
Core C++ 2026
20-21Oct. 2026 :: Tel-Aviv
A New Kind of Static
Tal Yacobi

The Idea: cross_static (P3334)

- Having a static variable that is shared between function template instantiations

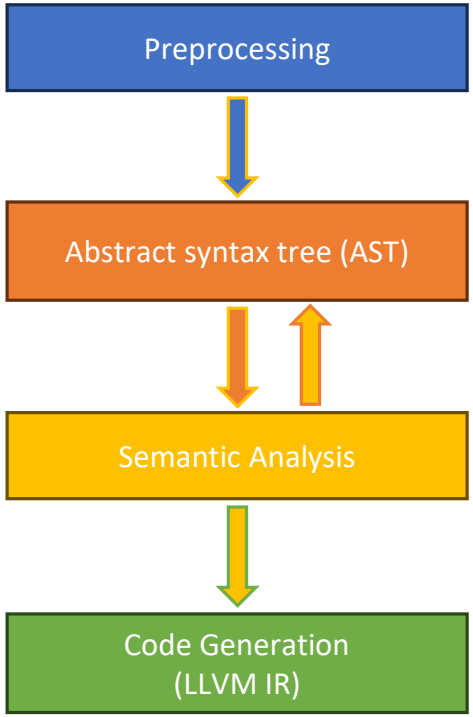
```
8   template <typename T> void f(T x) {
9       cross_static int count = 0;
10      ++count;
11      cout << count << " " << x << "\n";
12  }
13
14  int main() {
15      f<int>(x: 1);
16      f<int>(x: 2);
17      f<double>(x: 2.4);
18  }
```

```
1 1
2 2
3 2.4
```

Starting with Clang

- git clone https://github.com/llvm/llvm-project.git
- cd llvm-project
- mkdir build
- cd build
- cmake -DLLVM_ENABLE_PROJECTS=clang -DCMAKE_BUILD_TYPE=Release -GNinja ../llvm
- ninja clang

Clang Structure



Useful Commands

- Print IR: ./build/bin/clang++ -S -emit-llvm -o - <src>
- Print AST: ./build/bin/clang++ -Xclang -ast-dump -fsyntax-only <src>

Adding a New Keyword

./clang/include/clang/Basic/TokenKinds.def

tok::kw_cross_static

```
356   // C++ 2.11p1: Keywords.
357   KEYWORD(asm                         , KEYCXX|KEYGNU)
358   KEYWORD(bool                        , BOOLSUPPORT|KEYC23)
359   KEYWORD(catch                       , KEYCXX)
360   KEYWORD(class                       , KEYCXX)
361   KEYWORD(const_cast                  , KEYCXX)
362   KEYWORD(cross_static                , KEYCXX)
363   KEYWORD(delete                      , KEYCXX)
364   KEYWORD(dynamic_cast                , KEYCXX)
```

