

# Moving to the Apple LLVM Compiler

Session 307

**Bob Wilson**

Manager, LLVM Core Team

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

# 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



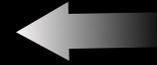
LLVM Back End

# 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

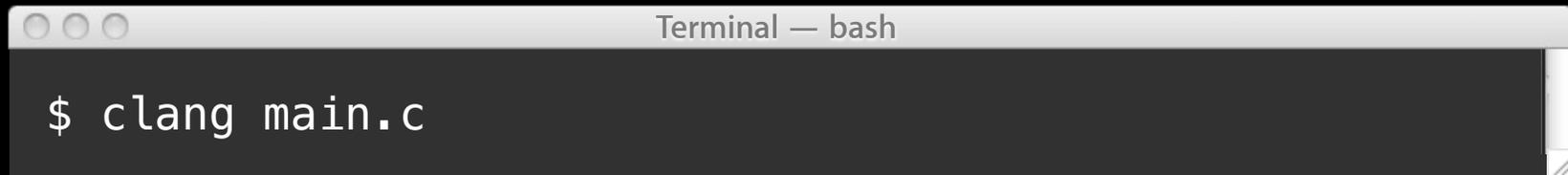
# Apple LLVM Compiler 3 Features

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

# Feature Highlights

**Doug Gregor**  
Clang Technical Lead

# Compiler “User Interface”

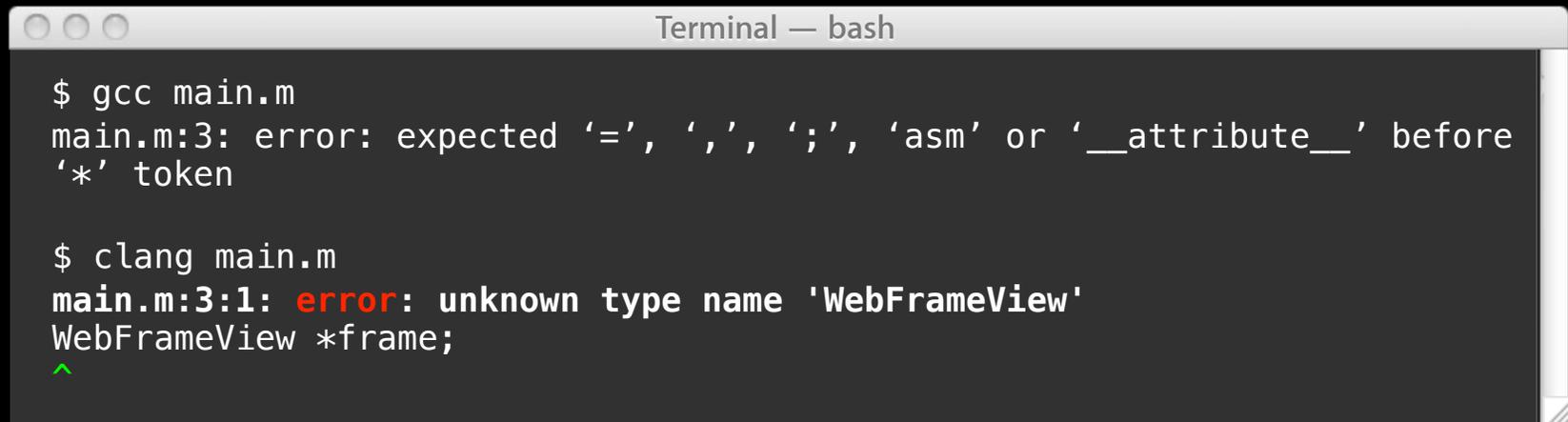
A terminal window titled "Terminal — bash" with three window control buttons (red, yellow, green) in the top-left corner. The terminal content shows a shell prompt "\$" followed by the command "clang main.c".

```
Terminal — bash
$ clang main.c
```

- 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



```
Terminal — bash
$ gcc main.m
main.m:3: error: expected '=', ',', ';', 'asm' or '__attribute__' before
'*' token

$ clang main.m
main.m:3:1: error: unknown type name 'WebFrameView'
WebFrameView *frame;
^
```

# Diagnostic Ranges

```
Terminal — bash

SKTLine.m:17:34: error: invalid operands to binary expression
('NSPoint' (aka 'struct _NSPoint') and 'CGFloat' (aka 'float')) [3]
    if (xDelta==0.0f && fabs(point - beginPoint.x)<=acceptableDistance) {
                        ~~~~~ ^ ~~~~~

compare.c:6:64: error: expression is not assignable
    if (oldLoc.x == newLoc.x && oldLoc.y == newLoc.y && oldLoc.z = newLoc.z)
                        ~~~~~~ ^
```

# Spell-Checking

```
Terminal — bash

typo.m:5:13: error: unknown receiver 'NSMutableArray'; did you mean
    'NSMutableArray'?
    array = [[NSMutableArray alloc] initWithObjects:@"one", @"two",nil];
              ^~~~~~

typo.m:6:3: error: use of undeclared identifier 'unsinged'; did you mean
    'unsigned'?
    unsinged size = [array count];
    ^~~~~~
```

