Moving to the Apple LLVM Compiler

Session 307

Bob Wilson
Manager, LLVM Core Team

Apple Has Moved to LLVM!

Now it's your turn…

Compiler History

- GNU GCC compiler
  - Portable and familiar
  - Good performance
- Limitations
  - Confusing error messages
  - No integration with other tools
  - Hard to improve

LLVM: A Modern Compiler

LLVM: A Modern Compiler

- Reusable compiler components
  - Clang front end
  - Optimizations
  - Machine code generators
  - Low-level tools
- Open source!
  - http://llvm.org/

LLVM Compilers in Xcode

LLVM-GCC: Legacy compiler

LLVM Back End

LLVM Compilers in Xcode

LLVM-GCC: Legacy compiler

LLVM Back End

GCC Front End

LLVM Compilers in Xcode

Apple LLVM Compiler

LLVM Back End

Clang Front End

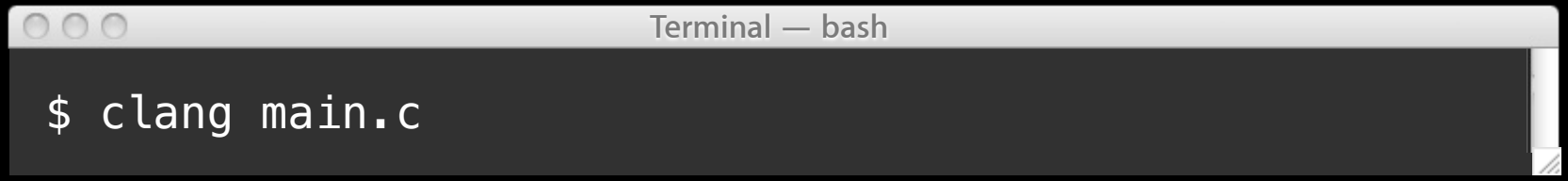
Apple LLVM Compiler 3 Features

- Helpful diagnostics
- Integrated with Xcode
- Static analyzer
- New C++ language features
- Great performance

Feature Highlights

Doug Gregor
Clang Technical Lead

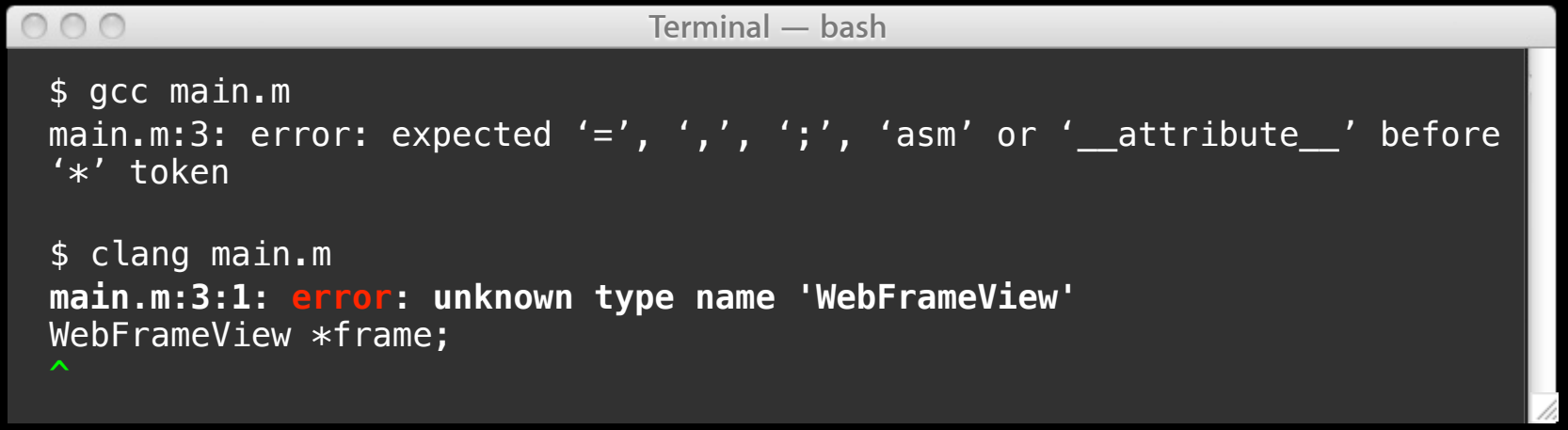
Compiler "User Interface"



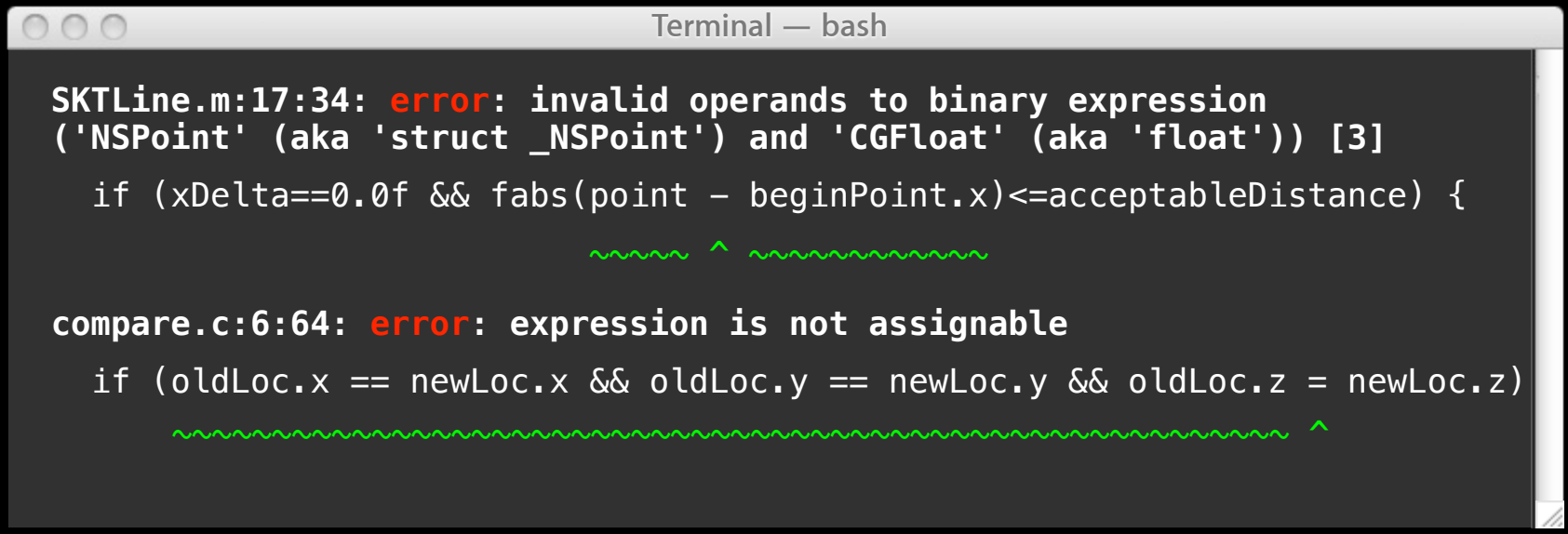
- Command-line interface matches GCC
- Drop-in compatible for makefile-based projects
  - Replace "`gcc`" with "`clang`"
  - Replace "`g++`" with "`clang++`"

Compiler Diagnostics

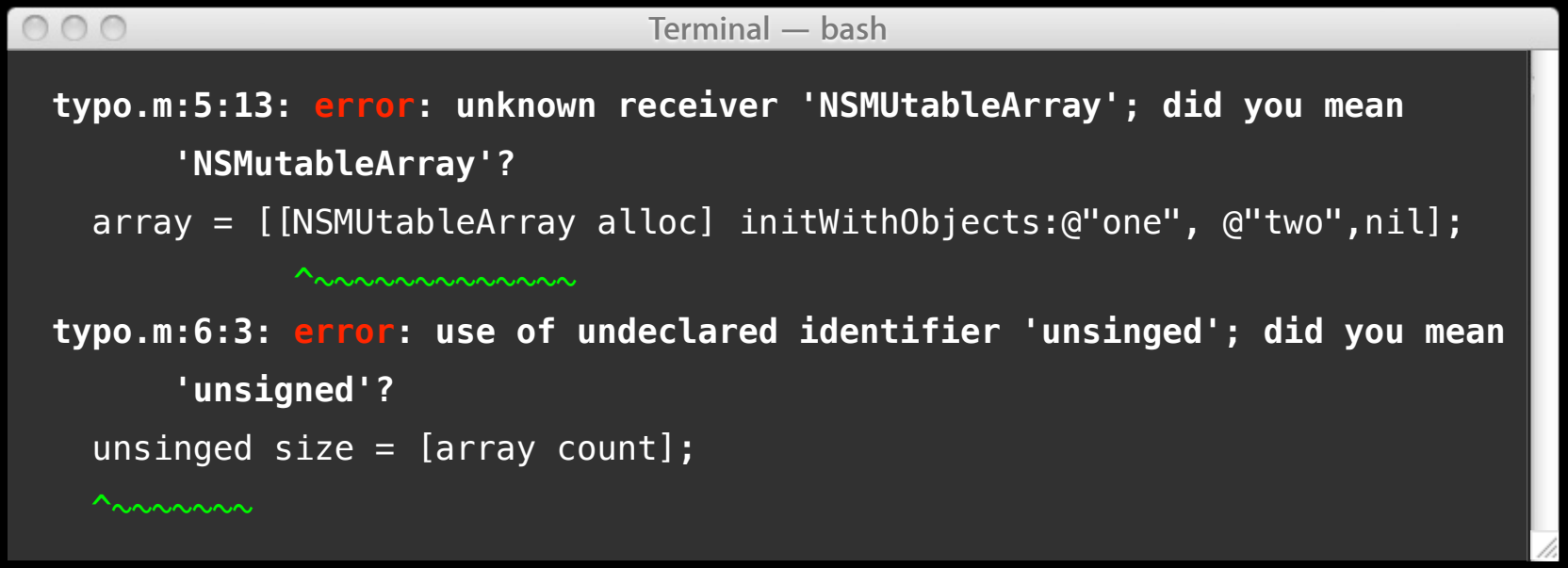
- "Diagnostics" are warnings and errors
  - Compiler finds a problem
  - Tries to diagnose why and explain it
- Clarity is key



