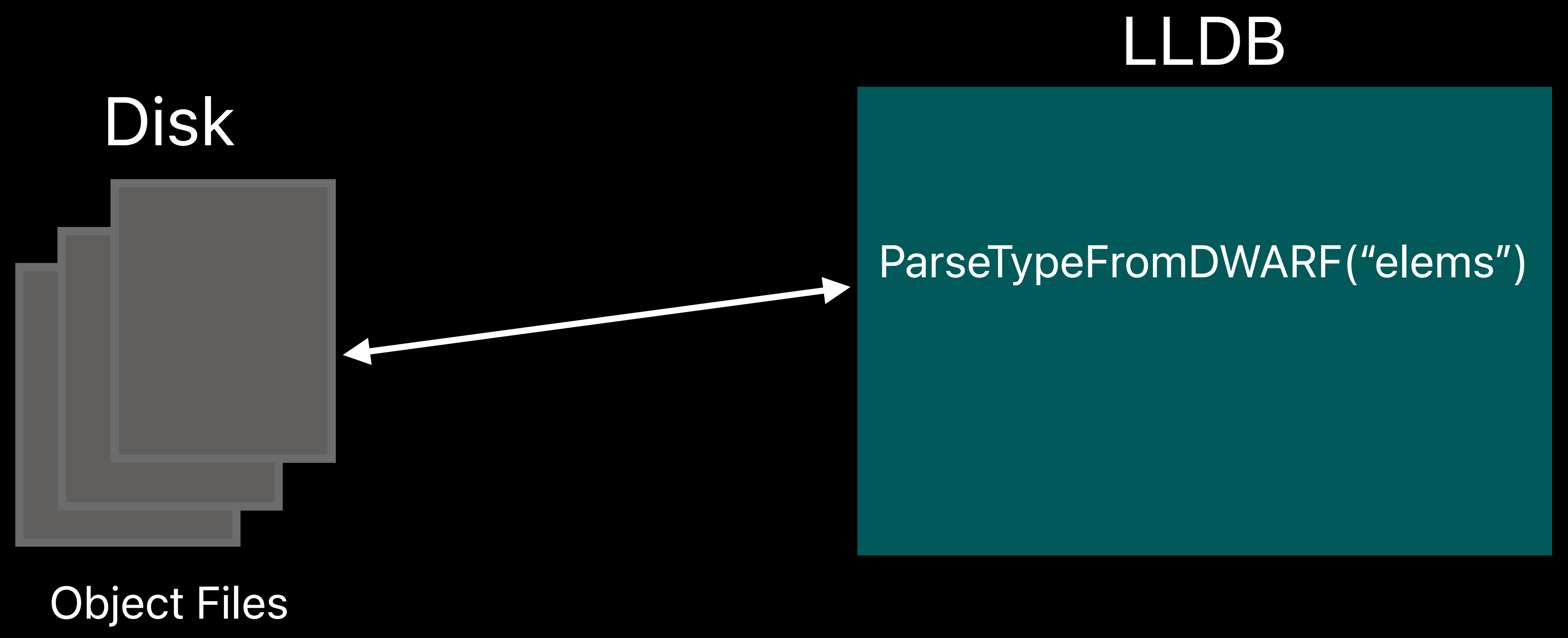
```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::basic_string<char, std::char_traits<char>,
std::allocator<char> >, std::allocator<std::basic_string<char,
std::char_traits<char>, std::allocator<char> > > >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::basic_string<char,
std::char_traits<char>, std::allocator<char> >,
std::allocator<std::basic_string<char, std::char_traits<char>,
std::allocator<char> > > > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

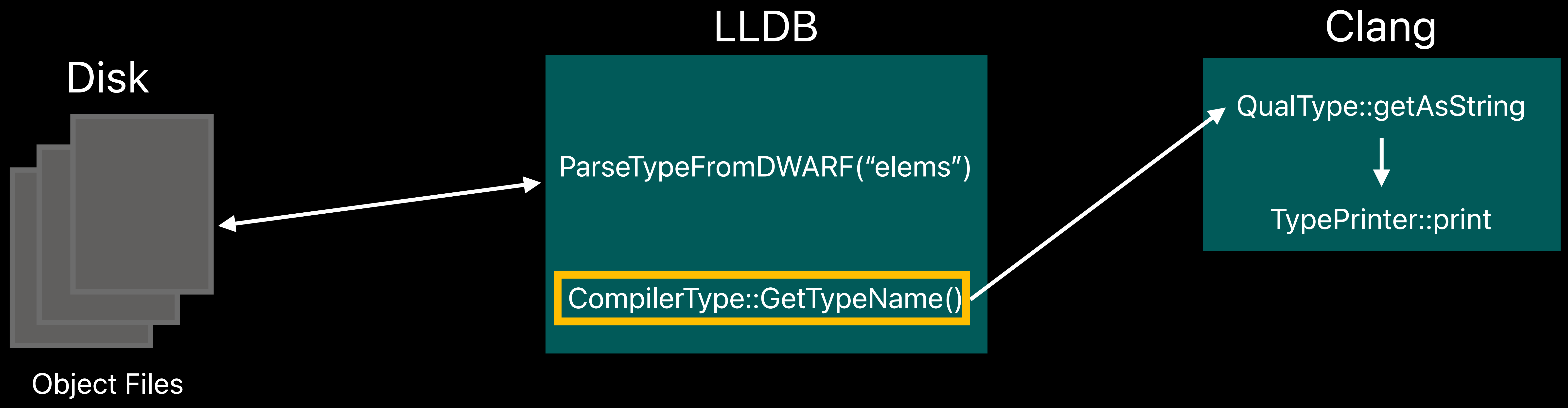
```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string> > >, 1>) elems = {
  __base_ = {
    *__range_ = size=2 {
      [0] = {
        first = 1
        second = size=1 {
          [0] = "foo"
        }
      }
      [1] = {
        first = 2
        second = size=1 {
          [0] = "bar"
        }
      }
    }
  }
}
```

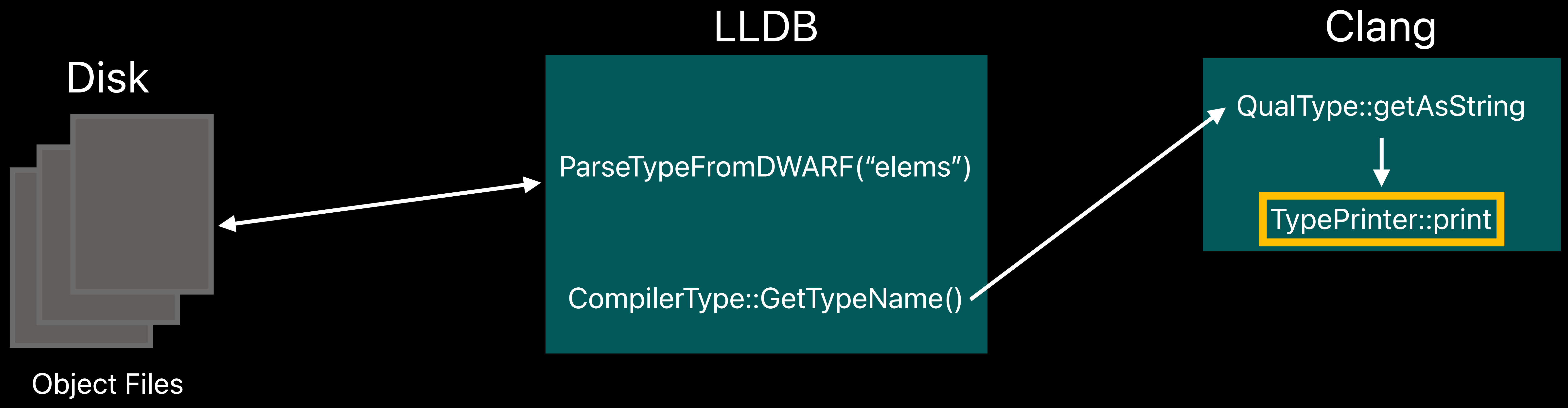

LLDB
Disk
ParseTypeFromDWARF("elems")
Object Files