# Fix-its

```
Terminal — bash

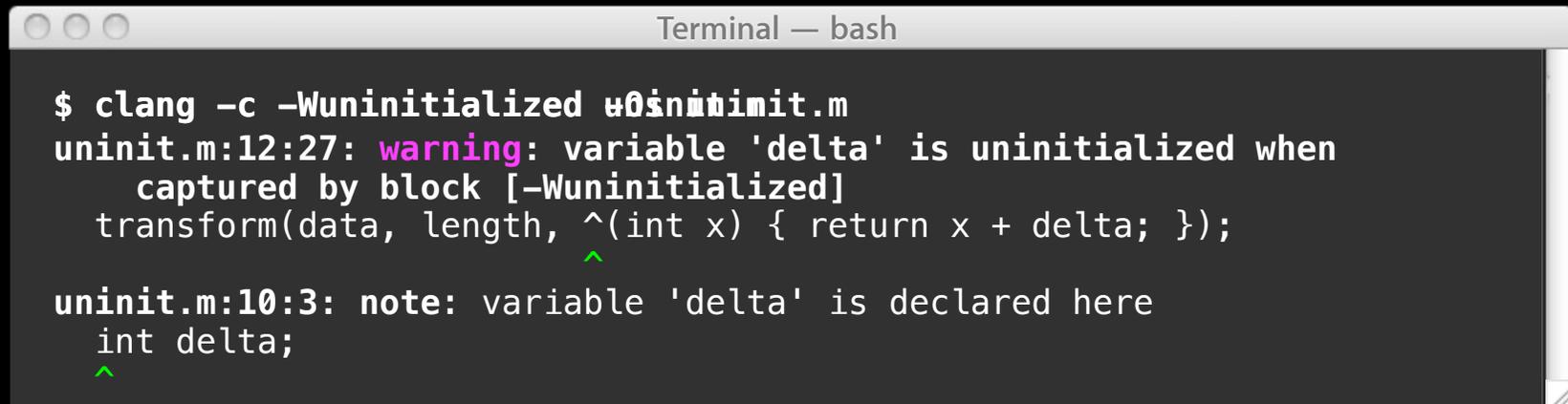
typo.m:5:12: error: unknown receiver 'NSMutableArray'; did you mean
      'NSMutableArray'?
    array = [NSMutableArray alloc] initWithObjects:@"one", @"two",nil];
              ~~~~~
              NSMutableArray

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 Release

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



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

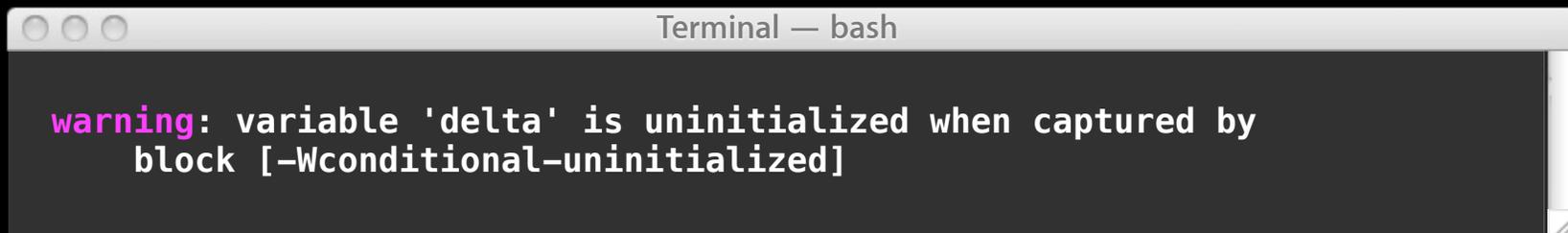
# -Wuninitialized Is Conservative

New

- 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

New

```
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

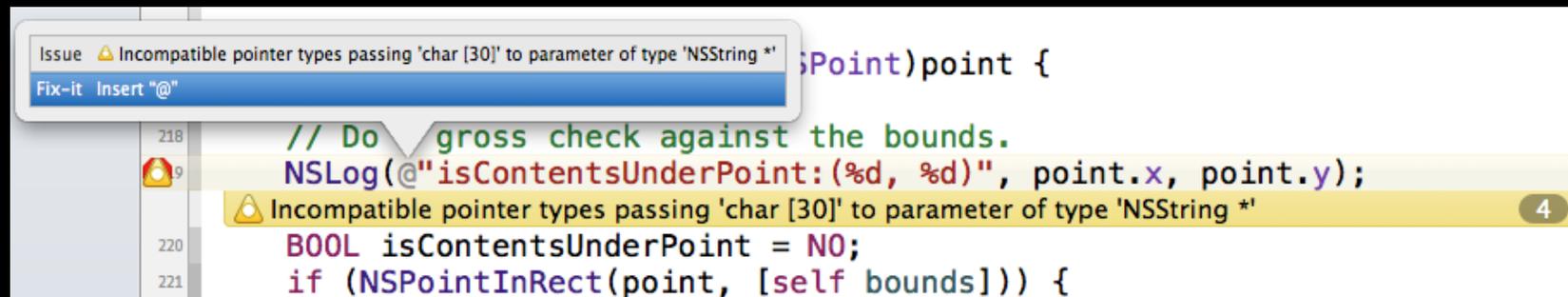
# Live Warnings and Errors

- 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



The screenshot shows a code editor with a warning message and a suggested fix. The warning message is: "Issue ⚠ Incompatible pointer types passing 'char [30]' to parameter of type 'NSString \*'". The suggested fix is: "Fix-it Insert '@'". The code being edited is:

```
218 // Do 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

```
215  
216 - (BOOL)isCont  
217  
218 // Do a gross check against the bounds.  
219 NSLog(@"isContentsUnderPoint:(%f, %d)", point.x, point.y);  
      Conversion specifies type 'int' but the argument has type 'CGFloat' (aka 'double') 2
```

Issue ⚠ Conversion specifies type 'int' but the argument has type 'CGFloat' (aka 'double')

Fix-it Replace "%d" with "%f"

# Fix-its with Options

```
329  
330 if (flags & requiresFullRedraw == 0)  
331 [self partialRedraw:dirtyRect];
```

Issue ⚠ & has lower precedence than ==; == will be evaluated first  
Fix-it Replace "requiresFullRedraw == 0" with "(requiresFullRedra...  
Fix-it Replace "flags & requiresFullRedraw" with "(flags & requiresF...

```
329  
330 if (flags & (requiresFullRedraw == 0))  
331 [self partialRedraw:dirtyRect]; ⚠ & has lower
```

Issue ⚠ & has lower precedence than ==; == will be evaluated first  
Fix-it Replace "requiresFullRedraw == 0" with "(requiresFullRedra...  
Fix-it Replace "flags & requiresFullRedraw" with "(flags & requiresF...