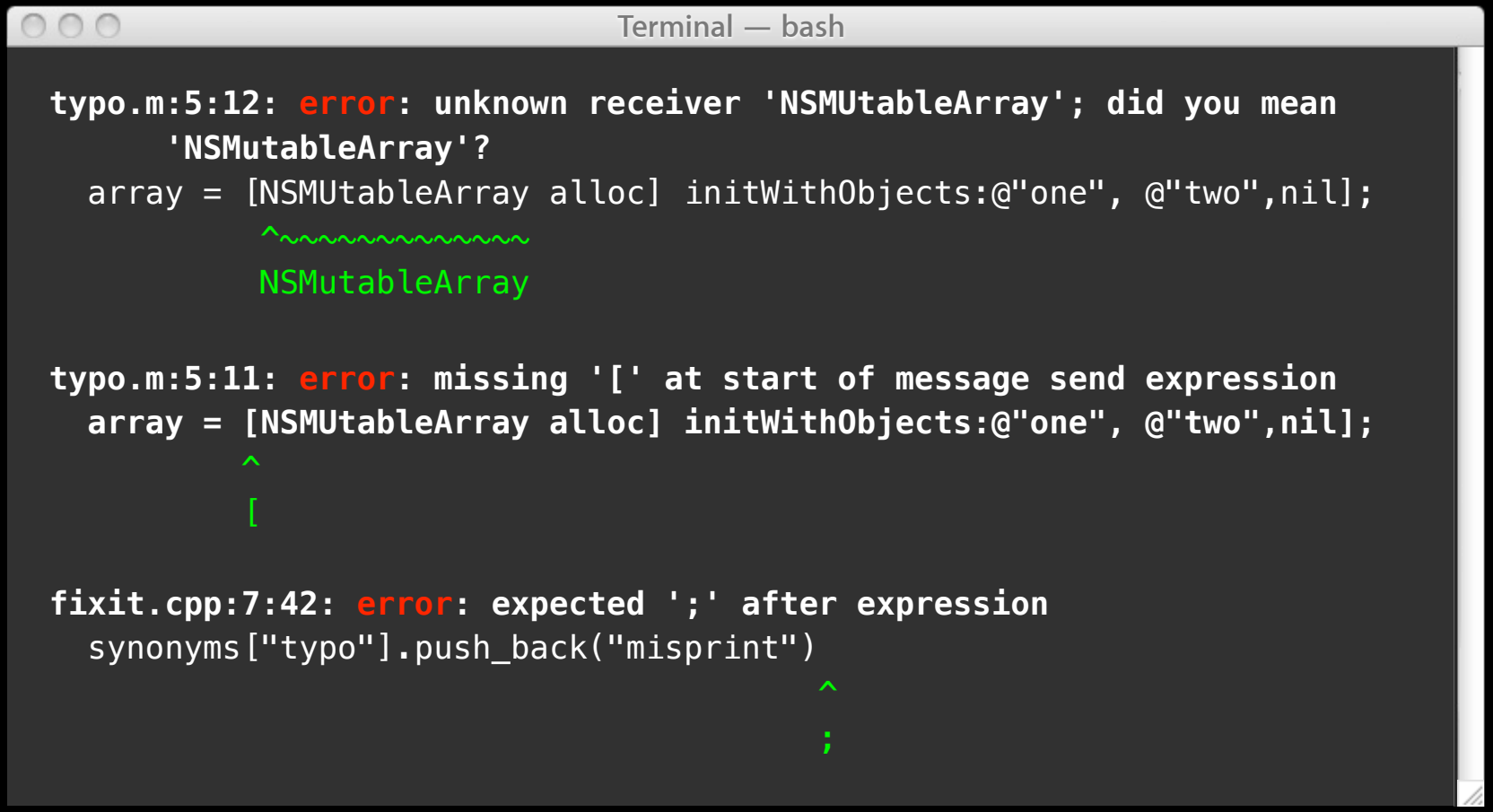
Diagnostic Ranges



Spell-Checking



Fix-its


Terminal — bash
typo.m:5:12: error: unknown receiver 'NSMUtableArray'; did you mean
'NSMutableArray'?
array = [NSMUtableArray alloc] initWithObjects:@"one", @"two",nil];
^~~~~~~~~~~~~~
NSMutableArray
typo.m:5:11: error: missing '[' at start of message send expression
array = [NSMUtableArray alloc] initWithObjects:@"one", @"two",nil];
^
[
fixit.cpp:7:42: error: expected ';' after expression
synonyms["typo"].push_back("misprint")
^
;

Uninitialized Warnings—Debug and Rele


```
- (void)updateData:(int *)data withLength:(int)length {
  int delta;
  // ...
  transform(data, length, ^(int x) { return x + delta; });
}
```

Terminal — bash

```
$ clang -c -Wuninitialized -O1 uninit.m
uninit.m:12:27: warning: variable 'delta' is uninitialized when
    captured by block [-Wuninitialized]
  transform(data, length, ^(int x) { return x + delta; });
                          ^
uninit.m:10:3: note: variable 'delta' is declared here
  int delta;
  ^
```

-Wuninitialized Is Conservative

- Only diagnose uninitialized variables when the compiler is certain

```
- (void)updateData:(int *)data withLength:(int)length {
    int delta;
    if ([self shouldTransform])
       delta = 5;
    // ...
    if ([self shouldTransform])
      transform(data, length, ^(int x) { return x + delta; });
}
```

- `-Wconditional-uninitialized` is more aggressive:

```
Terminal — bash

warning: variable 'delta' is uninitialized when captured by
    block [-Wconditional-uninitialized]
```

Inferred Result Type

Terminal — bash

```
infer.m:4:10: warning: incompatible pointer types initializing 'NSSet *'
    with an expression of type 'NSArray *' [-Wincompatible-pointer-types]

  NSSet *numbers = [[NSArray alloc] initWithObjects:@"one", @"two", nil];
         ^         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

NSArray.h:91:1: note: instance method 'initWithObjects:' is assumed to
      return an instance of its receiver type ('NSArray *')
- (id)initWithObjects:(id)firstObj, ...;
^
```

Xcode Integration

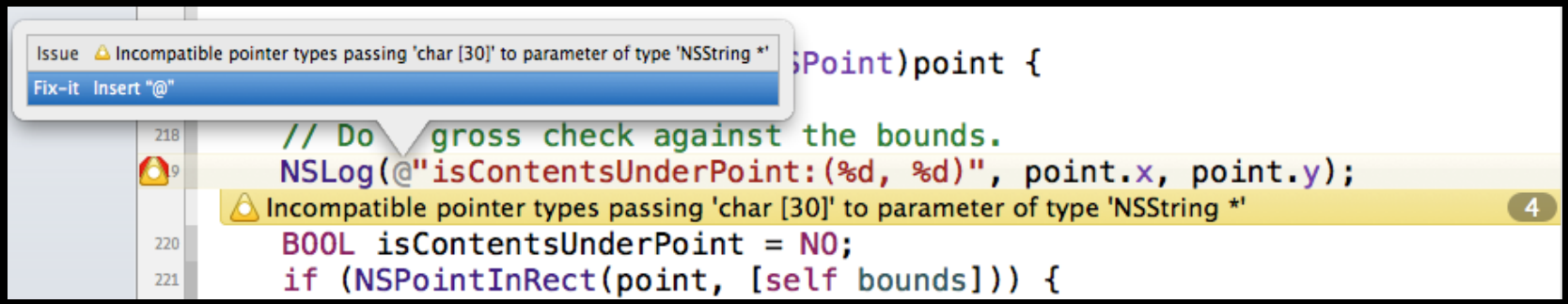
Live Warnings and Errors

- Immediate feedback from the compiler
  - Fix the issue ⟶ warning goes away
  - No remaining issues ⟶ file compiles cleanly