```
ClassTemplateSpecializationDecl 0x14acc01d8 class map definition
|-TemplateArgument type 'int'
| `-BuiltinType 0x14ac64910 'int'
|-TemplateArgument type 'std::vector<std::string>'
| `-RecordType 0x14acbef10 'std::vector<std::string>'
|   `-ClassTemplateSpecialization 0x14acbee08 'vector'
|- …
```



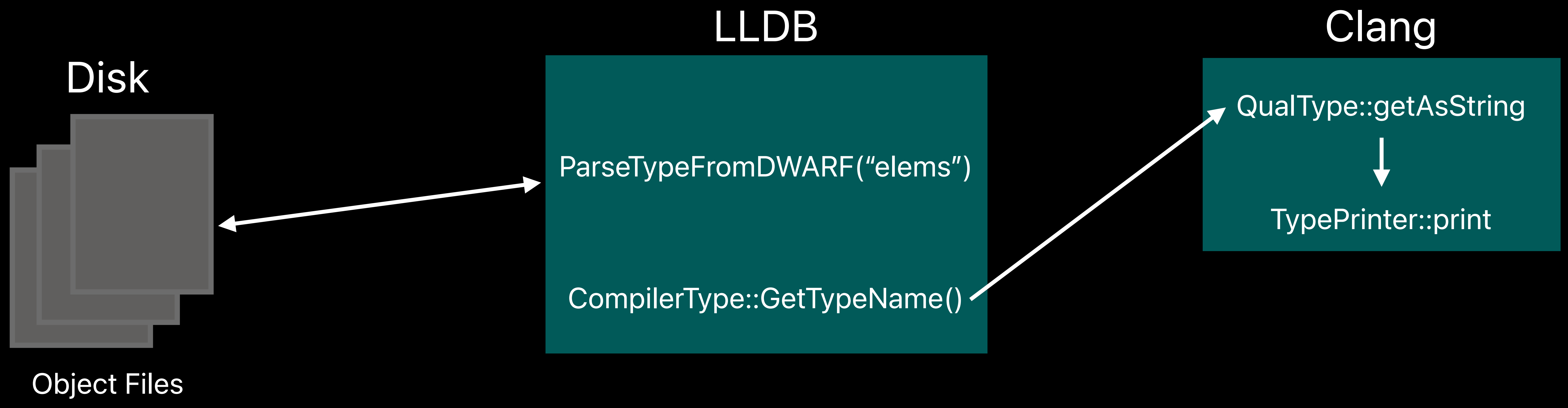
```
ClassTemplateSpecializationDecl 0x14acc01d8 class map definition
|-TemplateArgument type 'int'
| `-BuiltinType 0x14ac64910 'int'
|-TemplateArgument type 'std::vector<std::string>'
| `-RecordType 0x14acbef10 'std::vector<std::string>'
|   `-ClassTemplateSpecialization 0x14acbee08 'vector'
|- …
```



```
ClassTemplateSpecializationDecl 0x14acc01d8 class map definition
|-TemplateArgument type 'int'
| `-BuiltinType 0x14ac64910 'int'
|-TemplateArgument type 'std::vector<std::string>'
| `-RecordType 0x14acbef10 'std::vector<std::string>'
|   `-ClassTemplateSpecialization 0x14acbee08 'vector'
|- …
```

LLVM DEV MEETING
Raphael Isemann
Better C++ Debugging using Clang Modules in LLDB
LLVM.ORG
Expression Evaluation in LLDB

https://www.youtube.com/watch?v=vuNZLlHhy0k



```
ClassTemplateSpecializationDecl 0x14acc01d8 class map definition
|-TemplateArgument type 'int'
| `-BuiltinType 0x14ac64910 'int'
|-TemplateArgument type 'std::vector<std::string>'
| `-RecordType 0x14acbef10 'std::vector<std::string>'
|   `-ClassTemplateSpecialization 0x14acbee08 'vector'
|- …
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::basic_string<char, std::char_traits<char>,
std::allocator<char> >, std::allocator<std::basic_string<char,
std::char_traits<char>, std::allocator<char> > > >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::basic_string<char,
std::char_traits<char>, std::allocator<char> >,
std::allocator<std::basic_string<char, std::char_traits<char>,
std::allocator<char> > > > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::basic_string<char, std::char_traits<char>,
std::allocator<char> >, std::allocator<std::basic_string<char,
std::char_traits<char>, std::allocator<char> > > >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::basic_string<char,
std::char_traits<char>, std::allocator<char> >,
std::allocator<std::basic_string<char, std::char_traits<char>,
std::allocator<char> > > > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
$ clang++ -g main.cpp -fsyntax-only -Xclang -ast-dump
$ clang++ -g main.cpp -fsyntax-only -Xclang -ast-dump

