Debug Info

Status and Directions

Eric Christopher
echristo@google.com

Introduction

What works today?

What doesn't work?

Where are we going?

Debugging a few years ago?

```
std::cout << "My variable is: " << MyVar << "\n";



assert(false && "Why are we here?");
```

Debugging today!

So what really works?

Debugging

C, C++, Objective-C(++)

> 97% of the gdb testsuite, 100% of the lldb testsuite

C++ 11 Status

These things are done:

- rvalue references
- enum classes
- enums with fixed types
- enum forward declarations
- unions with special member functions
- inline namespaces
- nullptr_t
- lambdas*

So what doesn't work?

GDB Testsuite

Stabs?
Unused types
C++ Template Edge Cases
Labels
Line info for constants
Unused struct parameters
TLS variables

Line break interpretations

PR14330

Unused Struct Parameters

```
struct foo { long a, b, c, d; };
void func(foo f, int i) {
}

int main() {
  foo f;
  func(f, 3);
}
```

```
ptype func

type = void (foo, int)

type = void (int)
```

Referenced Constants

```
int main() {
  FILE *f = stderr;
}
```

So what doesn't work?

C++ 11 debugging support isn't complete

DWARF4 Missing Features

Optimized and LTO Debugging

Where are we going from here?

C++ 11 Status

These things are not done:

atomic types
template aliases
user defined literals
capture 'this' in a lambda

Immediate

DWARF4: Finish off full support

DWARF5: Fission

DWARF5: Accelerated Access

DWARF4

DWARF4 - Size Optimizations

Type Units

Encoding changes

Compression techniques

DWARF5 - Fission

Splitting debug information

Complete implementation

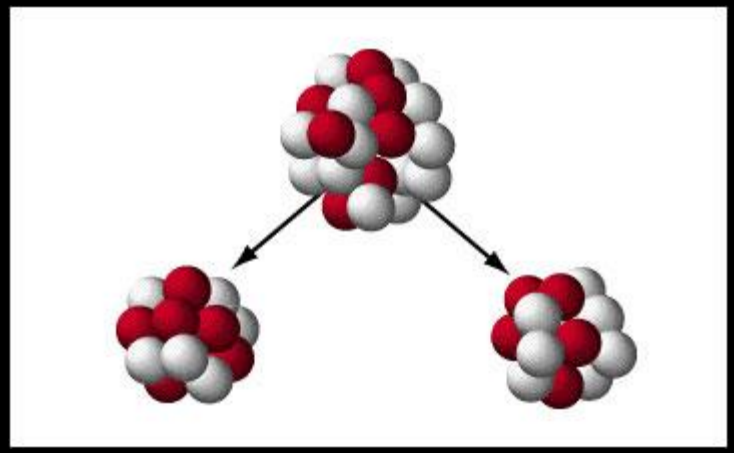
Submitted to committee

DWARF5 - Fission

Fewer Relocations

Parallelizable Linking

Faster initial link step



DWARF5 - Accelerated Access

Speeding up debugger access

Complete implementation

Submitted to committee

DWARF5 - Accelerated Access

Compact tables

Fast access

Extensible

Strictly specified contents

Near Term - LTO and Optimized Code

LTO and Optimized Code

Variable tracking

Type merging

Representation Segue

A.h:

```
class A {
  int a;
};

A a;
```

```
!llvm.dbg.cu = !{!0}


!0 = metadata !{i32 786449, metadata !1, i32 4, metadata !"clang version 3.3 (trunk 180775) (llvm/trunk 180776)", i1 false, metadata !"", i32 0, metadata !2, metadata !2, metadata !2, metadata !3, metadata !2, metadata !""} ; [ DW_TAG_compile_unit ] [/usr/local/google/home/echristo/tmp/bar.cpp] [DW_LANG_C_plus_plus]

!1 = metadata !{metadata !"bar.cpp", metadata !"/usr/local/google/home/echristo/tmp"}

!2 = metadata !{i32 0}

!3 = metadata !{metadata !4}

!4 = metadata !{i32 786484, i32 0, null, metadata !"a", metadata !"a", metadata !"", metadata !5, i32 5, metadata !6, i32 0, i32 1, %class.A* @a, null} ; [ DW_TAG_variable ] [a] [line 5] [def]

!5 = metadata !{i32 786473, metadata !1}          ; [ DW_TAG_file_type ] [/usr/local/google/home/echristo/tmp/bar.cpp]

!6 = metadata !{i32 786434, metadata !1, null, metadata !"A", i32 1, i64 32, i64 32, i32 0, i32 0, null, metadata !7, i32 0, null, null} ; [ DW_TAG_class_type ] [A] [line 1, size 32, align 32, offset 0] [from ]

!7 = metadata !{metadata !8, metadata !10}

!8 = metadata !{i32 786445, metadata !1, metadata !6, metadata !"a", i32 2, i64 32, i64 32, i64 0, i32 1, metadata !9} ; [ DW_TAG_member ] [a] [line 2, size 32, align 32, offset 0] [private] [from int]

!9 = metadata !{i32 786468, null, null, metadata !"int", i32 0, i64 32, i64 32, i64 0, i32 0, i32 5} ; [ DW_TAG_base_type ] [int] [line 0, size 32, align 32, offset 0, enc DW_ATE_signed]

!10 = metadata !{i32 786478, metadata !1, metadata !6, metadata !"A", metadata !"A", metadata !"", i32 1, metadata !11, i1 false, i1 false, i32 0, i32 0, null, i32 320, i1 false, null, null, i32 0, metadata !14, i32 1} ; [ DW_TAG_subprogram ] [line 1] [A]

!11 = metadata !{i32 786453, i32 0, i32 0, metadata !"", i32 0, i64 0, i64 0, i64 0, i32 0, null, metadata !12, i32 0, i32 0} ; [ DW_TAG_subroutine_type ] [line 0, size 0, align 0, offset 0] [from ]

!12 = metadata !{null, metadata !13}

!13 = metadata !{i32 786447, i32 0, i32 0, metadata !"", i32 0, i64 64, i64 64, i64 0, i32 1088, metadata !6} ; [ DW_TAG_pointer_type ] [line 0, size 64, align 64, offset 0] [artificial] [from A]

!14 = metadata !{i32 786468}
```

LTO - Type Merging

```
!4 = metadata !{i32 786484, i32 0, null, metadata !"a", metadata !"a", metadata
!"", metadata !5, i32 5, metadata !6, i32 0, i32 1, %class.A* @a, null} ; [
DW_TAG_variable ] [a] [line 5] [def]
```

```
!6 = metadata !{i32 786434, metadata !1, null, metadata !"A", i32 1, i64 32, i64
32, i32 0, i32 0, null, metadata !7, i32 0, null, null} ; [ DW_TAG_class_type ]
[A] [line 1, size 32, align 32, offset 0] [from ]
```

LTO - Type Merging

Foo.cpp:

```
#include "A.h"
#include "B.h"
```

Bar.cpp:

```
#include "B.h"
#include "A.h"
```

LTO - Type Merging

```
!4 = metadata !{i32 786484, i32 0, null, metadata !"a", metadata !"a", metadata
!"", metadata !5, i32 5, metadata !6, i32 0, i32 1, %class.A* @a, null} ; [
DW_TAG_variable ] [a] [line 5] [def]
```

```
!6 = metadata !{i32 786434, metadata !1, null, metadata !"A", i32 1, i64 32, i64
32, i32 0, i32 0, null, metadata !7, i32 0, null, null} ; [ DW_TAG_class_type ]
[A] [line 1, size 32, align 32, offset 0] [from ]
```

LTO - Line Tables Only

Line tables... -gline-tables-only

and some minimal DIEs.

C++ Modules

Plenty of ideas...

... no concrete plans

ASTs for types

with DWARF for line tables

and more DWARF for archival purposes?

How Can I Help?

C++ 11 Features

PR14330

Identify and report bugs

Discover size optimizations

Donuts?

Questions?

Image Credits

hyperboleandahalf.blogspot.com
retiredindelaware.blogspot.com
icanhazcheeseburger.com
chemiphysic.blogfa.com