```
329  
330 if ((flags & requiresFullRedraw) == 0)  
331 [self partialRedraw:dirtyRect]; ⚠ & has lower
```

# Additional Information via Notes



The screenshot shows the Xcode IDE interface. On the left, the Project Navigator displays a project named "Sketch" with 8 issues. Under the "SKTGraphic.m" file, a "Semantic Issue" is highlighted, indicating an "Incomplete implementation" for the method definition of 'drawingBoundsOfGraphics:'. The main editor window shows the Swift code for the SKTGraphic.m file, with the following code visible:

```
85 + (void)translateGraphics:(NSArray *)graphics byX:(CGFloat)dx;
86
87 // Return the total "bounds" of all of the graphics in the array
88 + (NSRect)boundsOfGraphics:(NSArray *)graphics;
89
90 // Return the total drawing bounds of all of the graphics in the array
91 + (NSRect)drawingBoundsOfGraphics:(NSArray *)graphics;
92
93 #pragma mark *** Persistence ***
```

A tooltip is visible over the method signature on line 91, displaying the message: "1. Method definition for 'drawingBoundsOfGraphics:' not found".

# Code Completion

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

```
// We store the method names in a map to get a sta
std::map<std::string, const CXXMethodDecl *> Metho
for (ThunksMapTy::const_iterator I = Thunks.begin(
    I != E; ++I) {
    const CXXMethodDecl *MD = I->first
    clang::CXXMethodDecl *first
    PredefinedExpr::computeName(PredefinedExpr::Pr
        MD);
```

# 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

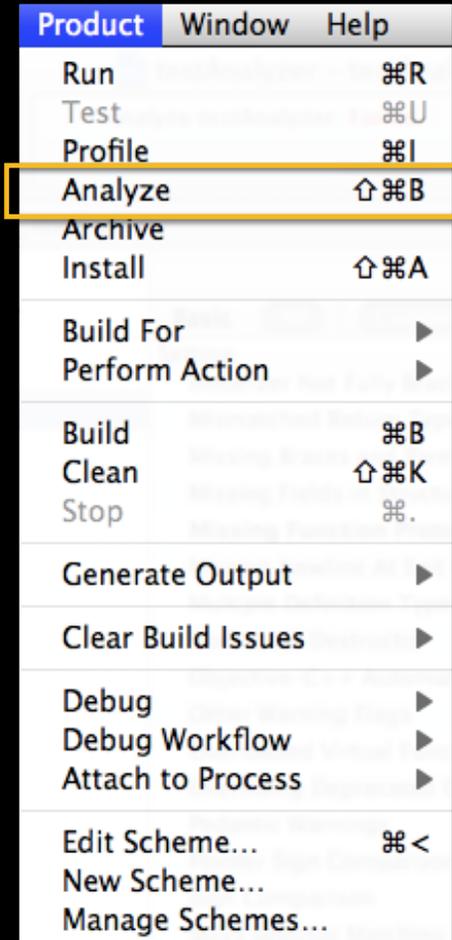
```
323 CFAbsoluteTime f5(int x) {  
324     CFbsoluteTime t = CFAbsoluteTimeGetCurrent();  
325     CFDateRef date = CFDateCreate(0, t);  
326     1. Call to function 'CFDateCreate' returns a Core Foundation object with a +1 retain count  
327     if (x)  
328         CFRelease(date);  
329     return t;  
330 }  
331 2. Object leaked: object allocated and stored into 'date' is not referenced later in this execu...
```

# Example: Bugs Involving Loops

```
void test_array() {  
    int vals[3];  
    for (unsigned i = 0 ; i < 3 ; ++i) {  
        vals[i] = i+1; → 1. Looping back to the head of the loop 3  
    }  
    // ...  
    // Do stuff.  
    // ...  
    for (unsigned i = 0 ; i <= 3 ; ++i) {  
        vals[i] = vals[i] * 2; → 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