Parse the new keyword

```
7
8   template <typename T> void f(T x) {
9       static int count = 0;
10      ++count;
11  }
12
13  int main() {
14      f<int>(x: 1);
15      f<int>(x: 2);
16      f<double>(x: 2.4);
17  }
```

```
|-FunctionTemplateDecl 0x58aa67b6cee8 <clang/cross_static_test.cpp:8:1, line:11:1> line:8:28 f
| |-TemplateTypeParmDecl 0x58aa67b6cc50 <col:11, col:20> col:20 referenced typename depth 0 index 0 T
| |-FunctionDecl 0x58aa67b6ce40 <col:23, line:11:1> line:8:28 f 'void (T)'
| | |-ParmVarDecl 0x58aa67b6cd30 <col:30, col:32> col:32 x 'T'
| | `-CompoundStmt 0x58aa67b6d138 <col:35, line:11:1>
| |   |-DeclStmt 0x58aa67b6d0e8 <line:9:5, col:31>
| |   | `-VarDecl 0x58aa67b6d010 <col:5, col:30> col:22 referenced count 'int' static cinit
| |   |   `-IntegerLiteral 0x58aa67b6d080 <col:30> 'int' 0
| |   `-UnaryOperator 0x58aa67b6d120 <line:10:5, col:7> 'int' lvalue prefix '++'
| |     `-DeclRefExpr 0x58aa67b6d100 <col:7> 'int' lvalue Var 0x58aa67b6d010 'count' 'int'
| |-FunctionDecl 0x58aa67b6d5e0 <line:8:23, line:11:1> line:8:28 used f 'void (int)' implicit_instantiation instantiated_from 0x58aa67b6ce40
| | |-TemplateArgument type 'int'
| | | `-BuiltinType 0x58aa67b1dc10 'int'
| | |-ParmVarDecl 0x58aa67b6d4d0 <col:30, col:32> col:32 x 'int'
| | `-CompoundStmt 0x58aa67b8bcf8 <col:35, line:11:1>
| |   |-DeclStmt 0x58aa67b8bca8 <line:9:5, col:31>
| |   | `-VarDecl 0x58aa67b8bbf0 <col:5, col:30> col:22 used count 'int' static cinit
| |   |   `-IntegerLiteral 0x58aa67b6d080 <col:30> 'int' 0
| |   `-UnaryOperator 0x58aa67b8bce0 <line:10:5, col:7> 'int' lvalue prefix '++'
| |     `-DeclRefExpr 0x58aa67b8bcc0 <col:7> 'int' lvalue Var 0x58aa67b8bbf0 'count' 'int'
| `-FunctionDecl 0x58aa67b8b9c0 <line:8:23, line:11:1> line:8:28 used f 'void (double)' implicit_instantiation instantiated_from 0x58aa67b6ce40
|   |-TemplateArgument type 'double'
|   | `-BuiltinType 0x58aa67b1dd30 'double'
|   |-ParmVarDecl 0x58aa67b8b8b0 <col:30, col:32> col:32 x 'double'
|   `-CompoundStmt 0x58aa67b8be20 <col:35, line:11:1>
|     |-DeclStmt 0x58aa67b8bdd0 <line:9:5, col:31>
|     | `-VarDecl 0x58aa67b8bd18 <col:5, col:30> col:22 used count 'int' static cinit
|     |   `-IntegerLiteral 0x58aa67b6d080 <col:30> 'int' 0
|     `-UnaryOperator 0x58aa67b8be08 <line:10:5, col:7> 'int' lvalue prefix '++'
|       `-DeclRefExpr 0x58aa67b8bde8 <col:7> 'int' lvalue Var 0x58aa67b8bd18 'count' 'int'
`-FunctionDecl 0x58aa67b6d1b8 <line:13:1, line:17:1> line:13:5 main 'int ()'
  `-CompoundStmt 0x58aa67b8bbc8 <col:12, line:17:1>
    |-CallExpr 0x58aa67b6d7b8 <line:14:5, col:13> 'void'
    | |-ImplicitCastExpr 0x58aa67b6d7a0 <col:5, col:10> 'void (*)(int)' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x58aa67b6d6e0 <col:5, col:10> 'void (int)' lvalue Function 0x58aa67b6d5e0 'f' 'void (int)' (FunctionTemplate 0x58aa67b6cee8 'f')
    | `-IntegerLiteral 0x58aa67b6d338 <col:12> 'int' 1
    |-CallExpr 0x58aa67b6da08 <line:15:5, col:13> 'void'
    | |-ImplicitCastExpr 0x58aa67b6d9f0 <col:5, col:10> 'void (*)(int)' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x58aa67b6d998 <col:5, col:10> 'void (int)' lvalue Function 0x58aa67b6d5e0 'f' 'void (int)' (FunctionTemplate 0x58aa67b6cee8 'f')
    | `-IntegerLiteral 0x58aa67b6d878 <col:12> 'int' 2
    `-CallExpr 0x58aa67b8bb98 <line:16:5, col:18> 'void'
      |-ImplicitCastExpr 0x58aa67b8bb80 <col:5, col:13> 'void (*)(double)' <FunctionToPointerDecay>
      | `-DeclRefExpr 0x58aa67b8bac0 <col:5, col:13> 'void (double)' lvalue Function 0x58aa67b8b9c0 'f' 'void (double)' (FunctionTemplate 0x58aa67b6cee8 'f')
      `-FloatingLiteral 0x58aa67b6dac8 <col:15> 'double' 2.400000e+00
```

