

Thoughts on Debugging

Mark Dalrymple
CocoaConf, August 2012

@borkware

<http://borkware.com/cocoaconf>



What *is* a Bug?



Ubuntu

Overview Code **Bugs** Blueprints Translations Answers

Microsoft has a majority market share

Ubuntu » Bugs » **Bug #1 (liberation)**

Reported by  [Mark Shuttleworth](#) on 2004-08-20

<http://WhatWillWeUse.com>

Problem Description (Read Only)

03-Jul-2010 11:28 PM Mark Dalrymple:
Summary:

If I use MPMediaItem to access the MPMediaItemPropertyUserGrouping property, MobileMusicPlayer crashes

What **is** a Bug?

- Bugs?
- Defects?
- Errors!

When I started debugging this problem 3 hours ago, I knew it was caused by me being a dumbass somewhere in my code.

— Gus Mueller

Don't Panic!

The Universal Troubleshooting Process

<http://troubleshooters.com/>

The Universal Troubleshooting Process

- Get the Attitude
- Make a damage control plan
- Get a complete and accurate symptom description
- Reproduce the symptom
- Do appropriate general maintenance
- Narrow it down to the root cause
- Repair or replace the defective component
- Test
- Take pride in your solution
- Prevent future occurrences of this problem

<http://troubleshooters.com/>

I. Get The Attitude

- Prepare your mental and physical work area
- You **can** solve it. It might take time.
- There's always an explanation

2. Make a Damage Control Plan

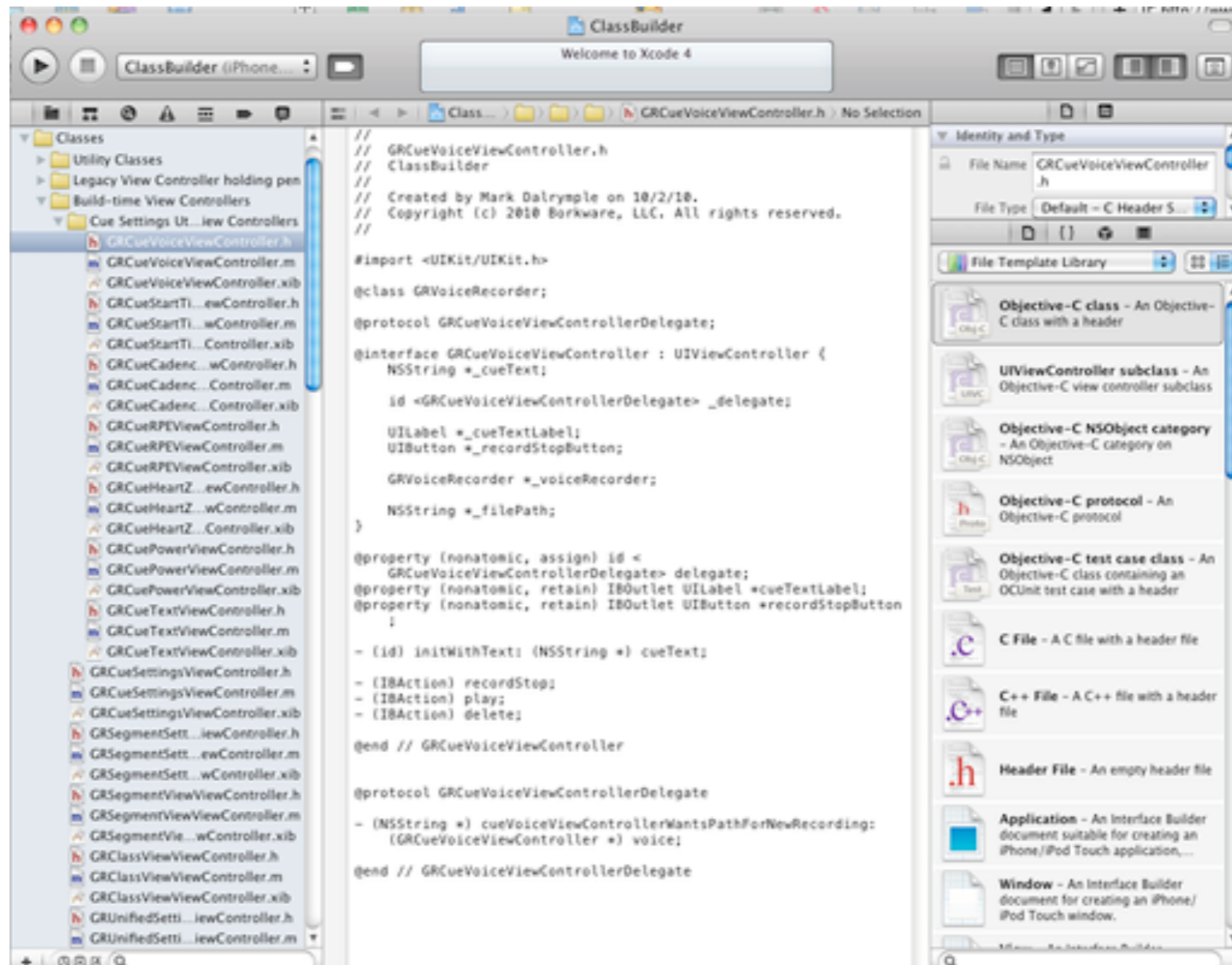
- Back up at-risk data
 - Especially configuration data
- Make sure your backups are good
- Know what a restore will entail