▼ Build Options

Run Static Analyzer

Yes ▲▼

# Controlling the Static Analyzer

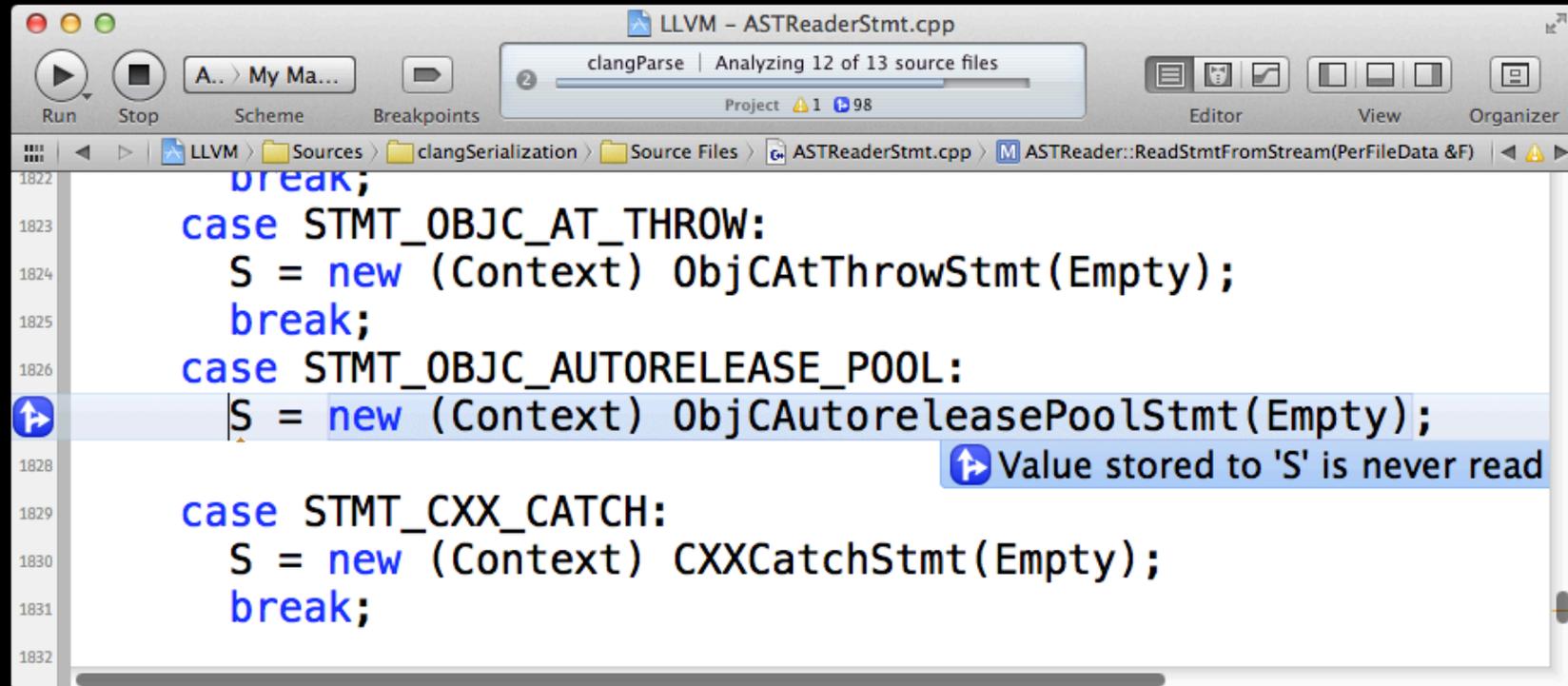
Enabling specific checkers



▼ Static Analyzer – Checkers	
Dead Stores	Yes ▲▼
Idempotent operations	Yes ▲▼
Objective-C 'self = [super init]'	Yes ▲▼

# Now Supports C++!

New



The screenshot shows an IDE window titled "LLVM - ASTReaderStmt.cpp". The interface includes a toolbar with "Run" and "Stop" buttons, a "Scheme" dropdown, and "Breakpoints" management. A progress bar indicates "clangParse | Analyzing 12 of 13 source files". The breadcrumb path is "LLVM > Sources > clangSerialization > Source Files > ASTReaderStmt.cpp > ASTReader::ReadStmtFromStream(PerFileData &F)". The code editor displays the following C++ code:

```
1822     break;
1823     case STMT_OBJC_AT_THROW:
1824         S = new (Context) ObjCAtThrowStmt(Empty);
1825         break;
1826     case STMT_OBJC_AUTORELEASE_POOL:
1827         S = new (Context) ObjCAutoreleasePoolStmt(Empty);
1828
1829     case STMT_CXX_CATCH:
1830         S = new (Context) CXXCatchStmt(Empty);
1831         break;
1832
```

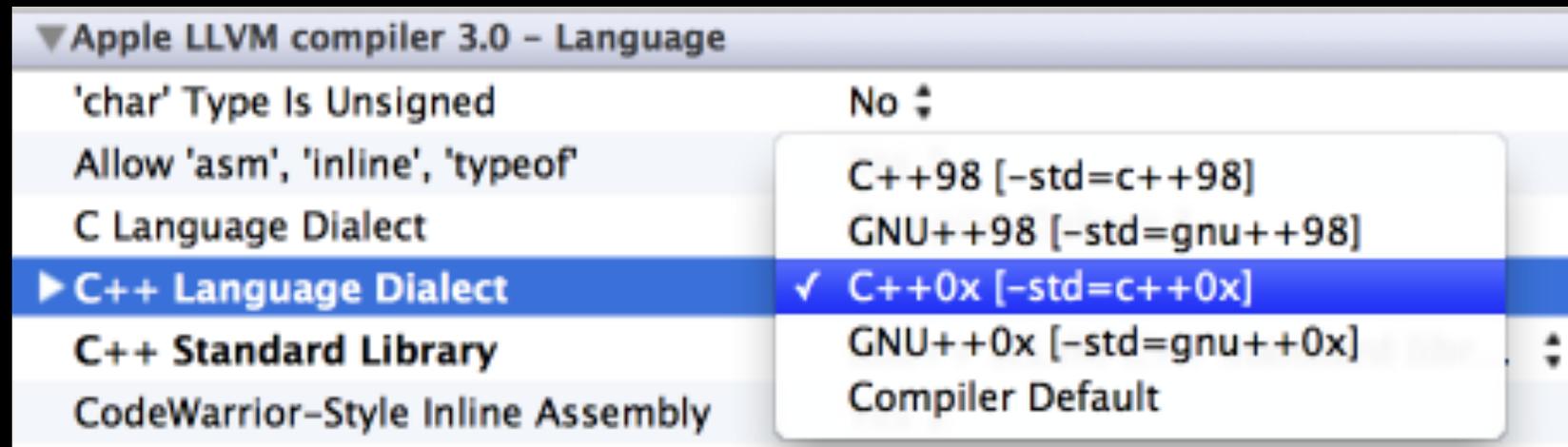
A blue tooltip points to the line `S = new (Context) ObjCAutoreleasePoolStmt(Empty);` with the message "Value stored to 'S' is never read".