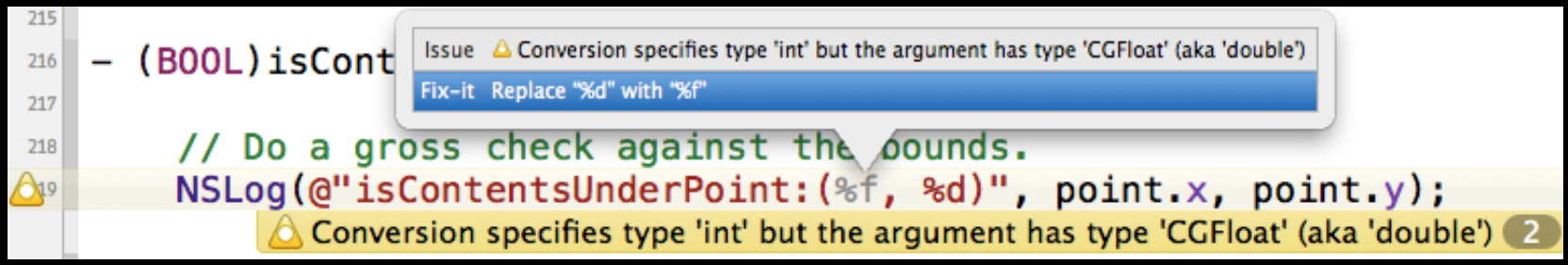
```
216 - (BOOL)isContentsUnderPoint:(NSPoint)point {
217
218     // Do a gross check against the bounds.
219     NSLog("isContentsUnderPoint:(%d, %d)", point.x, point.y);
220     BOOL isContentsUnderPoint = NO;
221     if (NSPointInRect(point, [self bounds])) {
```

Fix-its in Xcode

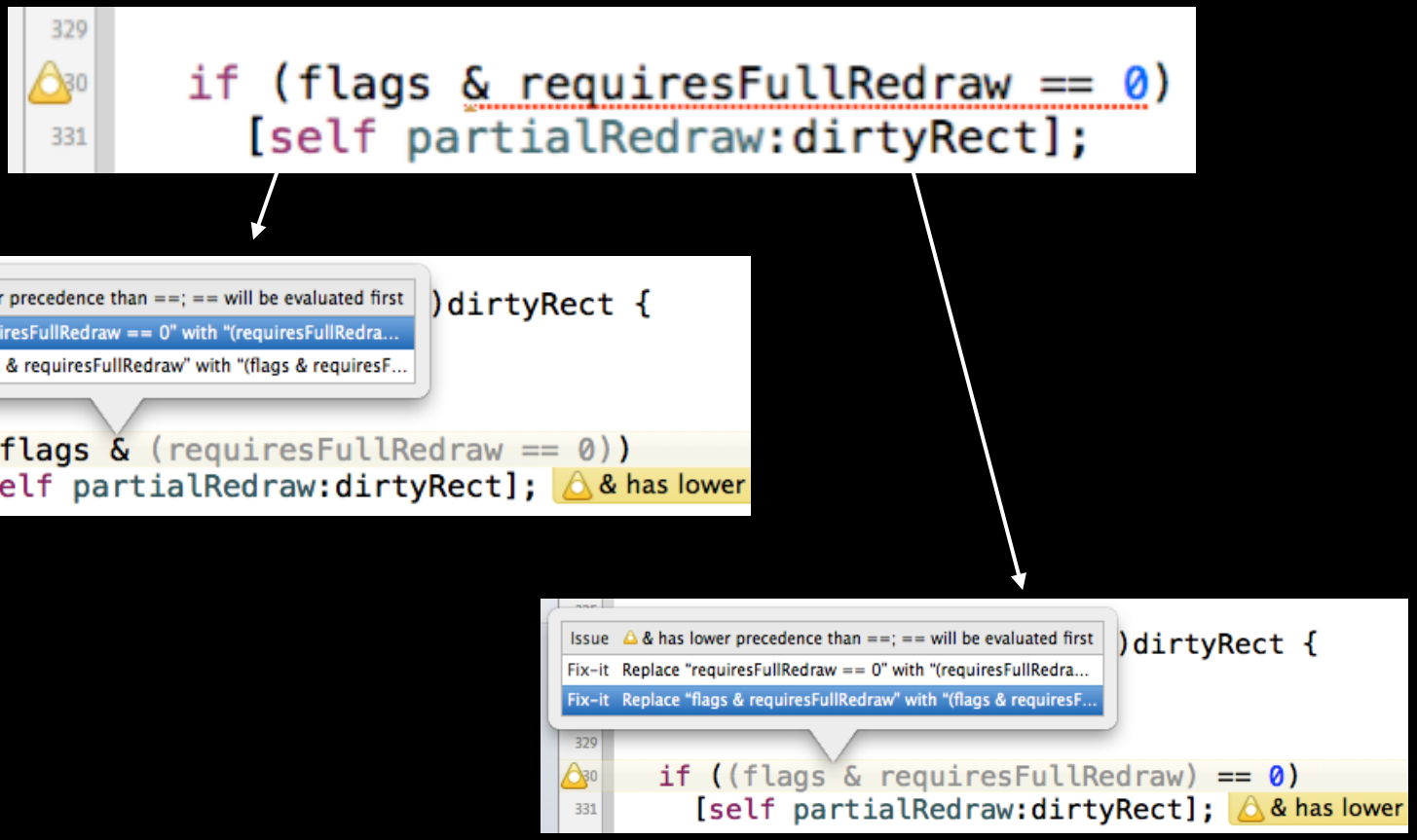
- Apply compiler's suggested fix to your source



