Debug Info on a Diet

The Tale of PR7554 and The Very Patient Users

PR7554

“debug information generated by clang much larger than gcc's, making linking clang objects 20% slower than gcc”

“That looks kind of… not good” - Invader Zim

Beginning in the Middle

- Filed in 2010, 5x overhead for Clang over GCC 4.4
- Various discussion/changes occurred
- Nico Webber updated in 2012, overhead was now ‘only’ 20%
- Provided logging_chrome.cc, 3.4x larger with Clang than GCC
- 18 months later, I start looking into this…

Indirection

– – –

```
int func(void (*)());
int func(int);

struct foo {
  enum { ID = 42 };
  static void bar();
  // Big, scary debug info here…
};
```

```
// Neither emit 'foo':
int i = func(foo::bar);

// Both emit 'foo':
int j = func(foo::ID);

struct bar {
  bar() {

    // Neither emit 'foo':
    func(foo::bar);

    // Clang emits 'foo', GCC does not:
    func(foo::ID);
  }
} b;
```

Implicit Special Members

```
struct foo {
  int i;
};

void func(foo*);

int main() {
  foo f;
  func(&f);
}
```

```
define void @test() {
  %f = alloca %struct.foo
  call void @foo(%struct.
foo* %f)
}
```

But…

```
DW_TAG_structure_type
  DW_AT_name “foo”
  DW_TAG_subprogram
    DW_AT_name “foo”
    DW_AT_artificial
    DW_TAG_format_parameter
      DW_AT_type foo*
      …
```

Member Function Templates

```
struct foo {
  template <typename T>
  void func() {}
};

inline void caller() {
  foo().func<int>();
}

foo f;
```

Again, no code for func<int> (like the implicit special members), yet the DWARF describes this function.

Only describe it when we actually IRGen it (eg: when '::caller' is actually called/live)

Looking At The Other Side

Sort the string section, diff the two, pick a good chunk of similar strings that are missing from GCC and go figure out why…

```
#include <fstream>

int main() {
  std::ifstream f;
  return f.bad();
}
```

```
GCC:
DW_TAG_class_type
  DW_AT_name “basic_ifstream<char>”
  DW_AT_declaration
  DW_TAG_template_type_parameter

Clang:
DW_TAG_class_type
  DW_AT_name “basic_ifstream<char>”
  ...170 lines later…

& that doesn’t include indirectly
referenced entities…
```

Incorrect Conclusions

Was this to blame?

```
extern template class
basic_ifstream<char>;
```

Not exactly...

```
struct a { };
template<typename T>
struct b : virtual a {
  void func() {
  }
};

extern template class b<int>;

int main() {
  b<int> x;
  x.func();
}
```

Key Function Optimization

```
___
struct base {
  virtual void func() {
  }
};
struct foo: base {
  enum { ID };
};
struct reg { reg(int); };
reg r(foo::ID);
```

```
GCC:
DW_TAG_structure_type
  DW_AT_name “base”
  DW_AT_declaration

Clang:
DW_TAG_structure_type
  DW_AT_name “base”
  DW_AT_subprogram
    ...
```

Numbers

Key Function Optimization:
  23% reduction in debug info size

Overall for Chrome:
  40% smaller executable than GCC, faster links than GCC

Overall for a large server binary:
  50% smaller .o debug info
  60% smaller .dwo debug info
  20% smaller executable debug info
  40% fewer relocations