# 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

New

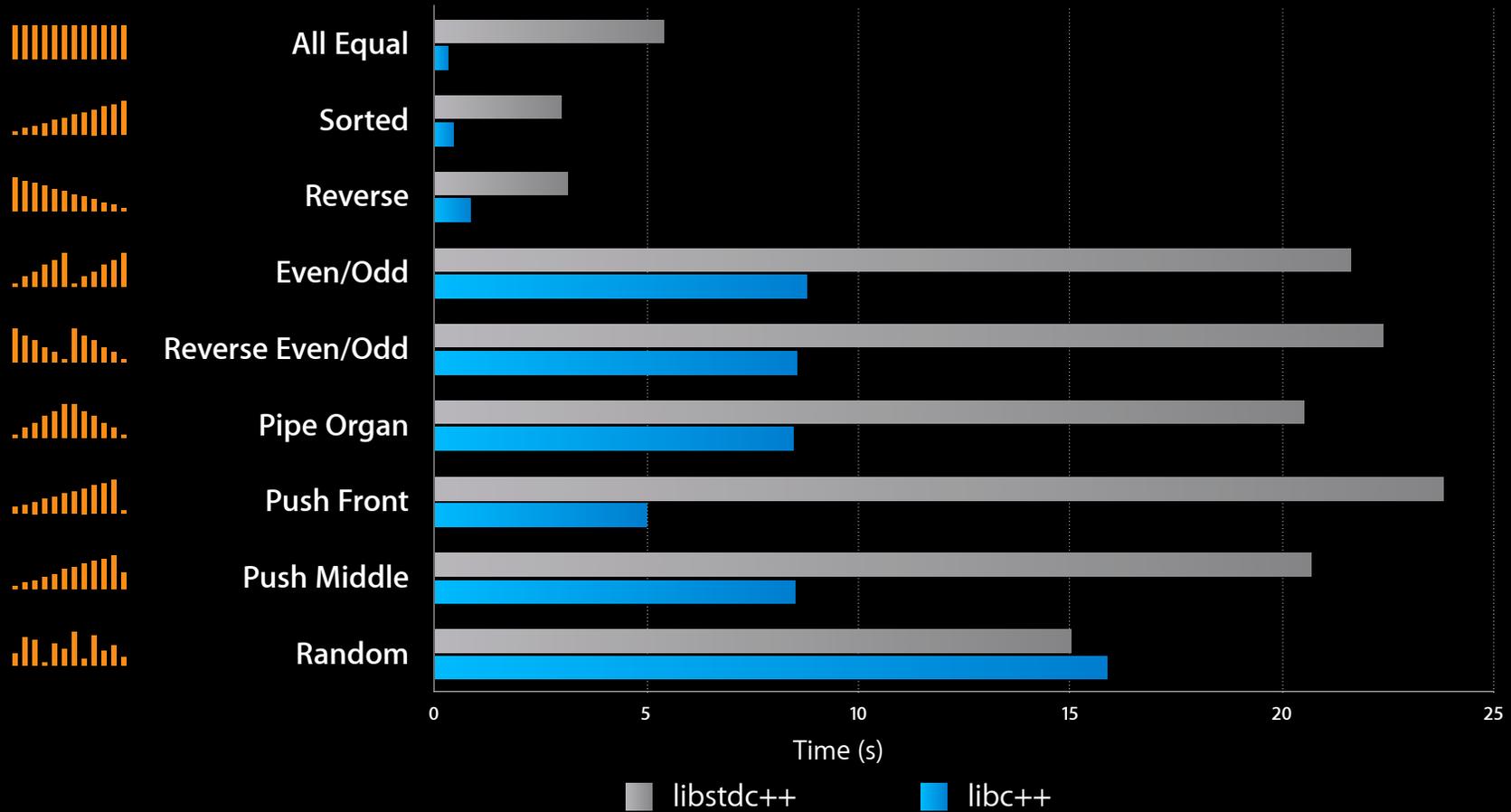
- Type inference with 'auto'
- Range-based for loop
- Rvalue references (move semantics)
- Variadic templates
- Null pointer constant
- Override controls
- Strongly typed enums
- Static assertions
- Extended SFINAE
- Deleted functions
- Extern templates
- Inline namespaces
- decltype
- noexcept

# libc++: LLVM C++ Standard Library



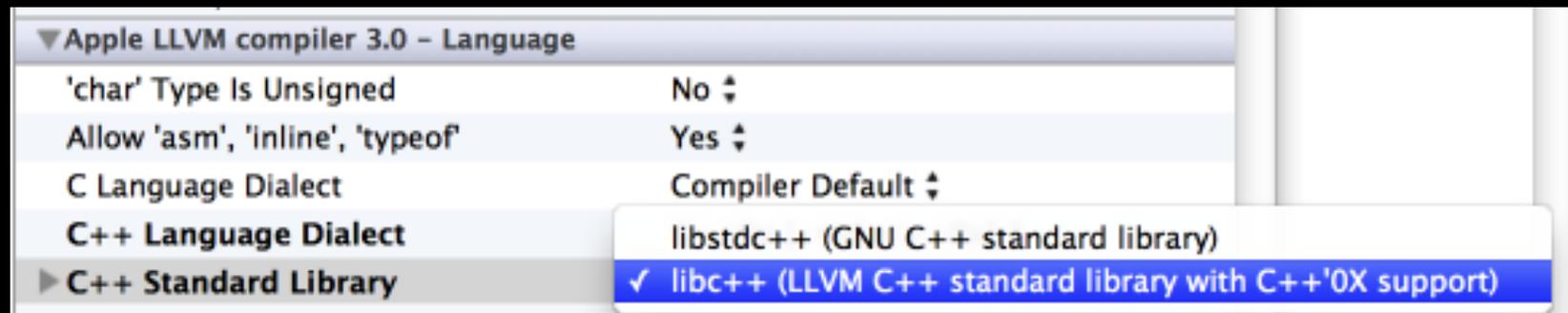
- Re-engineered from the ground up for C++0x
  - Complete support for C++0x language features
  - Better data structures and algorithms
  - New functionality (regular expressions, smart pointers, hash tables)
- Open source!
  - <http://libcxx.llvm.org/>