Fix-its in Xcode

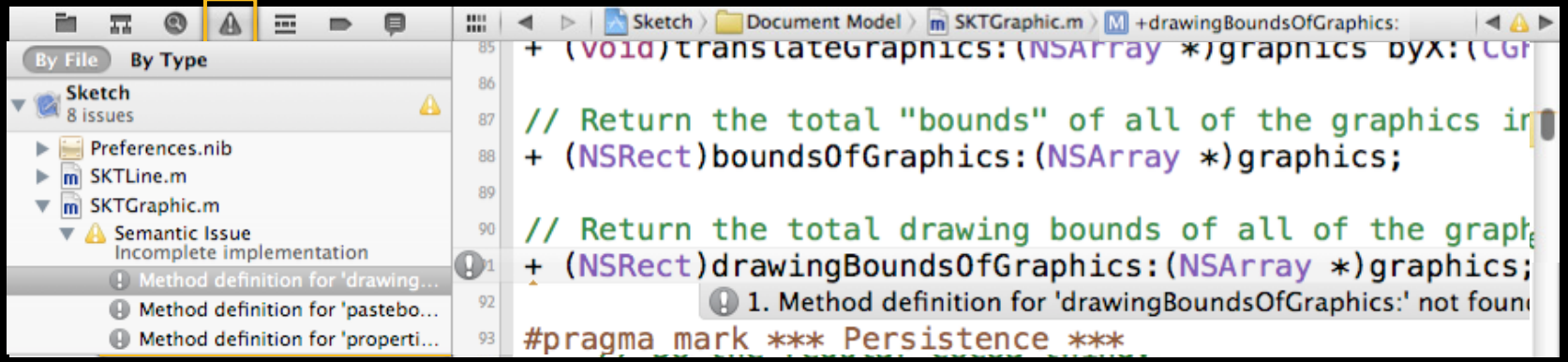


Fix-its with Options



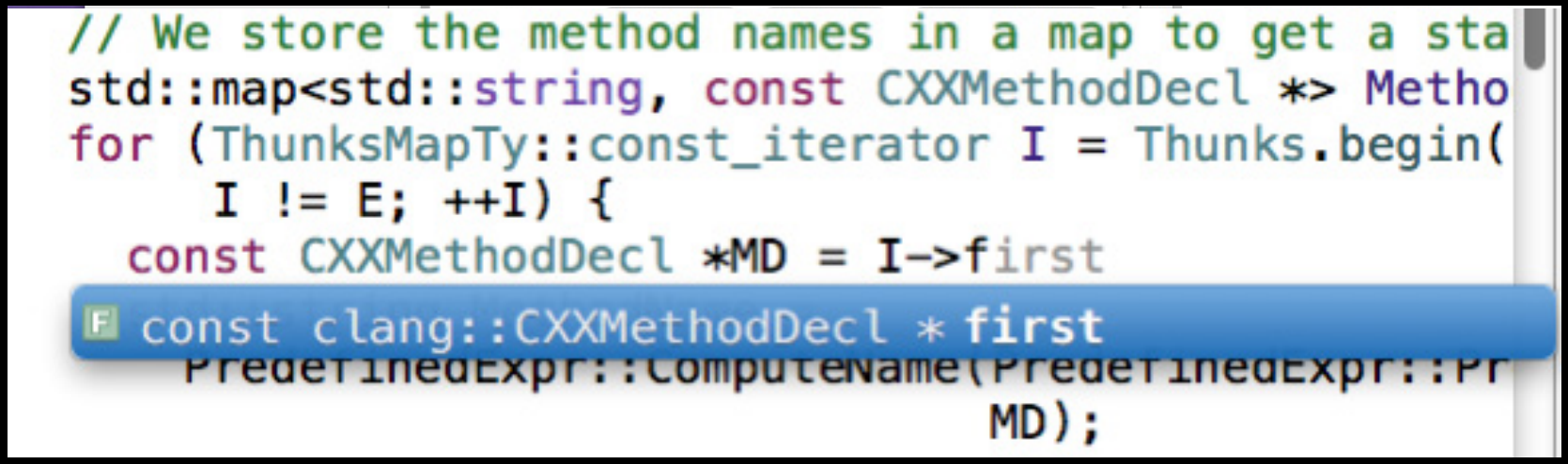
Additional Information via Notes

New



Code Completion

- Compiler understands code better than any other tool
  - More accurate completions
  - Knowledge of the full language

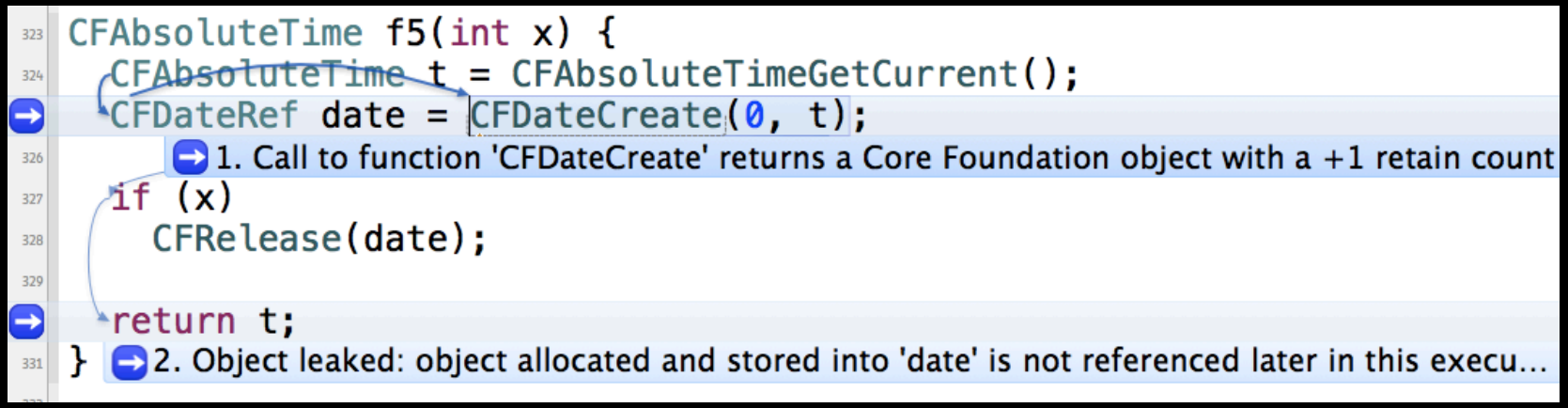


Xcode Static Analyzer

What Is It?

- Deep compiler analysis to find bugs
  - Reasons about all "paths" in the program
  - Finds bugs that depend on branches and specific execution paths
- Examples
  - Logic errors such as null dereferences
  - Misuses of APIs (Core Foundation, Cocoa, and UNIX)
  - Memory leaks

Example: Core Foundation Leaks



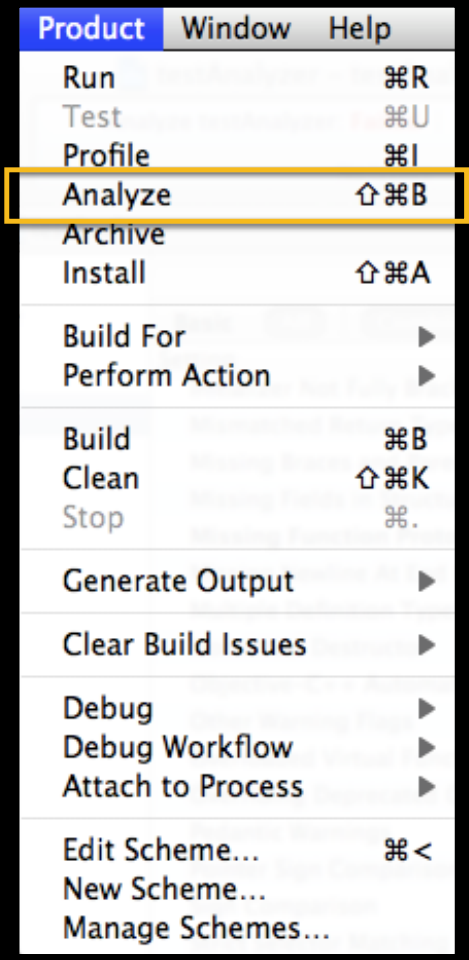
Example: Bugs Involving Loops

```
void test_array() {
  int vals[3];
  for (unsigned i = 0 ; i < 3 ; ++i) {
    vals[i] = i+1;
  }
  // ...
  // Do stuff.
  // ...
  for (unsigned i = 0; i <= 3 ; ++i) {
    vals[i] = vals[i] * 2;
  }
}
```

1. Looping back to the head of the loop 3
4. Looping back to the he... 3
7. The left operand of '*' is a garbage value

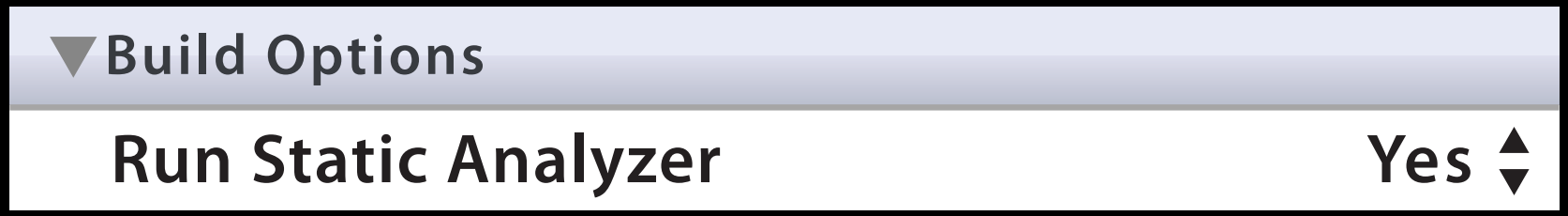
Workflow in Xcode

- Special "build" with extra analysis
- Trade off CPU time for finding more bugs
- Issues appear within…
  - Editor
  - Issue Navigator



Controlling the Static Analyzer

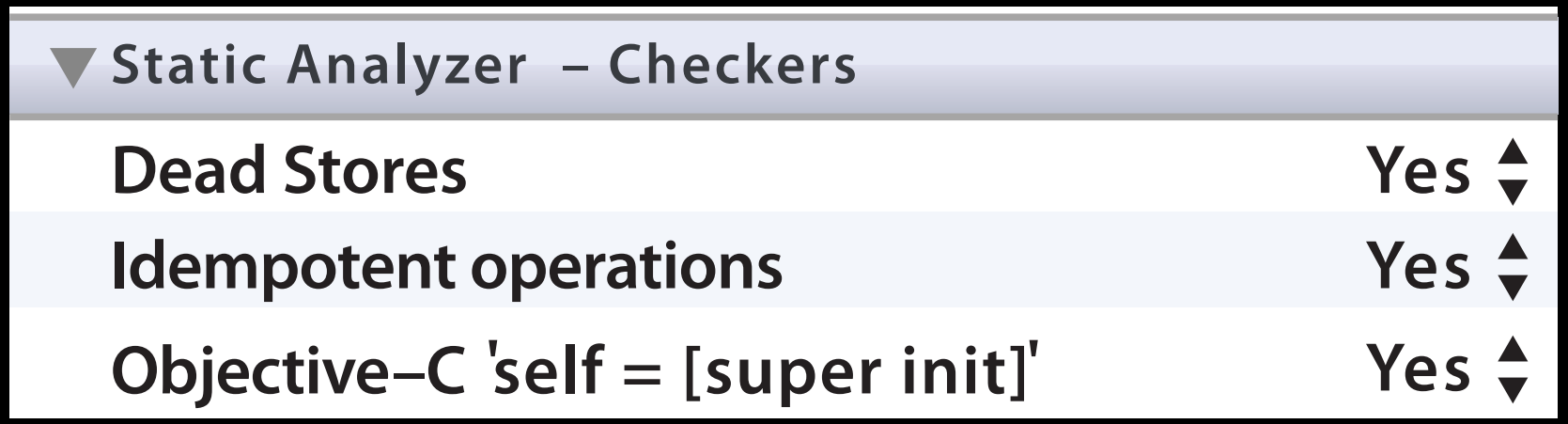
Always running the analyzer when doing a build