ClassTemplateSpecializationDecl 0x1202f8250 class basic_string definition
|- …
|-TemplateArgument type 'char'
| `-BuiltinType 0x12804aeb0 'char'
|-TemplateArgument type 'std::char_traits<char>'
| `-RecordType 0x1202eb050 'std::char_traits<char>'
|   `-ClassTemplateSpecialization 0x11993ffc0 'char_traits'
|-TemplateArgument type 'std::allocator<char>'
| `-RecordType 0x1202f81a0 'std::allocator<char>'
|   `-ClassTemplateSpecialization 0x1202f80c0 'allocator'
|-OwnerAttr 0x119f424c0 <<invalid sloc>> Inherited Implicit
|-PreferredNameAttr 0x119ea2c98 string
```

```
$ clang++ -g main.cpp -fsyntax-only -Xclang -ast-dump

ClassTemplateSpecializationDecl 0x1202f8250 class basic_string definition
|- …
|-TemplateArgument type 'char'
| `-BuiltinType 0x12804aeb0 'char'
|-TemplateArgument type 'std::char_traits<char>'
| `-RecordType 0x1202eb050 'std::char_traits<char>'
|   `-ClassTemplateSpecialization 0x11993ffc0 'char_traits'
|-TemplateArgument type 'std::allocator<char>'
| `-RecordType 0x1202f81a0 'std::allocator<char>'
|   `-ClassTemplateSpecialization 0x1202f80c0 'allocator'
|-OwnerAttr 0x119f424c0 <<invalid sloc>> Inherited Implicit
|-PreferredNameAttr 0x119ea2c98 string
```

```
void TypePrinter::printRecordBefore(const RecordType *T, raw_ostream &OS) {
  // Print the preferred name if we have one for this type.
  if (Policy.UsePreferredNames) {
    for (const auto *PNA : T->getDecl()->specific_attrs<PreferredNameAttr>()) {
      if (!declaresSameEntity(PNA->getTypedefType()->getAsCXXRecordDecl(),
                              T->getDecl()))
        continue;
```

(from clang/lib/AST/TypePrinter.cpp)

```
void TypePrinter::printRecordBefore(const RecordType *T, raw_ostream &OS) {
  // Print the preferred name if we have one for this type.
  if (Policy.UsePreferredNames) {
    for (const auto *PNA : T->getDecl()->specific_attrs<PreferredNameAttr>()) {
      if (!declaresSameEntity(PNA->getTypedefType()->getAsCXXRecordDecl(),
                              T->getDecl()))
        continue;
```

(from clang/lib/AST/TypePrinter.cpp)

```
(lldb) target modules dump ast
```

```
(lldb) target modules dump ast

ClassTemplateSpecializationDecl 0x107a34dc8 class basic_string definition
|- …
|-TemplateArgument type 'char'
| `-BuiltinType 0x11c27cab0 'char'
|-TemplateArgument type 'std::char_traits<char>'
| `-RecordType 0x107a303c0 'std::char_traits<char>'
|   `-ClassTemplateSpecialization 0x107a46a50 'char_traits'
|-TemplateArgument type 'std::allocator<char>'
| `-RecordType 0x107a30850 'std::allocator<char>'
|   `-ClassTemplateSpecialization 0x107a35590 'allocator'
```

```
(lldb) target modules dump ast

ClassTemplateSpecializationDecl 0x107a34dc8 class basic_string definition
|- …
|-TemplateArgument type 'char'
| `-BuiltinType 0x11c27cab0 'char'
|-TemplateArgument type 'std::char_traits<char>'
| `-RecordType 0x107a303c0 'std::char_traits<char>'
|   `-ClassTemplateSpecialization 0x107a46a50 'char_traits'
|-TemplateArgument type 'std::allocator<char>'
| `-RecordType 0x107a30850 'std::allocator<char>'
|   `-ClassTemplateSpecialization 0x107a35590 'allocator'
```

PreferredNameAttr???

```
$ dwarfdump a.out.dSYM
$ dwarfdump a.out.dSYM

0x0123:  DW_TAG_class_type
           DW_AT_name  (“map<…>")
           DW_TAG_template_type_parameter
             DW_AT_type        (0x8332 "int")
             DW_AT_name        ("_Key")
           DW_TAG_template_type_parameter
             DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
             DW_AT_name        (“_Tp")

0x0214:  DW_TAG_class_type
           DW_AT_name  (“std::vector<std::__1::basic_string<…”)
           DW_TAG_template_type_parameter
             DW_AT_type        (0x0dec "std::__1::basic_string<…")
             DW_AT_name        (“_Tp”)

0x0dec:  DW_TAG_class_type
           DW_AT_name  (“std::__1::basic_string<…”)
```

```
$ dwarfdump a.out.dSYM
                                          std::map
0x0123:  DW_TAG_class_type
           DW_AT_name  (“map<…>")
           DW_TAG_template_type_parameter
             DW_AT_type          (0x8332 "int")
             DW_AT_name          ("_Key")
           DW_TAG_template_type_parameter
             DW_AT_type          (0x0214 "std::__1::vector<std::__1::basic_string<…")
             DW_AT_name          (“_Tp")

0x0214:  DW_TAG_class_type                                             std::vector
           DW_AT_name  (“std::vector<std::__1::basic_string<…”)
           DW_TAG_template_type_parameter
             DW_AT_type          (0x0dec "std::__1::basic_string<…")
             DW_AT_name          (“_Tp”)

0x0dec:  DW_TAG_class_type
           DW_AT_name  (“std::__1::basic_string<…”)
                                                                       std::basic_string
```