# Pattern Recognition in sort ( )



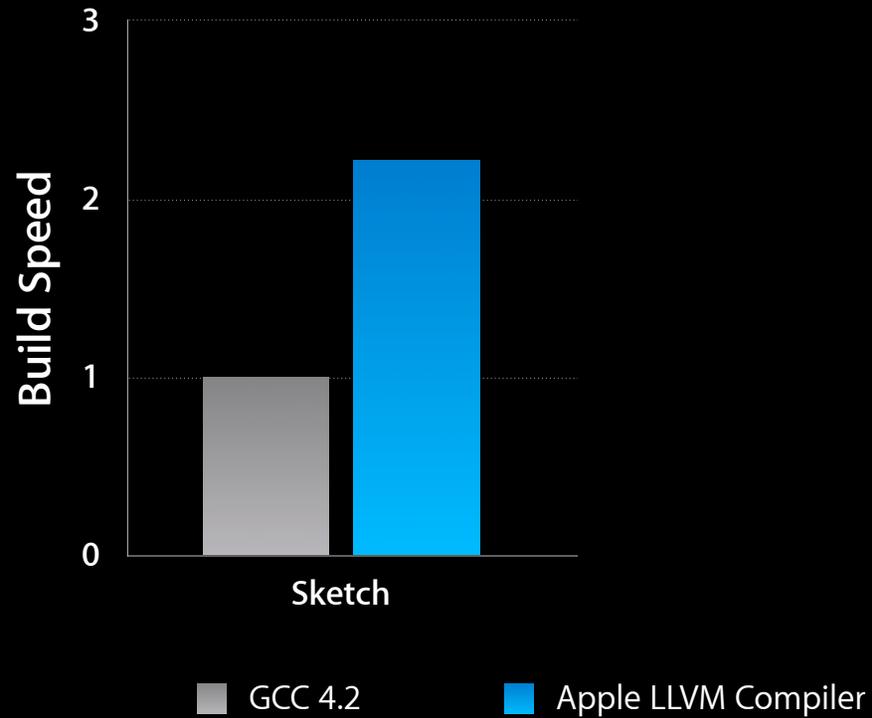
# libc++ Available in Xcode 4.2

- Deploys to Lion and iOS 5

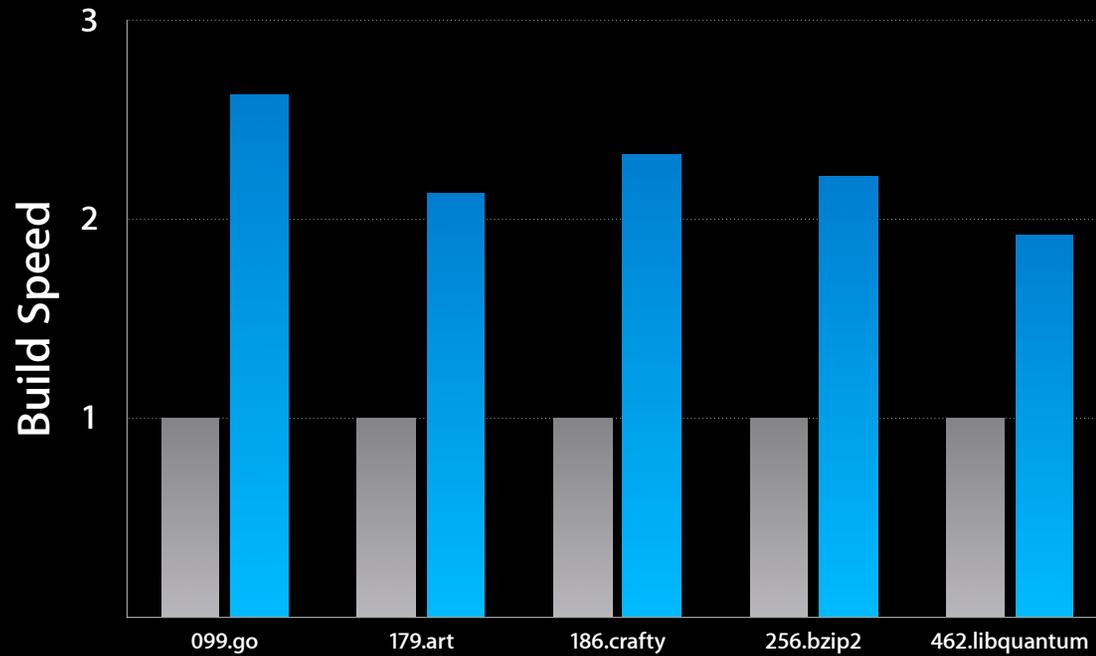


# Compiler Performance

# Fast Debug Builds

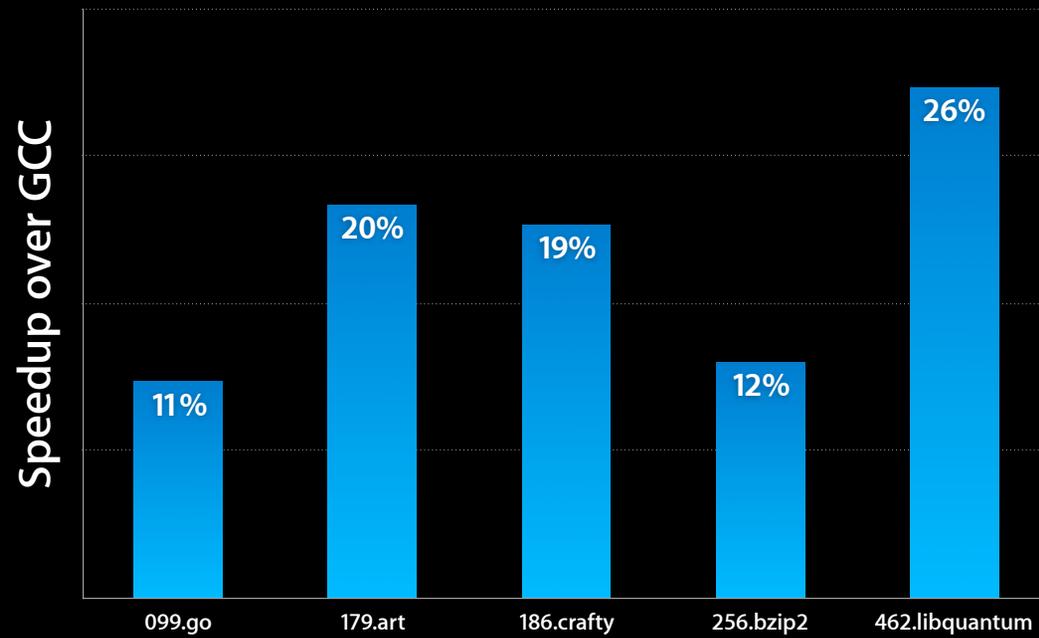


# Fast Release Builds



**2-3 times faster!**

# Great Optimization



# Vectorizing iOS Code

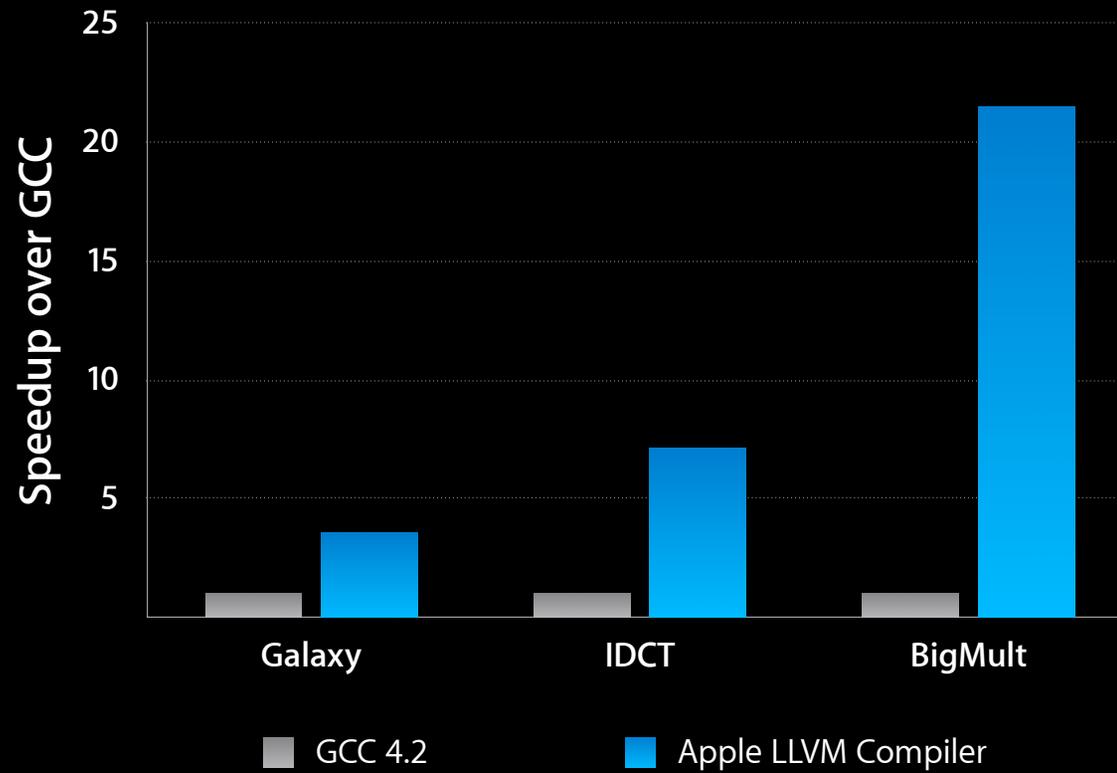
- Built-in vector types and operators

```
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

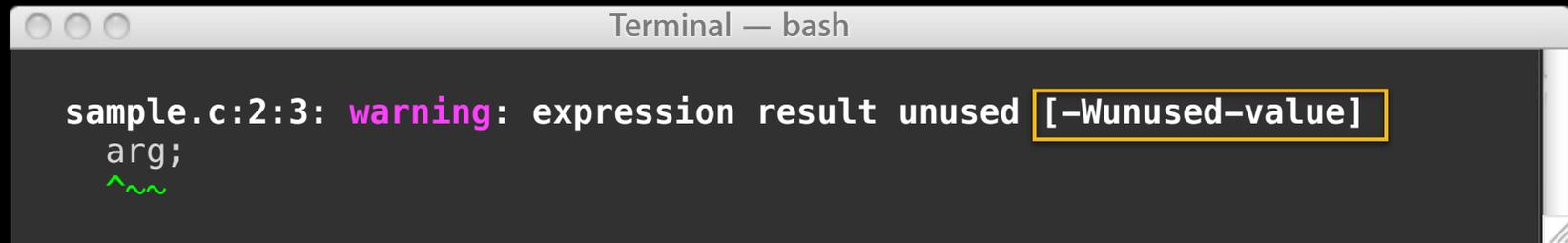


# Transition Tips

# Additional Warnings

- Apple LLVM Compiler provides more warnings

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

A terminal window titled "Terminal — bash" showing the compilation of a C file named "sample.c". The warning message is "sample.c:2:3: warning: expression result unused [-Wunused-value]" with a yellow box around the flag. Below the warning, the code "arg;" is shown with a green squiggly line under the semicolon.

```
Terminal — bash  