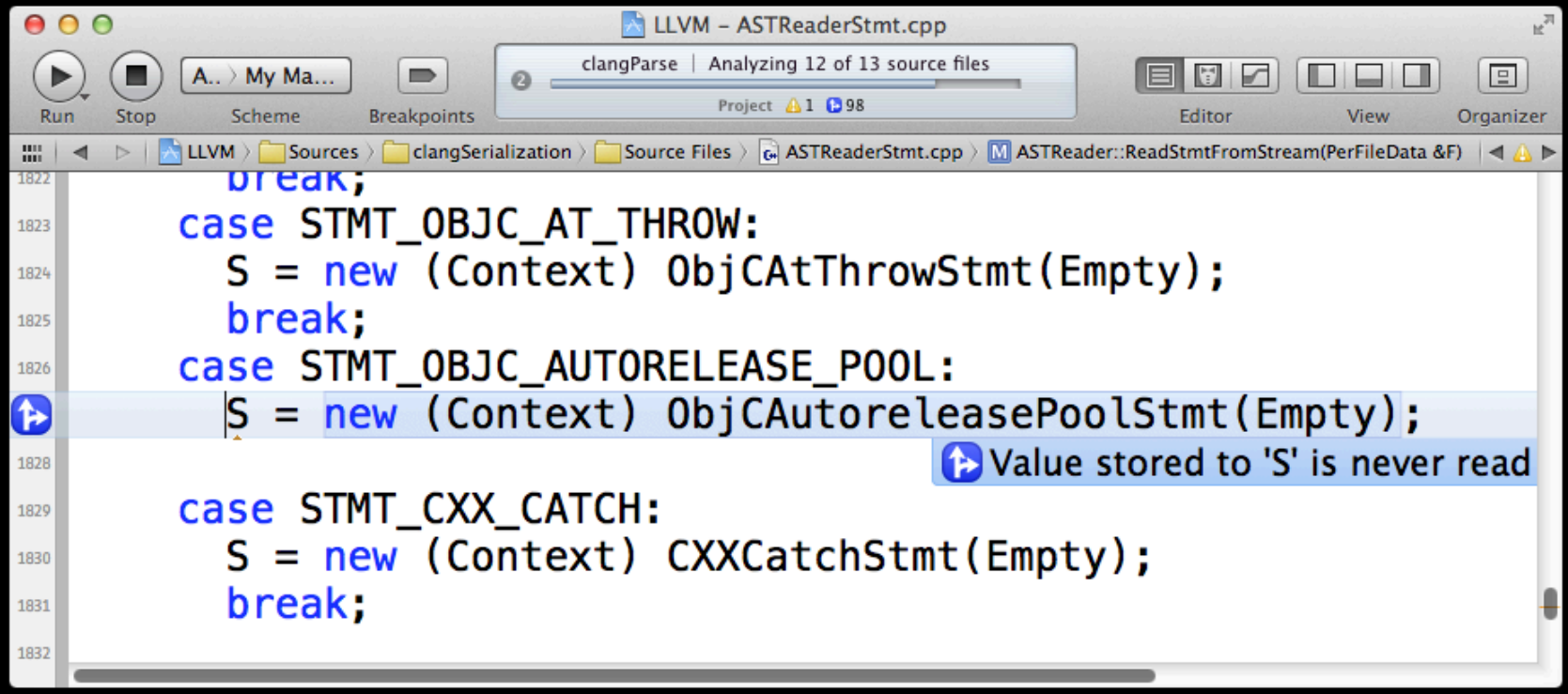
Controlling the Static Analyzer

Enabling specific checkers

New



Now Supports C++!

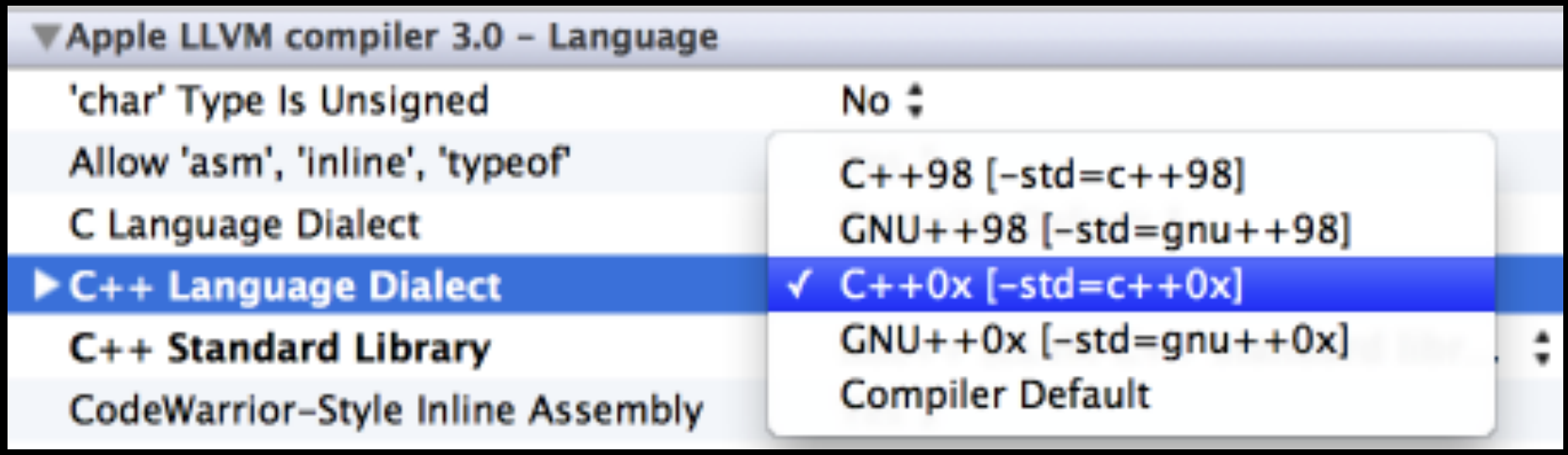


New C++ Features

C++0x

- Upcoming ISO C++ standard
- Major changes to both language and library
  - Many new language features
  - Expanded, improved C++ standard library
- Maintains backward compatibility with C++98/03