```
$ dwarfdump a.out.dSYM

0x0123:  DW_TAG_class_type
           DW_AT_name  (“map<…>")
           DW_TAG_template_type_parameter
             DW_AT_type         (0x8332 "int")
             DW_AT_name         ("_Key")
           DW_TAG_template_type_parameter
             DW_AT_type         (0x0214 "std::__1::vector<std::__1::basic_string<…")
             DW_AT_name         (“_Tp")

0x0214:  DW_TAG_class_type
           DW_AT_name  (“std::vector<std::__1::basic_string<…”)
           DW_TAG_template_type_parameter
             DW_AT_type         (0x0dec "std::__1::basic_string<…")
             DW_AT_name         (“_Tp”)

0x0dec:  DW_TAG_class_type
           DW_AT_name  (“std::__1::basic_string<…”)
```

```
$ dwarfdump a.out.dSYM

0x0123:  DW_TAG_class_type
           DW_AT_name  (“map<…>")
           DW_TAG_template_type_parameter
             DW_AT_type         (0x8332 "int")
             DW_AT_name         ("_Key")
           DW_TAG_template_type_parameter
             DW_AT_type         (0x0214 "std::__1::vector<std::__1::basic_string<…")
             DW_AT_name         (“_Tp")

0x0214:  DW_TAG_class_type
           DW_AT_name  (“std::vector<std::__1::basic_string<…”)
           DW_TAG_template_type_parameter
             DW_AT_type         (0x0dec "std::__1::basic_string<…")
             DW_AT_name         (“_Tp”)

0x0dec:  DW_TAG_class_type
           DW_AT_name  (“std::__1::basic_string<…”)
```

```
$ dwarfdump a.out.dSYM

0x0123:  DW_TAG_class_type
           DW_AT_name  (“map<…>")
           DW_TAG_template_type_parameter
             DW_AT_type        (0x8332 "int")
             DW_AT_name        ("_Key")
           DW_TAG_template_type_parameter
             DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
             DW_AT_name        (“_Tp")

0x0214:  DW_TAG_class_type
           DW_AT_name  (“std::vector<std::__1::basic_string<…”)
           DW_TAG_template_type_parameter
             DW_AT_type        (0x0dec "std::__1::basic_string<…")
             DW_AT_name        (“_Tp”)

0x0dec:  DW_TAG_class_type
           DW_AT_name  (“std::__1::basic_string<…”)
```

```
$ dwarfdump a.out.dSYM

0x0123:  DW_TAG_class_type
           DW_AT_name  (“map<…>")
           DW_TAG_template_type_parameter
             DW_AT_type        (0x8332 "int")
             DW_AT_name        ("_Key")
           DW_TAG_template_type_parameter
             DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
             DW_AT_name        (“_Tp")

0x0214:  DW_TAG_class_type
           DW_AT_name  (“std::vector<std::__1::basic_string<…”)
           DW_TAG_template_type_parameter
             DW_AT_type        (0x0dec "std::__1::basic_string<…")
             DW_AT_name        (“_Tp”)

0x0dec:  DW_TAG_class_type
           DW_AT_name  (“std::__1::basic_string<…”)
```

```
$ dwarfdump a.out.dSYM

 0x0123:  DW_TAG_class_type
            DW_AT_name  (“map<…>")
            DW_TAG_template_type_parameter
              DW_AT_type        (0x8332 "int")
              DW_AT_name        ("_Key")
            DW_TAG_template_type_parameter
              DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
              DW_AT_name        (“_Tp")

 0x0214:  DW_TAG_class_type
            DW_AT_name  (“std::vector<std::__1::basic_string<…”)
            DW_TAG_template_type_parameter
              DW_AT_type        (0x0b26 "string")
              DW_AT_name        (“_Tp”)

 0x0b26:  DW_TAG_typedef
              DW_AT_type        (0x0dec "std::__1::basic_string<…")
              DW_AT_name        (“string”)

 0x0dec:  DW_TAG_class_type
            DW_AT_name  (“std::__1::basic_string<…”)
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::basic_string<char, std::char_traits<char>,
std::allocator<char> >, std::allocator<std::basic_string<char,
std::char_traits<char>, std::allocator<char> > > >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::basic_string<char,
std::char_traits<char>, std::allocator<char> >,
std::allocator<std::basic_string<char, std::char_traits<char>,
std::allocator<char> > > > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string, std::allocator<std::string> >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::string,
std::allocator<std::string> > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string, std::allocator<std::string> >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::string,
std::allocator<std::string> > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
template <typename TA>
static void
printTo(raw_ostream &OS, ArrayRef<TA> Args, const PrintingPolicy &Policy,
        const TemplateParameterList *TPL, bool IsPack, unsigned ParmIndex) {
  if (TPL && Policy.SuppressDefaultTemplateArgs &&
    !Policy.PrintCanonicalTypes && !Args.empty() && !IsPack &&
    Args.size() <= TPL->size()) {
  ASTContext &Ctx = TPL->getParam(0)->getASTContext();
  llvm::SmallVector<TemplateArgument, 8> OrigArgs;
  for (const TA &A : Args)
    OrigArgs.push_back(getArgument(A));
  while (!Args.empty() &&
         isSubstitutedDefaultArgument(Ctx, getArgument(Args.back()),
                                      TPL->getParam(Args.size() - 1),
                                      OrigArgs, TPL->getDepth()))
    Args = Args.drop_back();
...
```

(from clang/lib/AST/TypePrinter.cpp)

```
template <typename TA>
static void
printTo(raw_ostream &OS, ArrayRef<TA> Args, const PrintingPolicy &Policy,
        const TemplateParameterList *TPL, bool IsPack, unsigned ParmIndex) {
  if (TPL && Policy.SuppressDefaultTemplateArgs &&
    !Policy.PrintCanonicalTypes && !Args.empty() && !IsPack &&
    Args.size() <= TPL->size()) {
  ASTContext &Ctx = TPL->getParam(0)->getASTContext();
  llvm::SmallVector<TemplateArgument, 8> OrigArgs;
  for (const TA &A : Args)
    OrigArgs.push_back(getArgument(A));
  while (!Args.empty() &&
         isSubstitutedDefaultArgument(Ctx, getArgument(Args.back()),
                                      TPL->getParam(Args.size() - 1),
                                      OrigArgs, TPL->getDepth()))
    Args = Args.drop_back();
...
```

(from clang/lib/AST/TypePrinter.cpp)