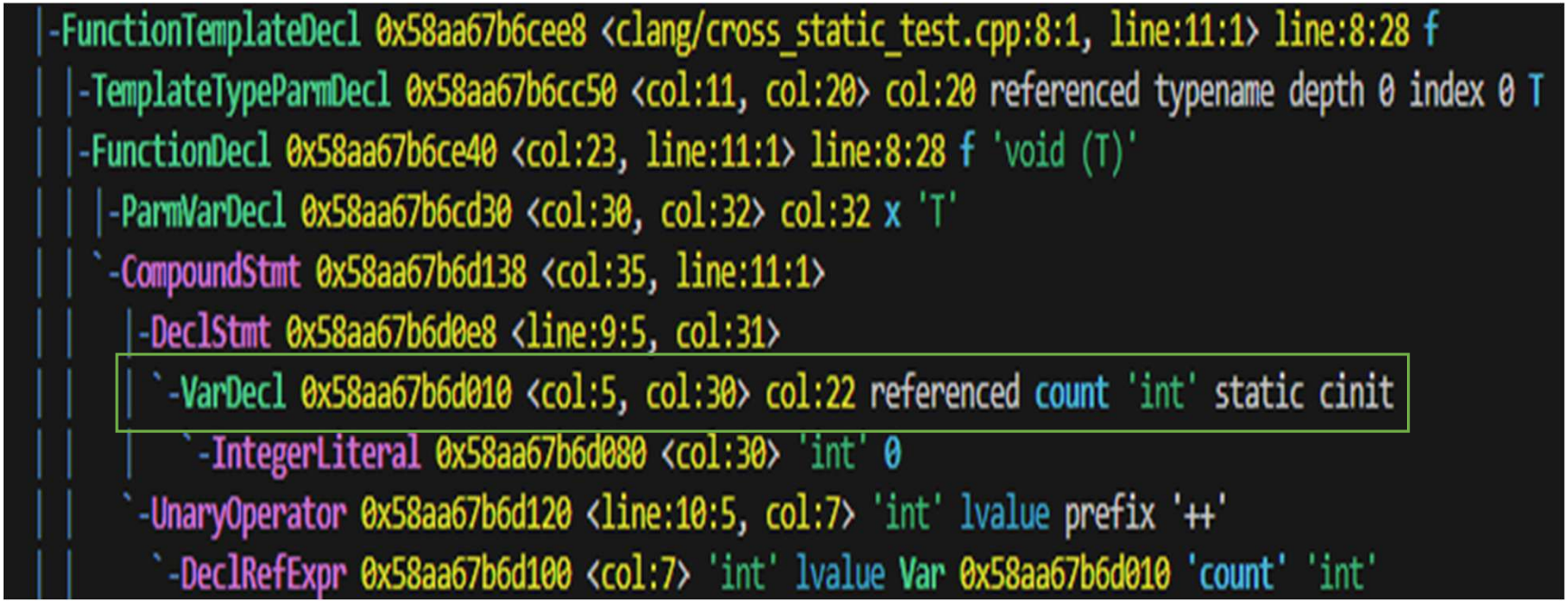
Main

```
`-FunctionDecl 0x58aa67b6d1b8 <line:13:1, line:17:1> line:13:5 main 'int ()'
  `-CompoundStmt 0x58aa67b8bbc8 <col:12, line:17:1>
    |-CallExpr 0x58aa67b6d7b8 <line:14:5, col:13> 'void'
    | |-ImplicitCastExpr 0x58aa67b6d7a0 <col:5, col:10> 'void (*)(int)' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x58aa67b6d6e0 <col:5, col:10> 'void (int)' lvalue Function 0x58aa67b6d5e0 'f' 'void (int)' (FunctionTemplate 0x58aa67b6cee8 'f')
    | `-IntegerLiteral 0x58aa67b6d338 <col:12> 'int' 1
    |-CallExpr 0x58aa67b6da08 <line:15:5, col:13> 'void'
    | |-ImplicitCastExpr 0x58aa67b6d9f0 <col:5, col:10> 'void (*)(int)' <FunctionToPointerDecay>
    | | `-DeclRefExpr 0x58aa67b6d998 <col:5, col:10> 'void (int)' lvalue Function 0x58aa67b6d5e0 'f' 'void (int)' (FunctionTemplate 0x58aa67b6cee8 'f')
    | `-IntegerLiteral 0x58aa67b6d878 <col:12> 'int' 2
    `-CallExpr 0x58aa67b8bb98 <line:16:5, col:18> 'void'
      |-ImplicitCastExpr 0x58aa67b8bb80 <col:5, col:13> 'void (*)(double)' <FunctionToPointerDecay>
      | `-DeclRefExpr 0x58aa67b8bac0 <col:5, col:13> 'void (double)' lvalue Function 0x58aa67b8b9c0 'f' 'void (double)' (FunctionTemplate 0x58aa67b6cee8 'f')
      `-FloatingLiteral 0x58aa67b6dac8 <col:15> 'double' 2.400000e+00
```