3. Get a Complete and Accurate Symptom Description

- HA!
- Try your best to get details
- Screen recordings are awesome.
I like ScreenFlow.



Sample Bug Report Movie



4. Reproduce the Symptom

- If it's reproducible, it's dead
- If it's intermittent, don't give up
- Be consistent with your test data

5. Do appropriate general maintenance

- Vacuum the database
- Remove Prefs / Application Support.
I like RooSwitch.
- Check the hardware
- Wave the Dead Chicken



5. Do appropriate general maintenance (2)

- Turn on Warnings (and fix them!)

<http://bit.ly/bored-zo-warnings>

- -Weverything
- -Werror
- Run the Static Analyzer

Big Nerd Ranch weblog: A Bit on Warnings

6. Narrow it down to the root cause

- Divide and Conquer / Binary Search
- Source code control is your friend
- Crashers are (usually) great
- You get better with practice



7. Repair or Replace the Defective Component

- Make your fix

8. Test

- Did the (right) symptom go away?
- Did you cause any new problems?

9. Take Pride in your Solution

- You done did good!
- Reflect. What went well? What didn't?

10. Prevent Future Occurrences of the Problem

- Repeat bugs are boring
- and embarrassing
- and embarrassing
- Beware of overcompensating

The Universal Troubleshooting Process

- Get the Attitude
- Make a damage control plan
- Get a complete and accurate symptom description
- Reproduce the symptom
- Do appropriate general maintenance
- Narrow it down to the root cause
- Repair or replace the defective component
- Test
- Take pride in your solution
- Prevent future occurrences of this problem

Own the Problem

Own it!

- Be responsible for the fix
- Finger pointing is unprofessional
- Reserve the right to say “neener neener neener” afterwards

Own it!

- Be responsible for the fix
- Finger pointing is unprofessional
- Reserve the right to say “neener neener neener” afterwards

Dude... enough with the soapbox...