```
bool clang::isSubstitutedDefaultArgument(ASTContext &Ctx, TemplateArgument Arg,
                                          const NamedDecl *Param,
                                          ArrayRef<TemplateArgument> Args,
                                          unsigned Depth) {
  // An empty pack is equivalent to not providing a pack argument.
  if (Arg.getKind() == TemplateArgument::Pack && Arg.pack_size() == 0)
    return true;

  if (auto *TTPD = dyn_cast<TemplateTypeParmDecl>(Param)) {
    return TTPD->hasDefaultArgument() &&
           isSubstitutedTemplateArgument(Ctx, Arg, TTPD->getDefaultArgument(),
                                          Args, Depth);
```

(from clang/lib/AST/TypePrinter.cpp)

```
bool clang::isSubstitutedDefaultArgument(ASTContext &Ctx, TemplateArgument Arg,
                                         const NamedDecl *Param,
                                         ArrayRef<TemplateArgument> Args,
                                         unsigned Depth) {
  // An empty pack is equivalent to not providing a pack argument.
  if (Arg.getKind() == TemplateArgument::Pack && Arg.pack_size() == 0)
    return true;

  if (auto *TTPD = dyn_cast<TemplateTypeParmDecl>(Param)) {
    return TTPD->hasDefaultArgument() &&
           isSubstitutedTemplateArgument(Ctx, Arg, TTPD->getDefaultArgument(),
                                         Args, Depth);
```

(from clang/lib/AST/TypePrinter.cpp)

```
static TemplateParameterList *CreateTemplateParameterList(
    ASTContext &ast,
    const TypeSystemClang::TemplateParameterInfos &template_param_infos,
    llvm::SmallVector<NamedDecl *, 8> &template_param_decls) {
    ...

      auto *TTP = TemplateTypeParmDecl::Create(
          ast, decl_context, SourceLocation(), SourceLocation(), depth,
          num_template_params, identifier_info, is_typename,
          parameter_pack_true);

      template_param_decls.push_back(TTP);
    }
  }

  TemplateParameterList *template_param_list =
     TemplateParameterList::Create(
         ast, SourceLocation(), SourceLocation(), template_param_decls,
         SourceLocation(), requires_clause);
  ...
  return template_param_list;
```

(from lldb/source/Plugins/TypeSystem/Clang/TypeSystemClang.cpp)

```
static TemplateParameterList *CreateTemplateParameterList(
    ASTContext &ast,
    const TypeSystemClang::TemplateParameterInfos &template_param_infos,
    llvm::SmallVector<NamedDecl *, 8> &template_param_decls) {
    ...

      auto *TTP = TemplateTypeParmDecl::Create(
          ast, decl_context, SourceLocation(), SourceLocation(), depth,
          num_template_params, identifier_info, is_typename,
          parameter_pack_true);

      template_param_decls.push_back(TTP);
    }
  }

  TemplateParameterList *template_param_list =
     TemplateParameterList::Create(
         ast, SourceLocation(), SourceLocation(), template_param_decls,
         SourceLocation(), requires_clause);
  ...
  return template_param_list;
```

(from lldb/source/Plugins/TypeSystem/Clang/TypeSystemClang.cpp)

```
static TemplateParameterList *CreateTemplateParameterList(
    ASTContext &ast,
    const TypeSystemClang::TemplateParameterInfos &template_param_infos,
    llvm::SmallVector<NamedDecl *, 8> &template_param_decls) {
    ...

      auto *TTP = TemplateTypeParmDecl::Create(
          ast, decl_context, SourceLocation(), SourceLocation(), depth,
          num_template_params, identifier_info, is_typename,
          parameter_pack_true);
      TTP->setDefaultArgument(???);
      template_param_decls.push_back(TTP);
    }
  }

  TemplateParameterList *template_param_list =
     TemplateParameterList::Create(
         ast, SourceLocation(), SourceLocation(), template_param_decls,
         SourceLocation(), requires_clause);
  ...
  return template_param_list;
```

(from lldb/source/Plugins/TypeSystem/Clang/TypeSystemClang.cpp)

```
$ dwarfdump a.out.dSYM

DW_TAG_class_type
  DW_AT_name  ("map<int, std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>,…")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x8332 "int")
    DW_AT_name        ("_Key")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Tp")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e06 "std::__1::less<int>")
    DW_AT_name        ("_Compare")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e5d "std::__1::allocator<std::__1::pair<const int, std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Allocator")
```

```
$ dwarfdump a.out.dSYM

DW_TAG_class_type
  DW_AT_name  ("map<int, std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>,…")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x8332 "int")
    DW_AT_name        ("_Key")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Tp")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e06 "std::__1::less<int>")
    DW_AT_name        ("_Compare")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e5d "std::__1::allocator<std::__1::pair<const int, std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Allocator")
```

```
$ dwarfdump a.out.dSYM

DW_TAG_class_type
  DW_AT_name  ("map<int, std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>,…")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x8332 "int")
    DW_AT_name        ("_Key")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Tp")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e06 "std::__1::less<int>")
    DW_AT_name        ("_Compare")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e5d "std::__1::allocator<std::__1::pair<const int, std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Allocator")
```

```
$ dwarfdump a.out.dSYM

DW_TAG_class_type
  DW_AT_name  ("map<int, std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>,…")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x8332 "int")
    DW_AT_name        ("_Key")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x0214 "std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Tp")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e06 "std::__1::less<int>")
    DW_AT_name        ("_Compare")
  DW_TAG_template_type_parameter
    DW_AT_type        (0x3e5d "std::__1::allocator<std::__1::pair<const int, std::__1::vector<std::__1::basic_string<…")
    DW_AT_name        ("_Allocator")
```

```
template<typename _Key, typename _Tp,
         typename _Allocator = std::allocator<std::pair<_Key, _Tp>>>
class map;

Or

template<typename _Key, typename _Tp,
         typename _Allocator = std::allocator<std::pair<int, std::vector<…>>>>
class map;

?
```

| The entry may also have a DW_AT_default_value attribute, which is a flag indicating
| that the value corresponds to the default argument for the template parameter.

- DWARFv5 Section 2.23

| The entry may also have a DW_AT_default_value attribute, which is a flag indicating
| that the value corresponds to the default argument for the template parameter.

- DWARFv5 Section 2.23