C++0x Available in Apple LLVM Compiler 3



C++0x in Apple LLVM Compiler 3

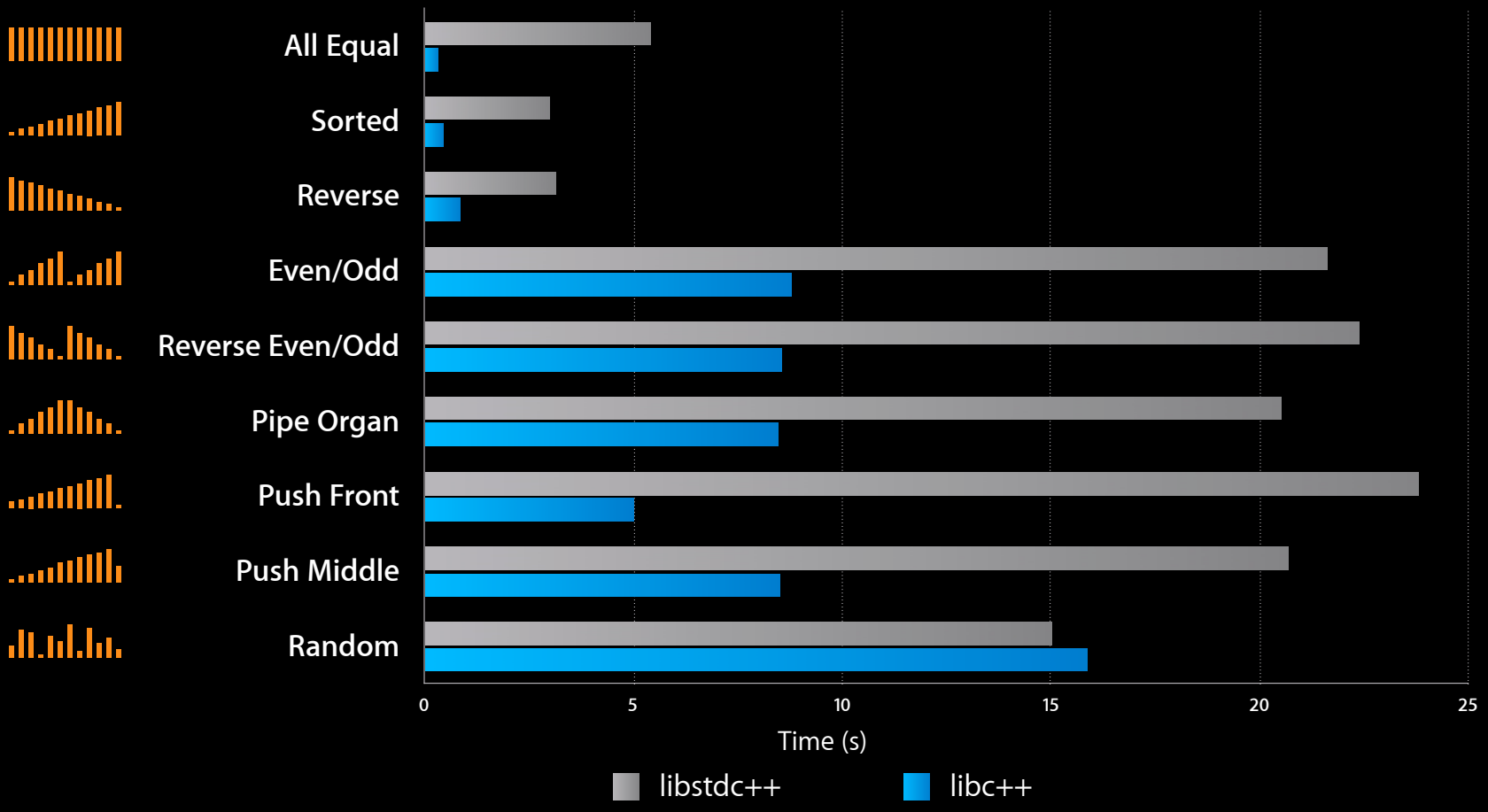
- Type inference with ‘auto’
- Range-based for loop
- Rvalue references (move semantics)
- Variadic templates
- Null pointer constant
- Override controls
- Strongly typed enums
- Static assertions
- Extended SFINAE
- Deleted functions
- Extern templates
- Inline namespaces
- decltype
- noexcept

LLVM Technologies In-Depth	Pacific Heights Thursday 2:00PM

libc++: LLVM C++ Standard Library

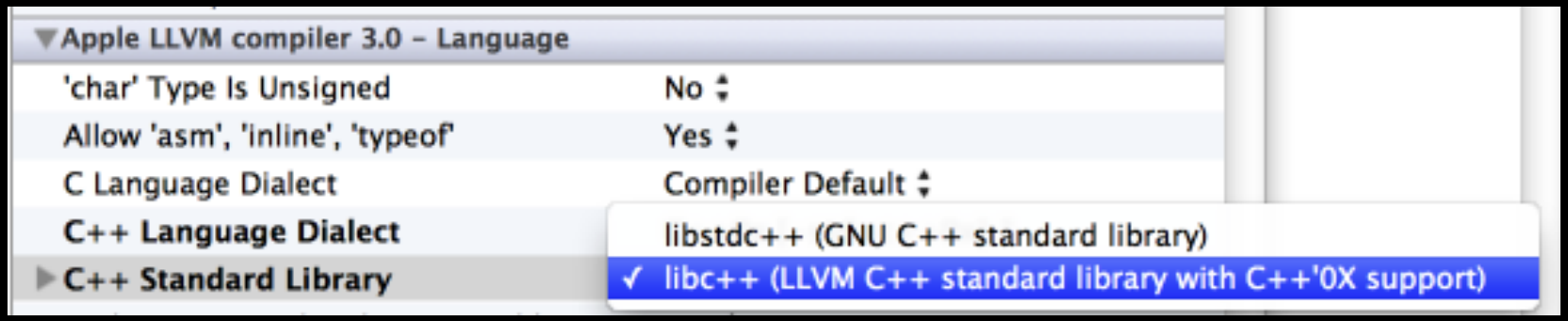
- Re-engineered from the ground up for C++0x
  - Complete support for C++0x language features
  - Better data structures and algorithms
  - New functionality (regular expressions, smart pointers, hash tables)
- Open source!
  - http://libcxx.llvm.org/