sample.c:2:3: warning: expression result unused [-Wunused-value]  
    arg;  
    ~~~
```

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

# Selectively Disable Warnings

- Use clang pragmas for more fine-grained control

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

Warning: Expression result unused

# Selectively Disable Warnings

- Use clang pragmas for more fine-grained control

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

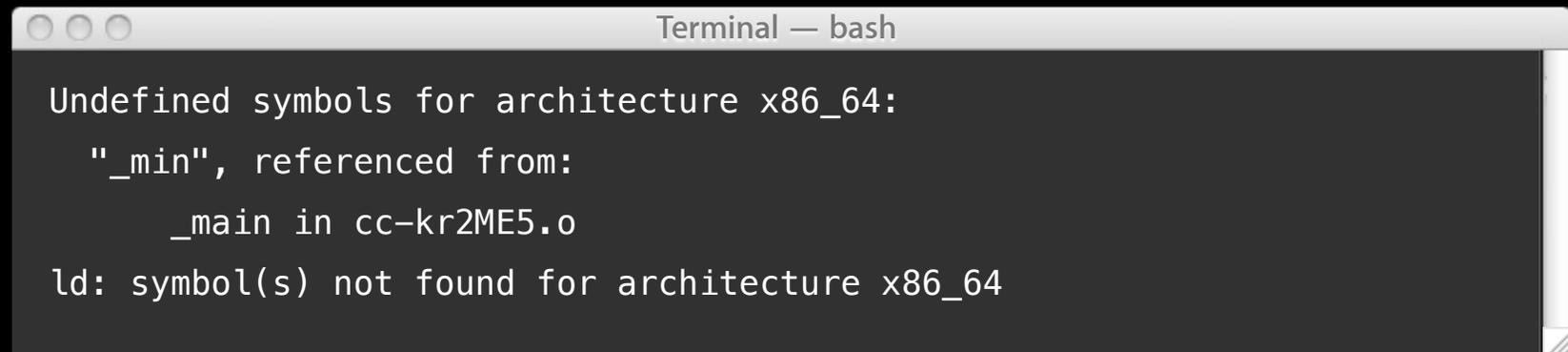
void f(int arg) {
    arg; // unused argument
}

#pragma clang diagnostic pop
```

# Fast Forward to 1999

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

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

A terminal window titled "Terminal — bash" showing a linker error. The output text is: "Undefined symbols for architecture x86\_64: \"\_min\", referenced from: \_main in cc-kr2ME5.o ld: symbol(s) not found for architecture x86\_64".

```
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
```