```
$ clang++ -gdwarf-5 map.cpp
$ dwarfdump a.out.dSYM
```

| The entry may also have a DW_AT_default_value attribute, which is a flag indicating
| that the value corresponds to the default argument for the template parameter.

- DWARFv5 Section 2.23

```
$ clang++ -gdwarf-5 map.cpp
$ dwarfdump a.out.dSYM
```

```
DW_TAG_class_type
  DW_AT_name  ("map<int, std::__1::vector<std::__1::basic_string<char, std::__1::char_traits<char>,…")
  DW_TAG_template_type_parameter
    DW_AT_type          (0x8332 "int")
    DW_AT_name          ("_Key")
  DW_TAG_template_type_parameter
    DW_AT_type          (0x0214 "std::__1::vector<std::__1::basic_string<…")
    DW_AT_name          ("_Tp")
  DW_TAG_template_type_parameter
    DW_AT_type          (0x3e06 "std::__1::less<int>")
    DW_AT_name          ("_Compare")
    DW_AT_default_value (true)
  DW_TAG_template_type_parameter
    DW_AT_type          (0x3e5d "std::__1::allocator<std::__1::pair<const int, std::__1::vector<std::__1::basic_string<…")
    DW_AT_name          ("_Allocator")
    DW_AT_default_value (true)
```

(Compile time)

Clang

```
clang::TemplateArgument {
  ...
  bool IsDefaulted : 1;
};
```

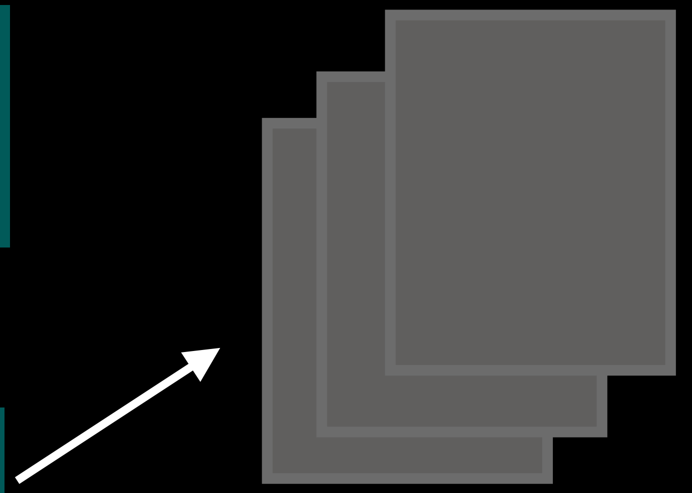
Sema

```
if(subst(Arg, Params[i]))
    Arg.SetIsDefaulted(true)
```

CGDebugInfo

```
if (Arg.GetIsDefaulted())
  addFlag(DW_AT_default_value)
```

Disk



Object Files

(Compile time)

Clang

```
clang::TemplateArgument {
  ...
  bool IsDefaulted : 1;
};
```

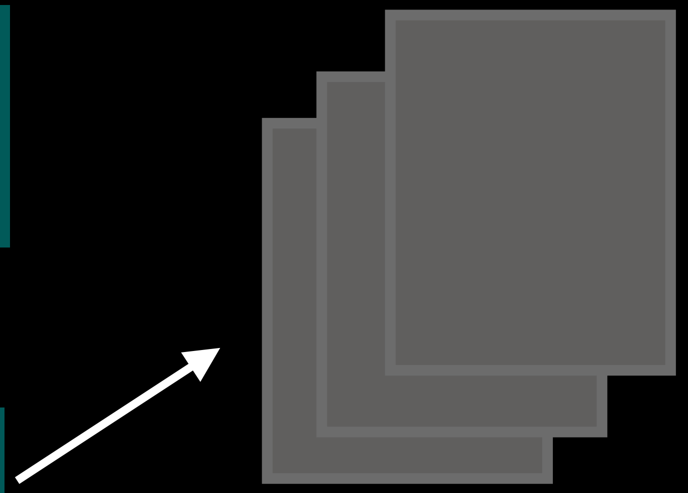
Sema

```
if(subst(Arg, Params[i]))
    Arg.SetIsDefaulted(true)
```

CGDebugInfo

```
if (Arg.GetIsDefaulted())
  addFlag(DW_AT_default_value)
```

Disk



Object Files

(Compile time)

Clang

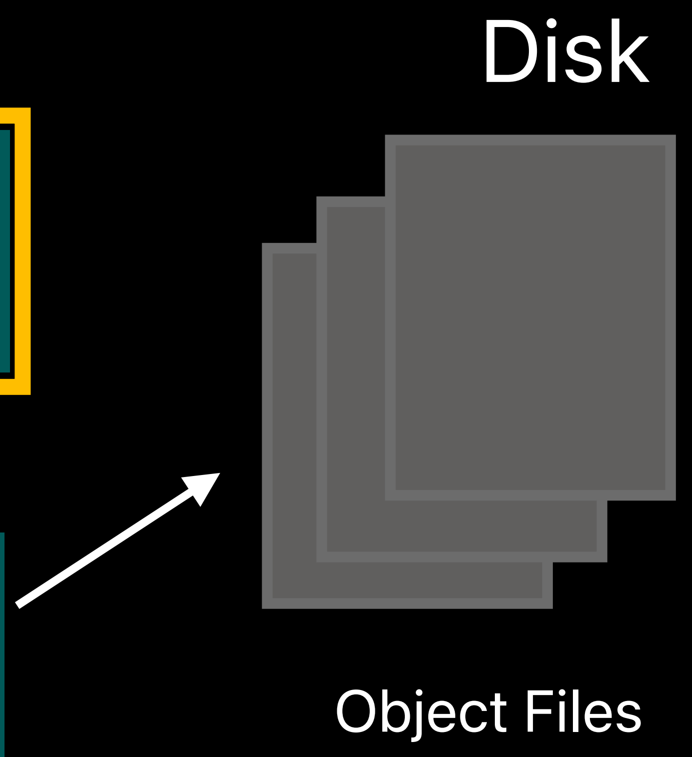
```
clang::TemplateArgument {
  ...
  bool IsDefaulted : 1;
};
```

Sema