Keep After it

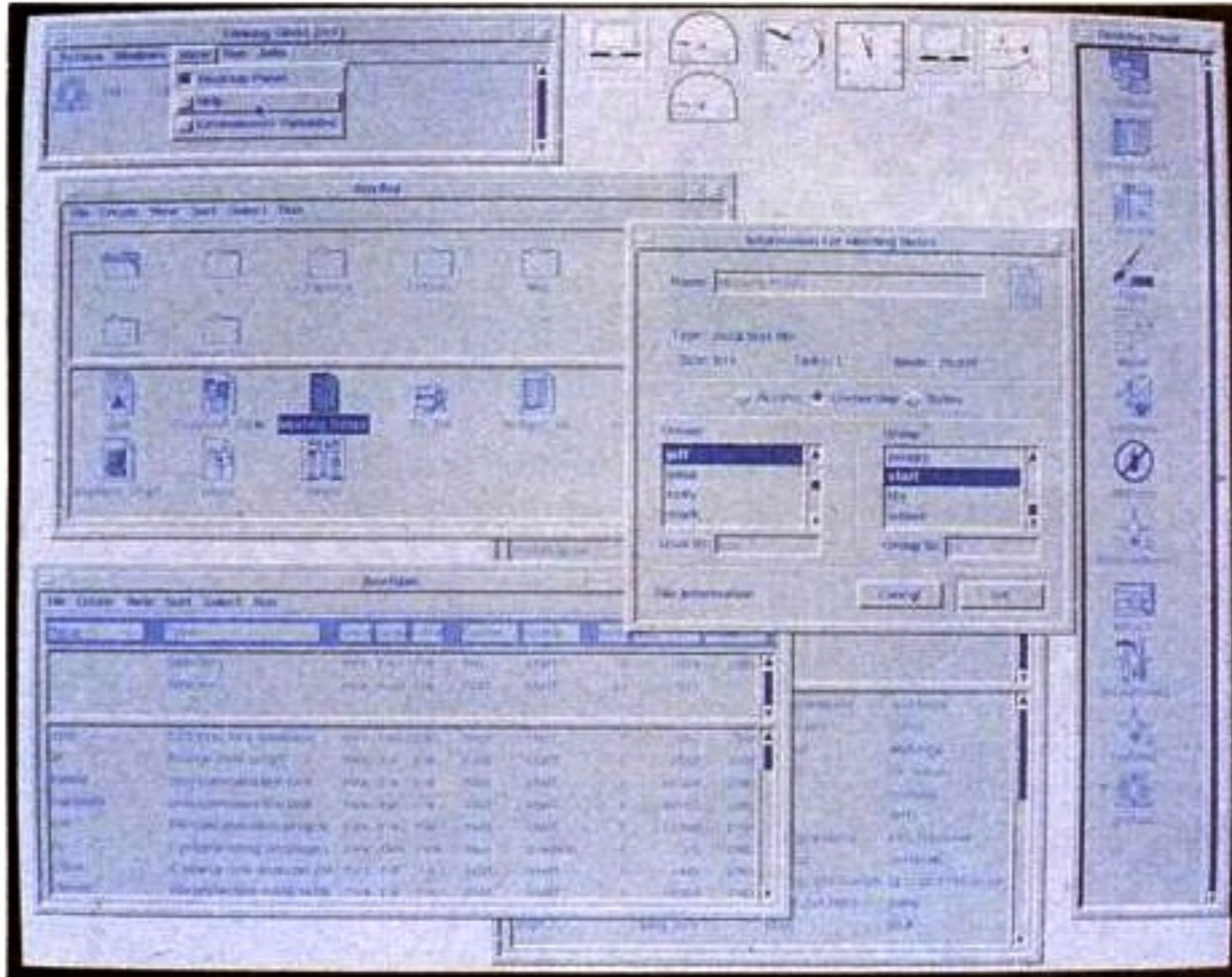
Caveman Debugging

Caveman Debugging

- Embrace the log
- Most useful if you have fast turnaround
- Handy for learning new API
- Don't over-do it
- DTrace can be handy, too
- Modern Xcode breakpoints - pretty good
- Consider a ring buffer if you keep it around