Instantiations

```
| |-FunctionDecl 0x58aa67b6d5e0 <line:8:23, line:11:1> line:8:28 used f 'void (int)' implicit_instantiation instantiated_from 0x58aa67b6ce40
| | |-TemplateArgument type 'int'
| | | `-BuiltinType 0x58aa67b1dc10 'int'
| | |-ParmVarDecl 0x58aa67b6d4d0 <col:30, col:32> col:32 x 'int'
| | `-CompoundStmt 0x58aa67b8bcf8 <col:35, line:11:1>
| |   |-DeclStmt 0x58aa67b8bca8 <line:9:5, col:31>
| |   | `-VarDecl 0x58aa67b8bbf0 <col:5, col:30> col:22 used count 'int' static cinit
| |   |   `-IntegerLiteral 0x58aa67b6d080 <col:30> 'int' 0
| |   `-UnaryOperator 0x58aa67b8bce0 <line:10:5, col:7> 'int' lvalue prefix '++'
| |     `-DeclRefExpr 0x58aa67b8bcc0 <col:7> 'int' lvalue Var 0x58aa67b8bbf0 'count' 'int'
| `-FunctionDecl 0x58aa67b8b9c0 <line:8:23, line:11:1> line:8:28 used f 'void (double)' implicit_instantiation instantiated_from 0x58aa67b6ce40
|   |-TemplateArgument type 'double'
|   | `-BuiltinType 0x58aa67b1dd30 'double'
|   |-ParmVarDecl 0x58aa67b8b8b0 <col:30, col:32> col:32 x 'double'
|   `-CompoundStmt 0x58aa67b8be20 <col:35, line:11:1>
|     |-DeclStmt 0x58aa67b8bdd0 <line:9:5, col:31>
|     | `-VarDecl 0x58aa67b8bd18 <col:5, col:30> col:22 used count 'int' static cinit
|     |   `-IntegerLiteral 0x58aa67b6d080 <col:30> 'int' 0
|     `-UnaryOperator 0x58aa67b8be08 <line:10:5, col:7> 'int' lvalue prefix '++'
|       `-DeclRefExpr 0x58aa67b8bde8 <col:7> 'int' lvalue Var 0x58aa67b8bd18 'count' 'int'
```

Function Template Declaration



DeclSpec.h: add cross-static specifier

```
217  class DeclSpec {
333    LLVM_PREFERRED_TYPE(bool)
334    unsigned CrossStatic : 1;
```

```
455  DeclSpec(AttributeFactory &attrFactory)
456      : StorageClassSpec(SCS_unspecified),
457        ThreadStorageClassSpec(TSCS_unspecified),
458        SCS_extern_in_linkage_spec(false),
459        CrossStatic(false),
460        TypeSpecWidth(static_cast<unsigned>(TypeSpecifierWidth::Unspecified)),
461        TypeSpecComplex(TSC_unspecified),
```

```
483    bool isCrossStatic() const { return CrossStatic; }
484    void SetCrossStatic(bool Value) {
485      CrossStatic = Value;
486    }
```

Prase tok::kw_cross_static

```
4088    case tok::kw_cross_static:
4089      DS.SetCrossStatic(true);
4090      [[fallthrough]];
4091    case tok::kw_static:
4092      if (DS.getThreadStorageClassSpec() == DeclSpec::TSCS___thread)
4093        Diag(Tok, diag::ext_thread_before) << "static";
4094      isInvalid = DS.SetStorageClassSpec(Actions, DeclSpec::SCS_static, Loc,
4095                                         PrevSpec, DiagID, Policy);
4096      isStorageClass = true;
4097      break;
```