```
if(subst(Arg, Params[i]))
    Arg.SetIsDefaulted(true)
```

CGDebugInfo

```
if (Arg.GetIsDefaulted())
  addFlag(DW_AT_default_value)
```



(Compile time)

Clang

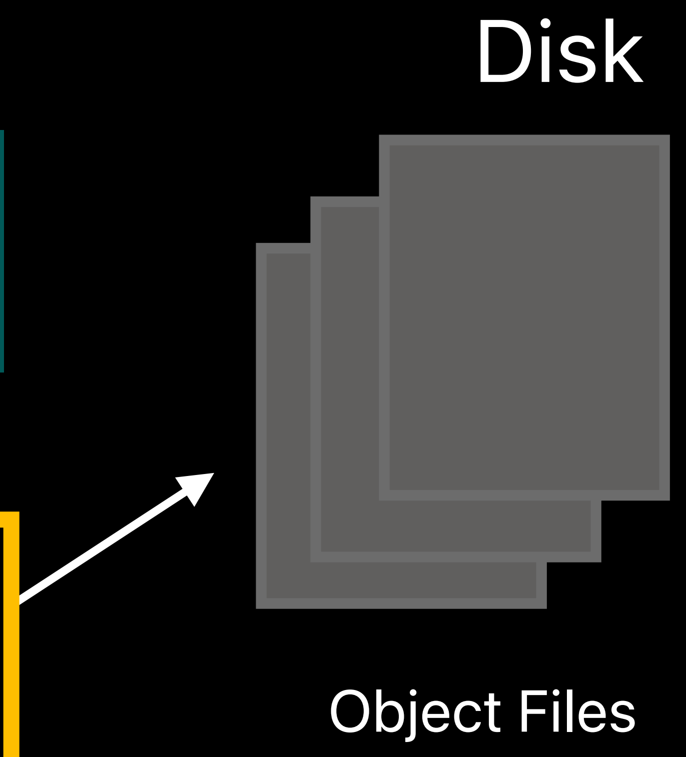
```
clang::TemplateArgument {
  ...
  bool IsDefaulted : 1;
};
```

Sema

```
if(subst(Arg, Params[i]))
    Arg.SetIsDefaulted(true)
```

CGDebugInfo

```
if (Arg.GetIsDefaulted())
  addFlag(DW_AT_default_value)
```



(Compile time)

(Debug time)

Clang

```
clang::TemplateArgument {
  ...
  bool IsDefaulted : 1;
};
```

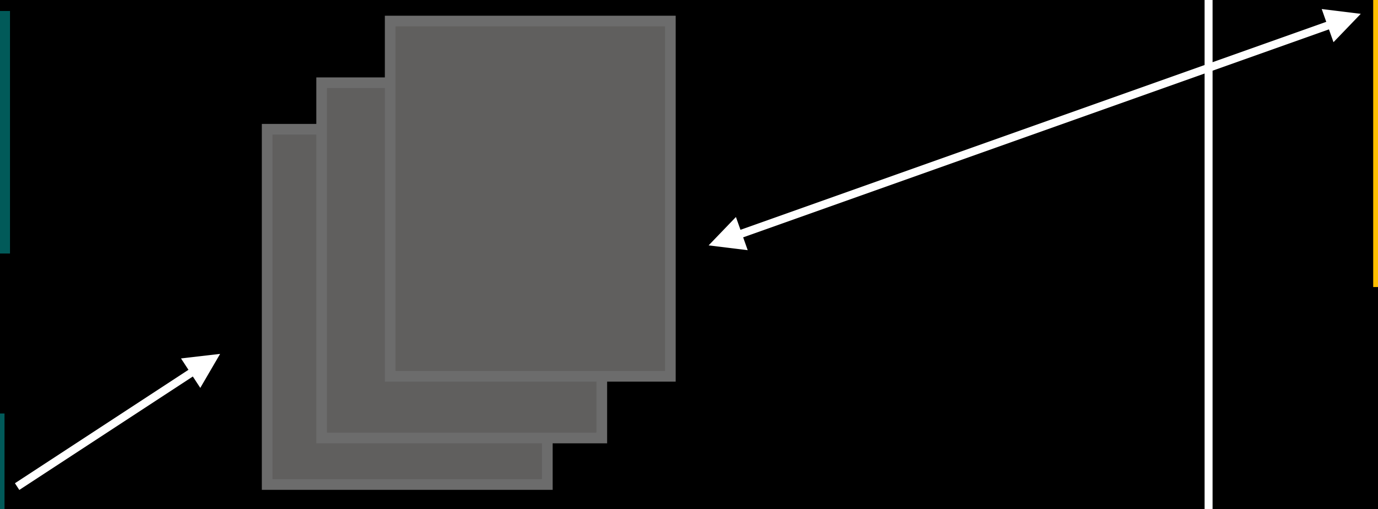
Sema