Pattern Recognition in `sort()`



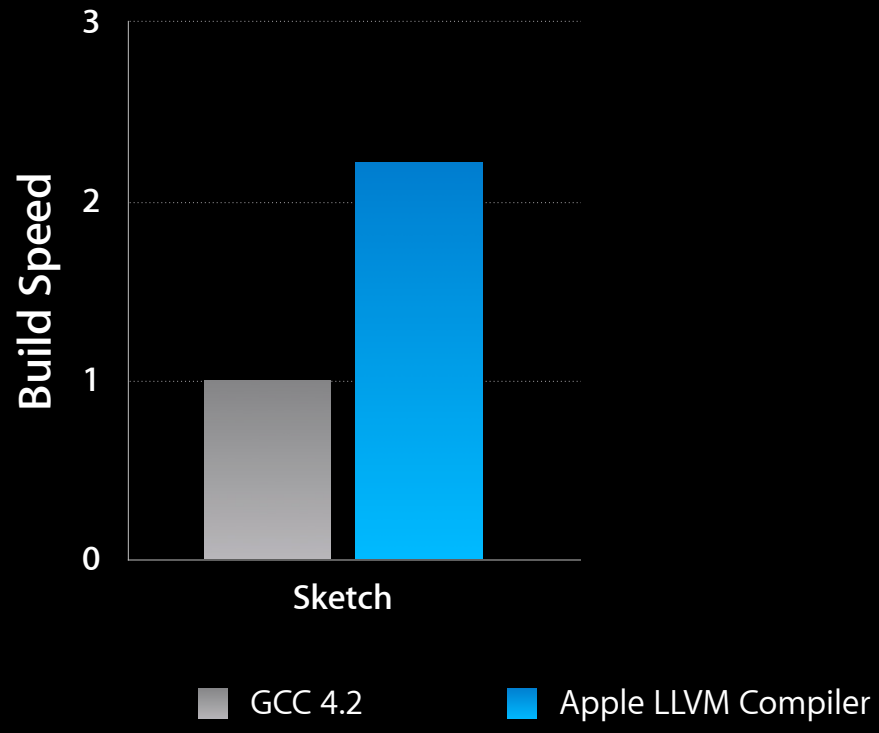
libc++ Available in Xcode 4.2

- Deploys to Lion and iOS 5

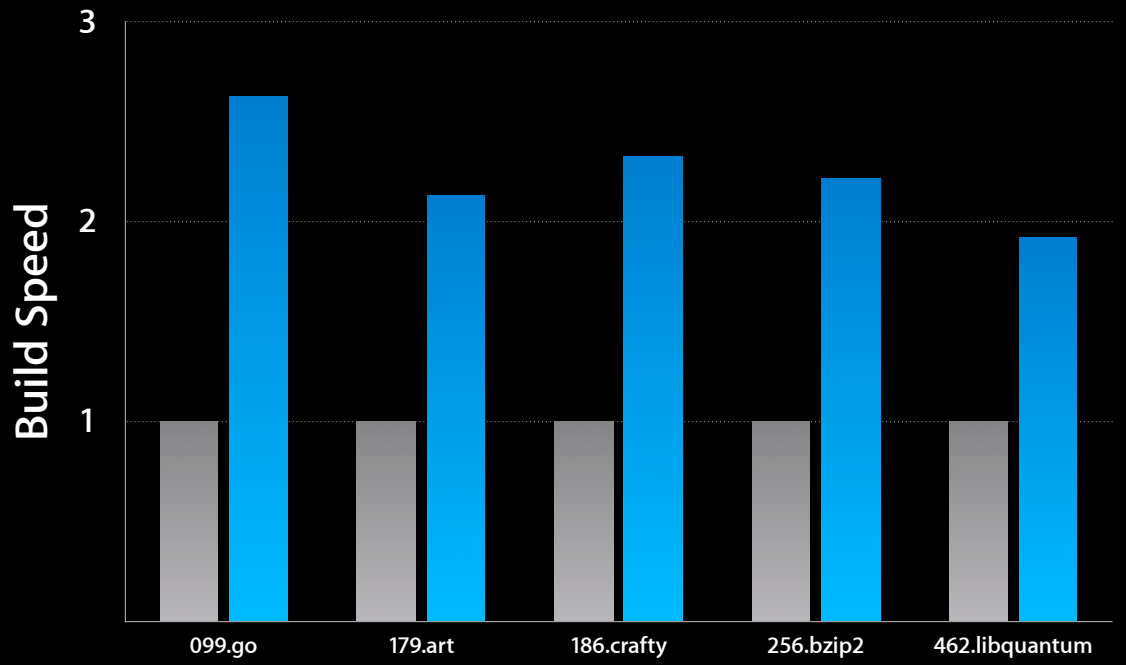


Compiler Performance

Fast Debug Builds

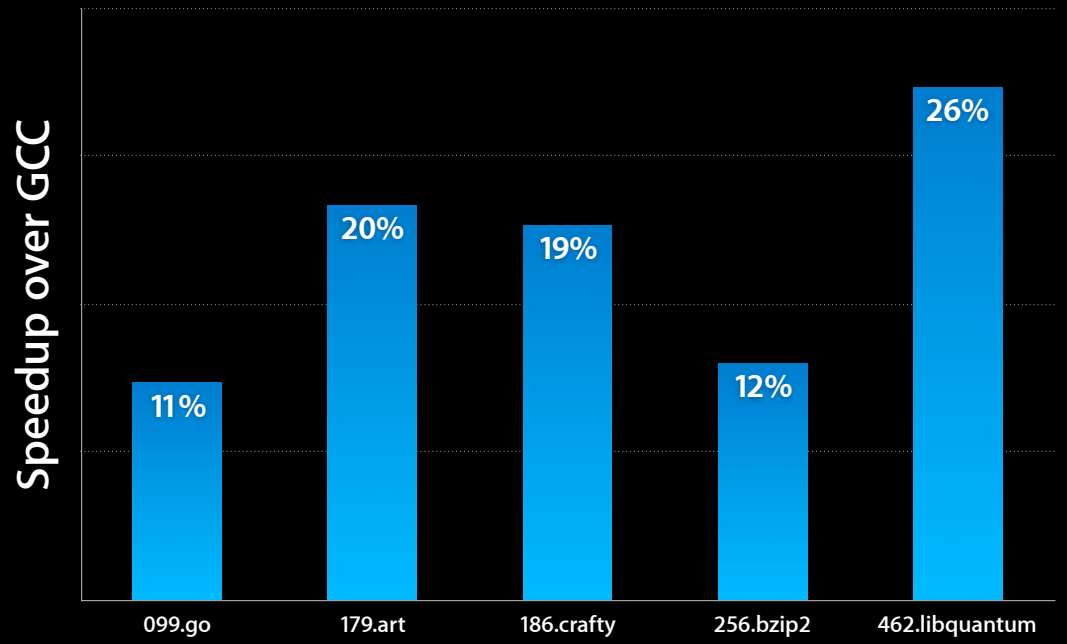


Fast Release Builds



2–3 times faster!

Great Optimization



Vectorizing iOS Code

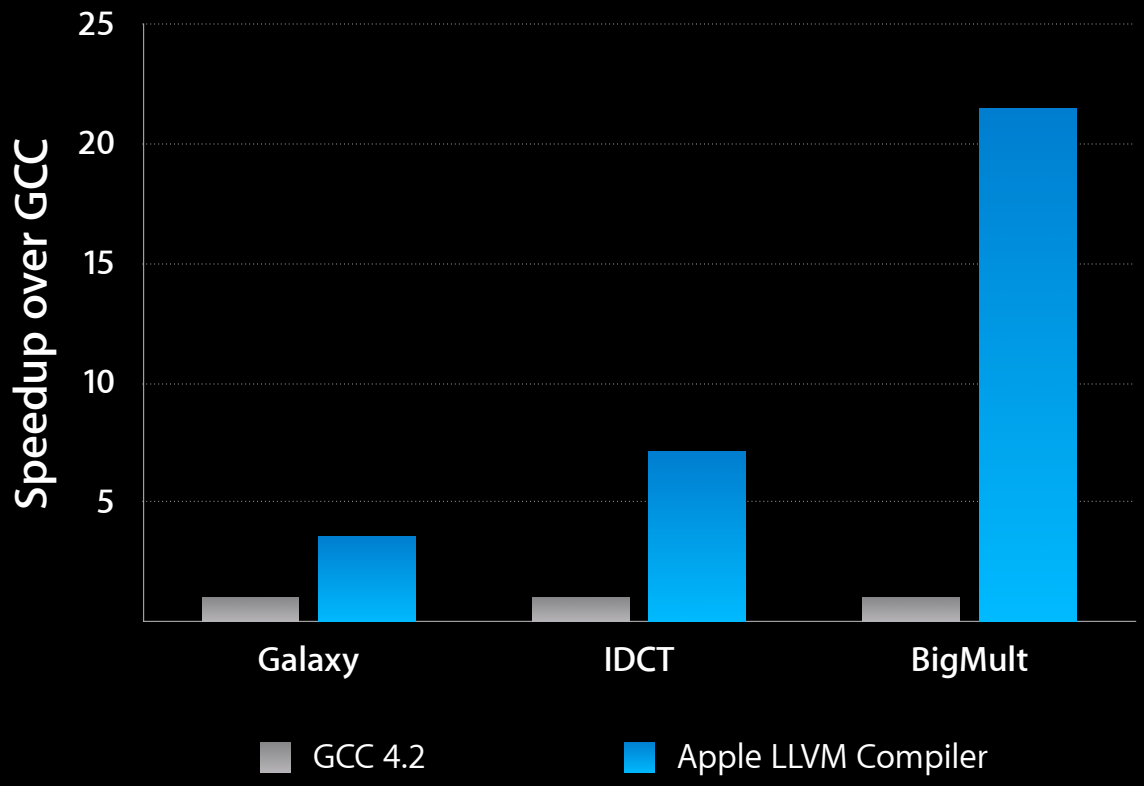
- Built-in vector types and operators

```
int32x4_t accumulate (int32x4_t *x, int32x4_t *y) {
  int32x4_t vecsum = { 0, 0, 0, 0 };
  for (int n = 0; n < 100; ++n)
    vecsum += x[n] * y[n];
  return vecsum;
}
```

- Intrinsics for more complex operations

```
t9 = vqrdmulhq_s16 (a1, vx[1]);
t8 = vqsubq_s16 (t9, vx[7]);
t1 = vqaddq_s16 (vqrdmulhq_s16 (a1, vx[7]), vx[1]);
```

Vector Performance

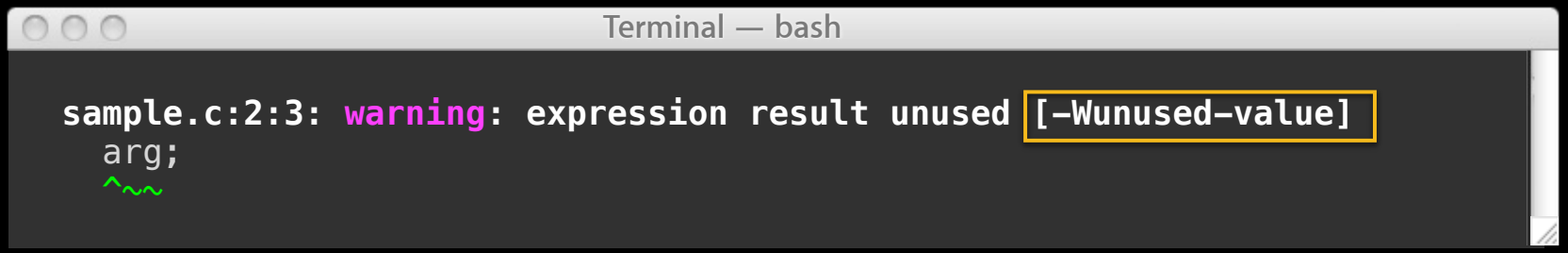


Transition Tips

Additional Warnings

- Apple LLVM Compiler provides more warnings

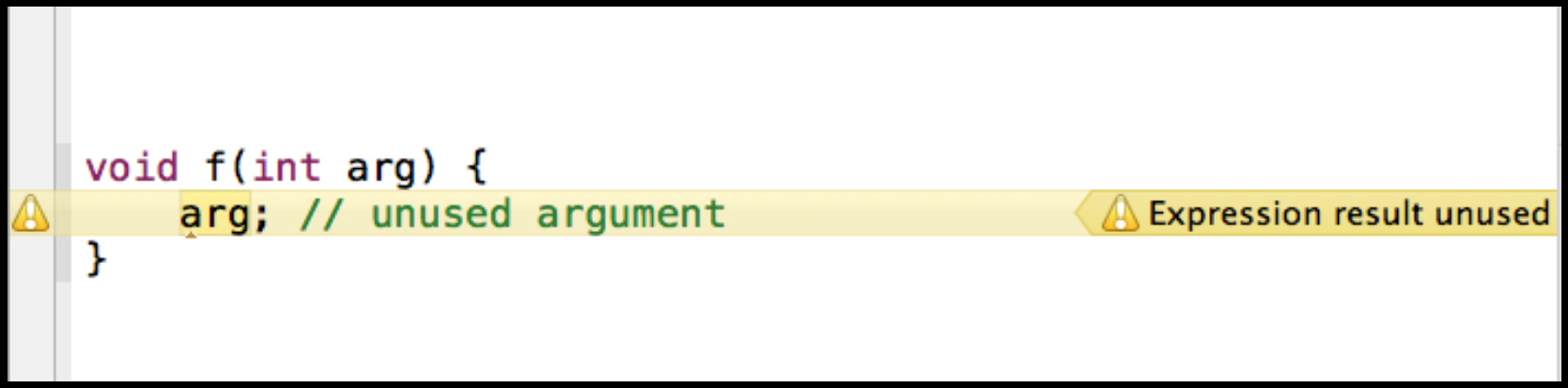
```
void f(int arg) {
  (void)arg; // unused argument
}
```