Decl.h: add cross static parameter

```
 56  namespace clang {
926  class VarDecl : public DeclaratorDecl, public Redeclarable<VarDecl> {
```

```
class NonParmVarDeclBitfields {
1123
1124      /// Whether this variable is cross static or not.
1125      LLVM_PREFERRED_TYPE(bool)
1126      unsigned IsCrossStatic : 1;
```

```
1594    bool isCrossStatic() const {
1595      return isa<ParmVarDecl>(this) ? false : NonParmVarDeclBits.IsCrossStatic;
1596    }
1597    void setCrossStatic(bool CS) {
1598      assert(!isa<ParmVarDecl>(this));
1599      NonParmVarDeclBits.IsCrossStatic = CS;
1600    }
```

DiagnosticSemaKinds.td: add errors

```
13694  def err_cross_static_outside_of_template : Error<
13695    "cross_static can only be specified inside of a function template context (%0)">;
13696  def err_cross_static_inside_of_template_spec : Error<
13697    "cross_static can't be specified inside of a function template specialization context (%0)">;
```

SemaDecl.cpp: emit error

```
7695  NamedDecl *Sema::ActOnVariableDeclarator(
7719    if (D.getDeclSpec().isCrossStatic()) {
7720      const auto *FD: FunctionDecl const * = dyn_cast<FunctionDecl>(Val: cast<Decl>(Val: DC));
7721      if (!FD) {
7722        Diag(Loc: D.getIdentifierLoc(), DiagID: diag::err_cross_static_outside_of_template) << Name;
7723        return nullptr;
7724      }
7725      switch (FD->getTemplatedKind()) {
7726      case FunctionDecl::TK_NonTemplate:
7727        Diag(Loc: D.getIdentifierLoc(), DiagID: diag::err_cross_static_outside_of_template) << Name;
7728        return nullptr;
7729      case FunctionDecl::TK_FunctionTemplateSpecialization:
7730        Diag(Loc: D.getIdentifierLoc(), DiagID: diag::err_cross_static_inside_of_template_spec) << Name;
7731        return nullptr;
7732      default:
7733        break;
7734      }
7735    }
```

SemaDecl.cpp: copy cross-static to VarDecl

```
7695    NamedDecl *Sema::ActOnVariableDeclarator(
8057
8058      NewVD->setCrossStatic(CS: D.getDeclSpec().isCrossStatic());
8059
```

```
6050    void Sema::BuildVariableInstantiation(
6051            VarDecl *NewVar, VarDecl *OldVar,
6087      NewVar->setCrossStatic(CS: OldVar->isCrossStatic());
```

```cpp

template <typename T> void f(T x) {
    static int count = 0;
    ++count;
}

int main() {
    f<int>(x: 1);
    f<int>(x: 2);
    f<double>(x: 2.4);
}

```

```llvm
@_ZZ1fIiEvT_E5count = linkonce_odr dso_local global i32 0, comdat, align 4
@_ZZ1fIdEvT_E5count = linkonce_odr dso_local global i32 0, comdat, align 4

; Function Attrs: mustprogress noinline norecurse optnone uwtable
define dso_local noundef i32 @main() #0 {
  call void @_Z1fIiEvT_(i32 noundef 1)
  call void @_Z1fIiEvT_(i32 noundef 2)
  call void @_Z1fIdEvT_(double noundef 2.400000e+00)
  ret i32 0
}

; Function Attrs: mustprogress noinline nounwind optnone uwtable
define linkonce_odr dso_local void @_Z1fIiEvT_(i32 noundef %0) #1 comdat {
  %2 = alloca i32, align 4
  store i32 %0, ptr %2, align 4
  %3 = load i32, ptr @_ZZ1fIiEvT_E5count, align 4
  %4 = add nsw i32 %3, 1
  store i32 %4, ptr @_ZZ1fIiEvT_E5count, align 4
  ret void
}

; Function Attrs: mustprogress noinline nounwind optnone uwtable
define linkonce_odr dso_local void @_Z1fIdEvT_(double noundef %0) #1 comdat {
  %2 = alloca double, align 8
  store double %0, ptr %2, align 8
  %3 = load i32, ptr @_ZZ1fIdEvT_E5count, align 4
  %4 = add nsw i32 %3, 1
  store i32 %4, ptr @_ZZ1fIdEvT_E5count, align 4
  ret void
}