```
if(subst(Arg, Params[i]))
    Arg.SetIsDefaulted(true)
```

CGDebugInfo

```
if (Arg.GetIsDefaulted())
  addFlag(DW_AT_default_value)
```

Disk

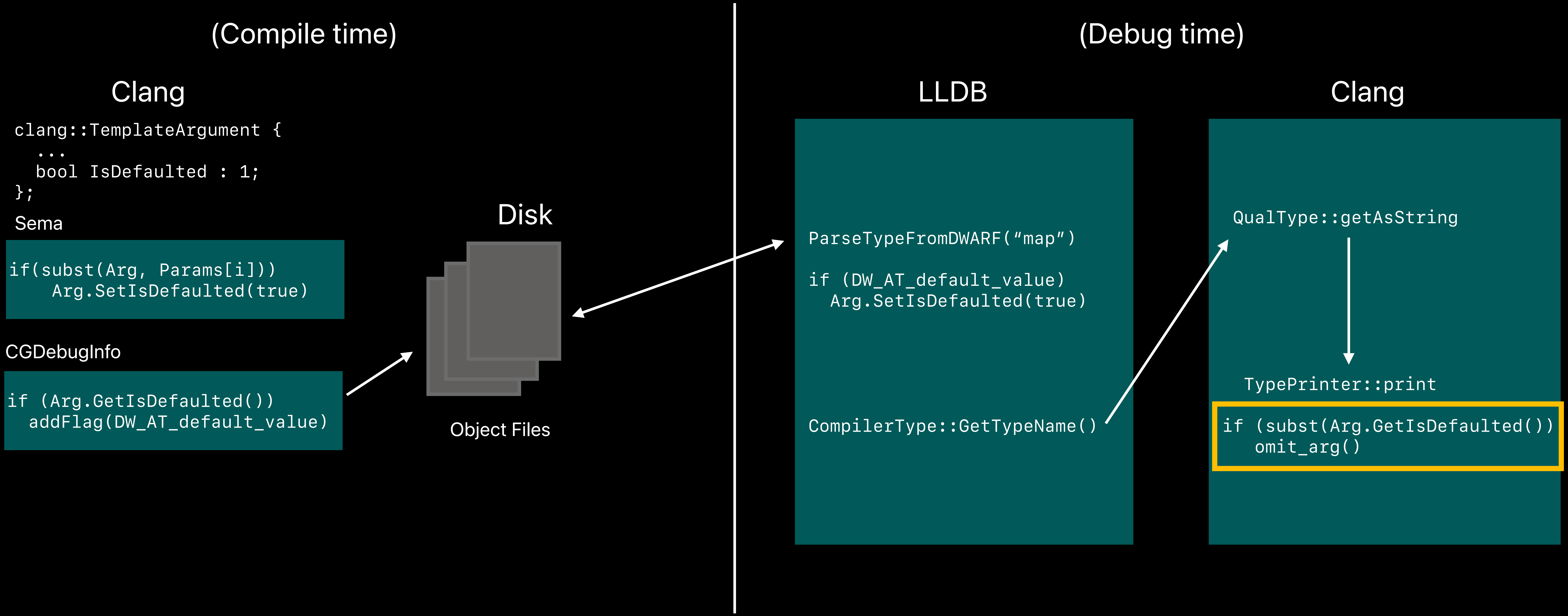


LLDB

```
ParseTypeFromDWARF("map")

if (DW_AT_default_value)
  Arg.SetIsDefaulted(true)
```

```
CompilerType::GetTypeName()
```



```
clang::TemplateArgument {
  ...
  bool IsDefaulted : 1;
};
```

Sema

```
if(subst(Arg, Params[i]))
    Arg.SetIsDefaulted(true)
```

CGDebugInfo

```
if (Arg.GetIsDefaulted())
  addFlag(DW_AT_default_value)
```

LLDB

```
ParseTypeFromDWARF("map")

if (DW_AT_default_value)
  Arg.SetIsDefaulted(true)


CompilerType::GetTypeName()
```

Clang

```
QualType::getAsString

TypePrinter::print

if (subst(Arg.GetIsDefaulted())
   omit_arg()
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string, std::allocator<std::string> >, std::less<int>,
std::allocator<std::pair<const int, std::vector<std::string,
std::allocator<std::string> > > > > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string> > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string> > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string> > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
namespace ranges {
template <range _Range>
  requires is_object_v<_Range>
class ref_view : public view_interface<ref_view<_Range>> {
  _Range* __range_;
```

(from libcxx/include/__ranges/ref_view.h)

```
(lldb) type synthetic list
…
^std::__[[:alnum:]]+::ranges::ref_view<.+>$:  libc++ std::ranges::ref_view synthetic children
…
```

```
lldb::ChildCacheState
lldb_private::formatters::LibcxxStdRangesRefViewSyntheticFrontEnd::Update() {
  ValueObjectSP range_ptr =
      GetChildMemberWithName(m_backend,{ConstString( "__range_")});
  if (!range_ptr)
    return lldb::ChildCacheState::eRefetch;

  lldb_private::Status error;
  m_range_sp = range_ptr->Dereference(error);
  …
}
```

(from lldb/source/Plugins/Language/CPlusPlus/LibCxxRangesRefView.cpp)

```
lldb::ChildCacheState
lldb_private::formatters::LibcxxStdRangesRefViewSyntheticFrontEnd::Update() {
  ValueObjectSP range_ptr =
      GetChildMemberWithName(m_backend,{ConstString( "__range_")});
  if (!range_ptr)
    return lldb::ChildCacheState::eRefetch;

  lldb_private::Status error;
  m_range_sp = range_ptr->Dereference(error);
  …
}
```

(from lldb/source/Plugins/Language/CPlusPlus/LibCxxRangesRefView.cpp)

```
lldb::ChildCacheState
lldb_private::formatters::LibcxxStdRangesRefViewSyntheticFrontEnd::Update() {
  ValueObjectSP range_ptr =
      GetChildMemberWithName(m_backend,{ConstString( "__range_")});
  if (!range_ptr)
    return lldb::ChildCacheState::eRefetch;

  lldb_private::Status error;
  m_range_sp = range_ptr->Dereference(error);
  …
}
```

(from lldb/source/Plugins/Language/CPlusPlus/LibCxxRangesRefView.cpp)

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string> > >, 1>) elems = {
  __base_ = {
    __range_ = 0x000000016fdfee18 size=2
  }
}
```

```
(lldb) p elems

(std::ranges::elements_view<std::ranges::ref_view<std::map<int,
std::vector<std::string> > >, 1>) elems = {
  __base_ = {
    *__range_ = size=2 {
      [0] = {
        first = 1
        second = size=1 {
          [0] = "foo"
        }
      }
      [1] = {
        first = 2
        second = size=1 {
          [0] = "bar"
        }
      }
    }
  }
}
```

Conclusion

LLDB uses Clang for type introspection

Conclusion

LLDB uses Clang for type introspection

LLDB relies on DWARF for AST reconstruction

Conclusion

LLDB uses Clang for type introspection

LLDB relies on DWARF for AST reconstruction

Improving debugging experience is often a balance of where we want to shift complexity to (debugger vs. compiler)

Conclusion

LLDB uses Clang for type introspection

LLDB relies on DWARF for AST reconstruction

Improving debugging experience is often a balance of where we want to shift complexity to (debugger vs. compiler)

Keep debugging and LLDB in mind when developing new language or standard library features