- Suppress the warning
  - Add `-Wno-unused-value` to your build settings

Selectively Disable Warnings

- Use clang pragmas for more fine-grained control



Selectively Disable Warnings

- Use clang pragmas for more fine-grained control

```
#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wunused-value"

void f(int arg) {
    arg; // unused argument
}

#pragma clang diagnostic pop
```

Fast Forward to 1999

- GCC defaults to C89
- Apple LLVM Compiler defaults to C99
- Inline function semantics

```
inline double min(double x, double y) {
  return x < y ? x : y;
}
```

Terminal — bash

```
Undefined symbols for architecture x86_64:
  "_min", referenced from:
      _main in cc-kr2ME5.o
ld: symbol(s) not found for architecture x86_64
```

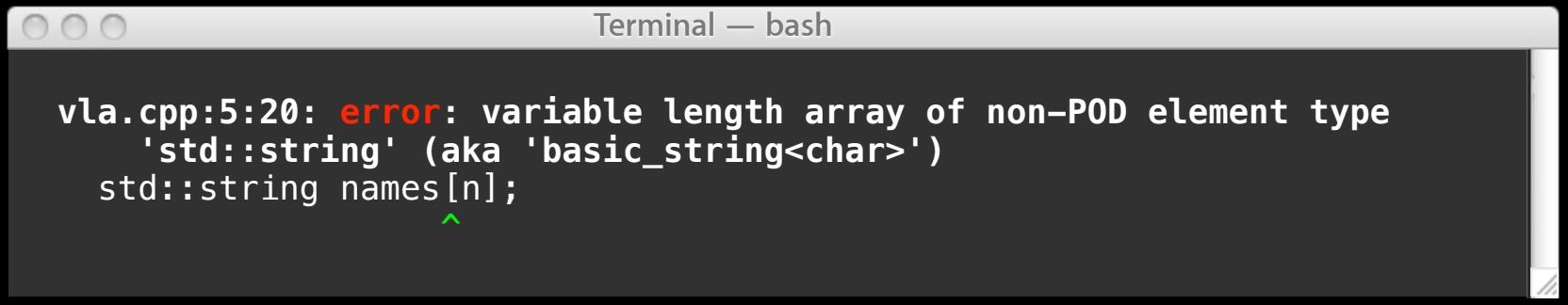
Variable-Length Arrays in C++

- Allowed for Plain Old Data (POD) types

```
int array[n];
```

- Disallowed for non-POD types

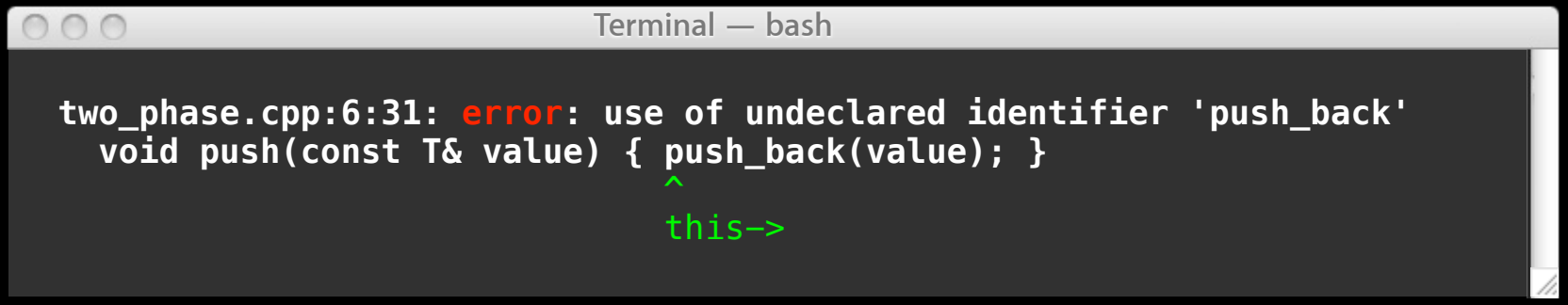
```
std::string array[n];
```



Name Lookup Semantics in Templates

```
template<typename T, typename Container>
class Stack : Container {
public:
  void push(const T& value) { push_back(value); }
};

Stack<int, std::vector<int> > integer_stack;
```



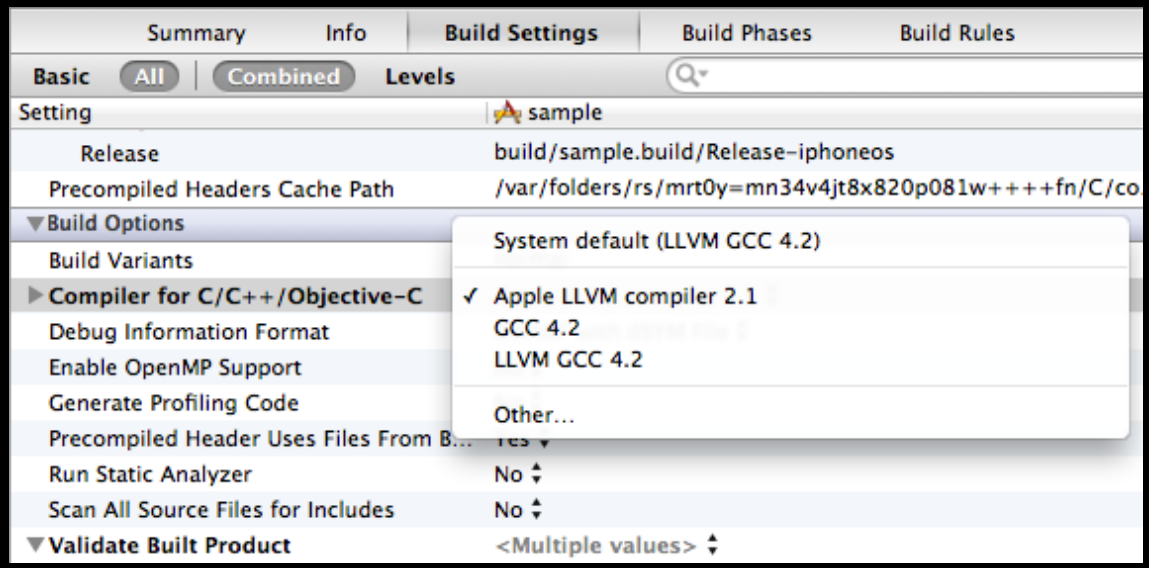
Moving to Automatic Reference Counting

- First clean up warnings in non-ARC mode
- Some Objective-C warnings have become errors
- ARC formalizes best practices

Objective-C Advancements In–Depth	Mission Friday 11:30AM

2011 LLVM Releases

Selecting a Compiler in Xcode 4.1



Compilers in Xcode 4.1

Apple LLVM Compiler version 2.1

- Bug fix release
- New `-Wuninitialized` option
- Better C++ standard conformance

Compilers in Xcode 4.2

Apple LLVM Compiler 3.0

- Automatic Reference Counting (ARC)
- gcov code coverage support
- C++0x
- Better code generation

LLVM Technologies In-Depth	Pacific Heights Thursday 2:00PM

Compiler Release Summary

Two great Xcode releases

Xcode 4.1	Xcode 4.2
GCC 4.2	GCC 4.2
LLVM-GCC	LLVM-GCC
LLVM Compiler 2.1	LLVM Compiler 3.0

It's Time to Move to LLVM!

- GCC is going away
- Apple LLVM Compiler is the future
  - Helps you find bugs
  - Great performance
  - New features
- Switch your apps now!

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

LLVM Project
Open-Source LLVM Project Home
http://llvm.org

Apple Bug Reporter
http://developer.apple.com/bugreporter

Apple Developer Forums
http://devforums.apple.com

Related Sessions

LLVM Technologies In-Depth	Pacific Heights Thursday 2:00PM
Objective-C Advancements In–Depth	Mission Friday 11:30AM

Labs

LLVM Lab	Developer Tools Lab A Wednesday 2:00PM
Objective–C and Automatic Reference Counting Lab	Developer Tools Lab B Thursday 2:00PM


WWDC2011