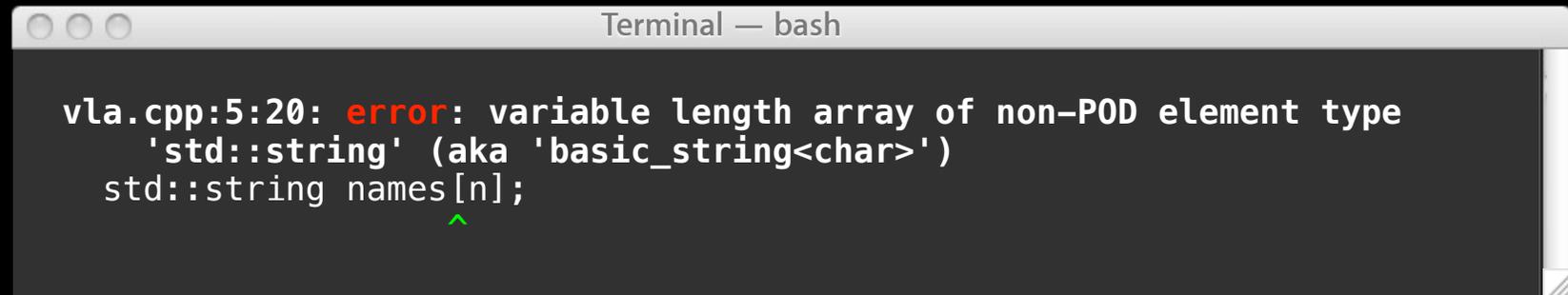
# Variable-Length Arrays in C++

- Allowed for Plain Old Data (POD) types

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

- Disallowed for non-POD types

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

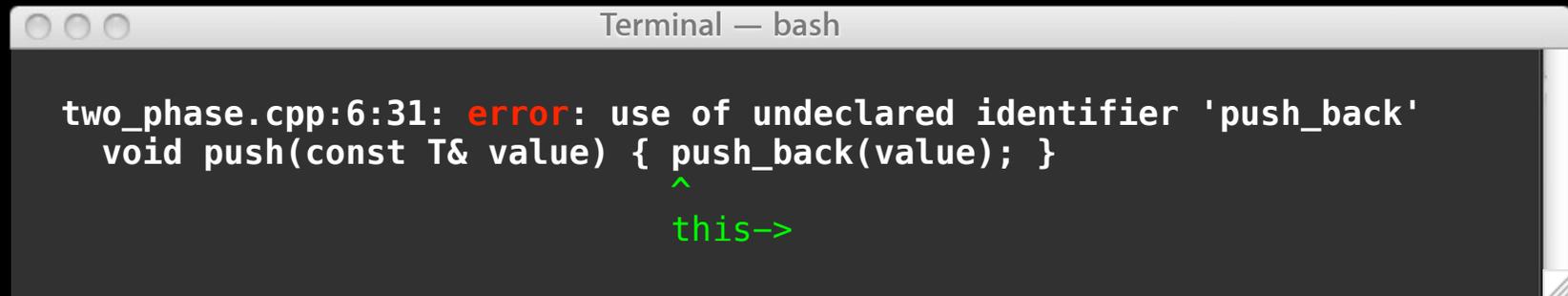


```
Terminal — bash  
  
vla.cpp:5:20: error: variable length array of non-POD element type  
    'std::string' (aka 'basic_string<char>')  
    std::string names[n];  
                    ^
```

# Name Lookup Semantics in Templates

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

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

A terminal window titled "Terminal — bash" showing a compilation error. The error message is "two\_phase.cpp:6:31: error: use of undeclared identifier 'push\_back'", with a green caret pointing to the "push\_back" in the function definition. The code snippet shown is "void push(const T& value) { push\_back(value); }".

```
Terminal — bash

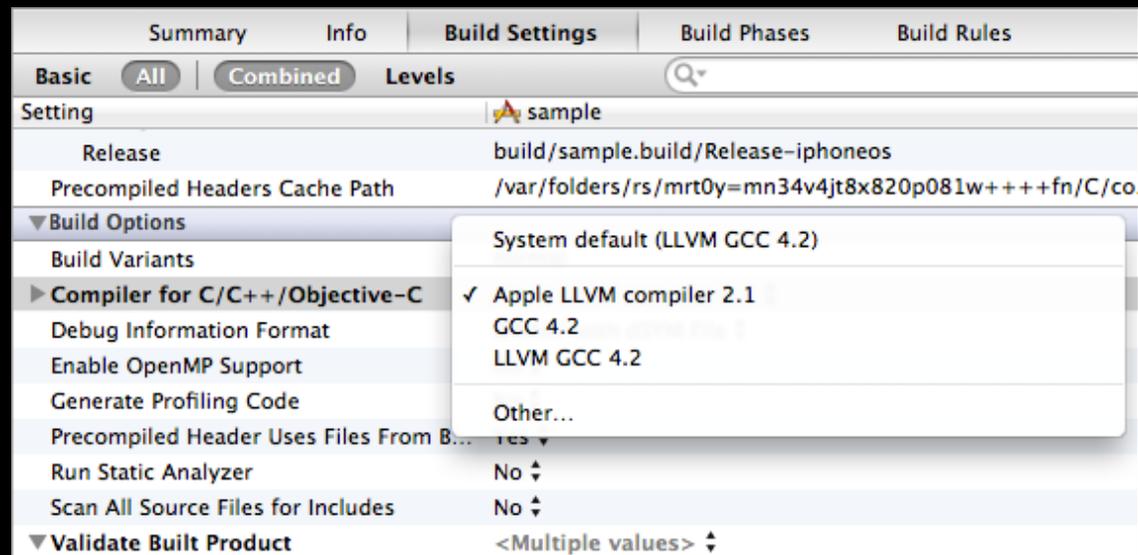
two_phase.cpp:6:31: error: use of undeclared identifier 'push_back'
    void push(const T& value) { push_back(value); }
                                ^
                                this->
```

# Moving to Automatic Reference Counting

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

# 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

# Compiler Release Summary

Two great Xcode releases

## Xcode 4.1

GCC 4.2

LLVM-GCC

LLVM Compiler 2.1

## Xcode 4.2

GCC 4.2

LLVM-GCC

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](mailto: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