```

Code Gen: What to do?

```
256  llvm::Constant *CodeGenModule::getOrCreateStaticVarDecl(
257      const VarDecl &D, llvm::GlobalValue::LinkageTypes Linkage) {
258    // In general, we don't always emit static var decls once before we reference
259    // them. It is possible to reference them before emitting the function that
260    // contains them, and it is possible to emit the containing function multiple
261    // times.
262    if (llvm::Constant *ExistingGV = StaticLocalDeclMap[&D])
263      return ExistingGV;
```

Decl.h: Add original VarDecl Address

```
 56  namespace clang {
926  class VarDecl : public DeclaratorDecl, public Redeclarable<VarDecl> {
978
979    /// A pointer to the original VarDecl this
980    /// instance was instantiated from
981    VarDecl *Origin = nullptr;
982
```

```
2819  VarDecl *VarDecl::getTemplatedOrigin() const {
2820    return Origin;
2821  }
2822
2823  void VarDecl::setTemplatedOrigin(VarDecl *VD) {
2824    Origin = VD;
2825  }
```

Save original VarDecl on instantiation

```
6050    void Sema::BuildVariableInstantiation(
6051            VarDecl *NewVar, VarDecl *OldVar,
6068
6069    NewVar->setTemplatedOrigin(Template: OldVar);
6070
```

CGDecl.h: use template origin address

```
256  llvm::Constant *CodeGenModule::getOrCreateStaticVarDecl(
257      const VarDecl &D, llvm::GlobalValue::LinkageTypes Linkage) {
258    // In general, we don't always emit static var decls once before we reference
259    // them. It is possible to reference them before emitting the function that
260    // contains them, and it is possible to emit the containing function multiple
261    // times.
262    const VarDecl *DeclAddr = &D;
263    if (D.isCrossStatic())
264      DeclAddr = D.getTemplatedOrigin();
265
266    if (llvm::Constant *ExistingGV = StaticLocalDeclMap[DeclAddr])
267      return ExistingGV;
```

Init Guard

```
@_ZZ1fIiEvT_E5count = linkonce_odr dso_local global i32 0, comdat, align 4
@_ZZ1fIdEvT_E5count = linkonce_odr dso_local global i32 0, comdat, align 4

; Function Attrs: mustprogress noinline norecurse optnone uwtable
define dso_local noundef i32 @main() #0 {
  call void @_Z1fIiEvT_(i32 noundef 1)
  call void @_Z1fIiEvT_(i32 noundef 2)
  call void @_Z1fIdEvT_(double noundef 2.400000e+00)
  ret i32 0
}

; Function Attrs: mustprogress noinline nounwind optnone uwtable
define linkonce_odr dso_local void @_Z1fIiEvT_(i32 noundef %0) #1 comdat {
  %2 = alloca i32, align 4
  store i32 %0, ptr %2, align 4
  %3 = load i32, ptr @_ZZ1fIiEvT_E5count, align 4
  %4 = add nsw i32 %3, 1
  store i32 %4, ptr @_ZZ1fIiEvT_E5count, align 4
  ret void
}

; Function Attrs: mustprogress noinline nounwind optnone uwtable
define linkonce_odr dso_local void @_Z1fIdEvT_(double noundef %0) #1 comdat {
  %2 = alloca double, align 8
  store double %0, ptr %2, align 8
  %3 = load i32, ptr @_ZZ1fIdEvT_E5count, align 4
  %4 = add nsw i32 %3, 1
  store i32 %4, ptr @_ZZ1fIdEvT_E5count, align 4
  ret void
}

```

Emit init guard

```
2691  void ItaniumCXXABI::EmitGuardedInit(CodeGenFunction &CGF,
2692                                      const VarDecl &D,
2735    // Create the guard variable if we don't already have it (as we
2736    // might if we're double-emitting this function body).
2737    const VarDecl *DeclAddr = &D;
2738    if (D.isCrossStatic())
2739      DeclAddr = D.getTemplatedOrigin();
2740
2741    llvm::GlobalVariable *guard = CGM.getStaticLocalDeclGuardAddress(D: DeclAddr);
```

Questions?



P334



Example