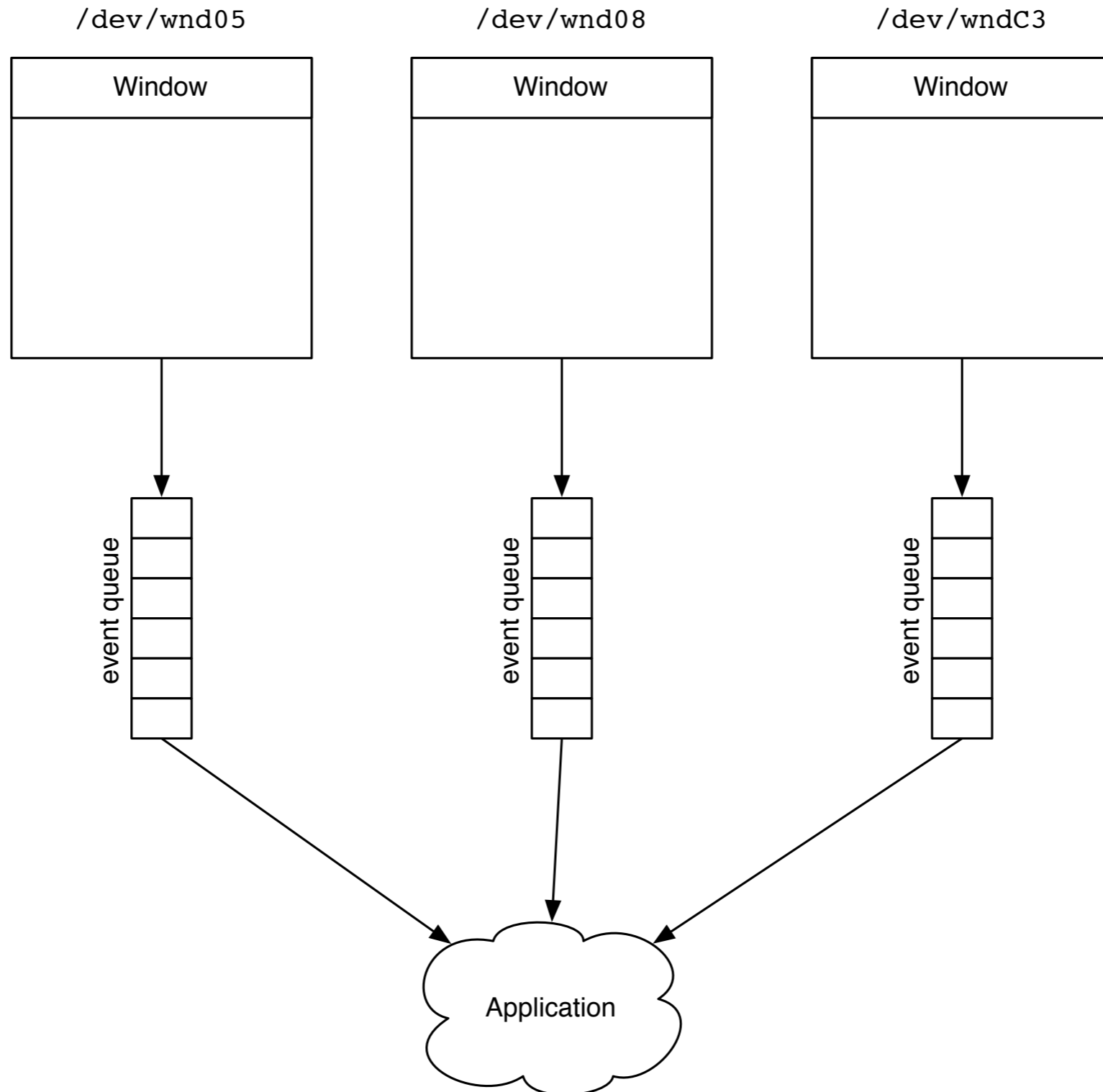
GTMLog

My first (professional) bug



Looking Glass brings the look and feel of Motif to the Sun View environment, along with point-and-click ease of use.

My first (professional) bugfix



Some Caveman Examples

```
- (id) initWithFrame: (CGRect) frame {
    if ((self = [super initWithFrame: frame])) {
        NSLog (@"SNORK: ACTUALLY INITING");
        [self doPoints];
        _singlePath = YES;
    }
    return self;
} // initWithFrame
```

```
NSURL *ubiquitousFolder = [fm URLForUbiquityContainerIdentifier: nil];

if ([ubiquitousFolder path]) {
    NSLog(@"FLOK %@", [fm subpathsOfDirectoryAtPath: [ubiquitousFolder path]
        error: nil]);
}
NSLog (@"LOAD THEM");
_iCloudURLs = [[NSMutableArray alloc] init];
```

The Debugger

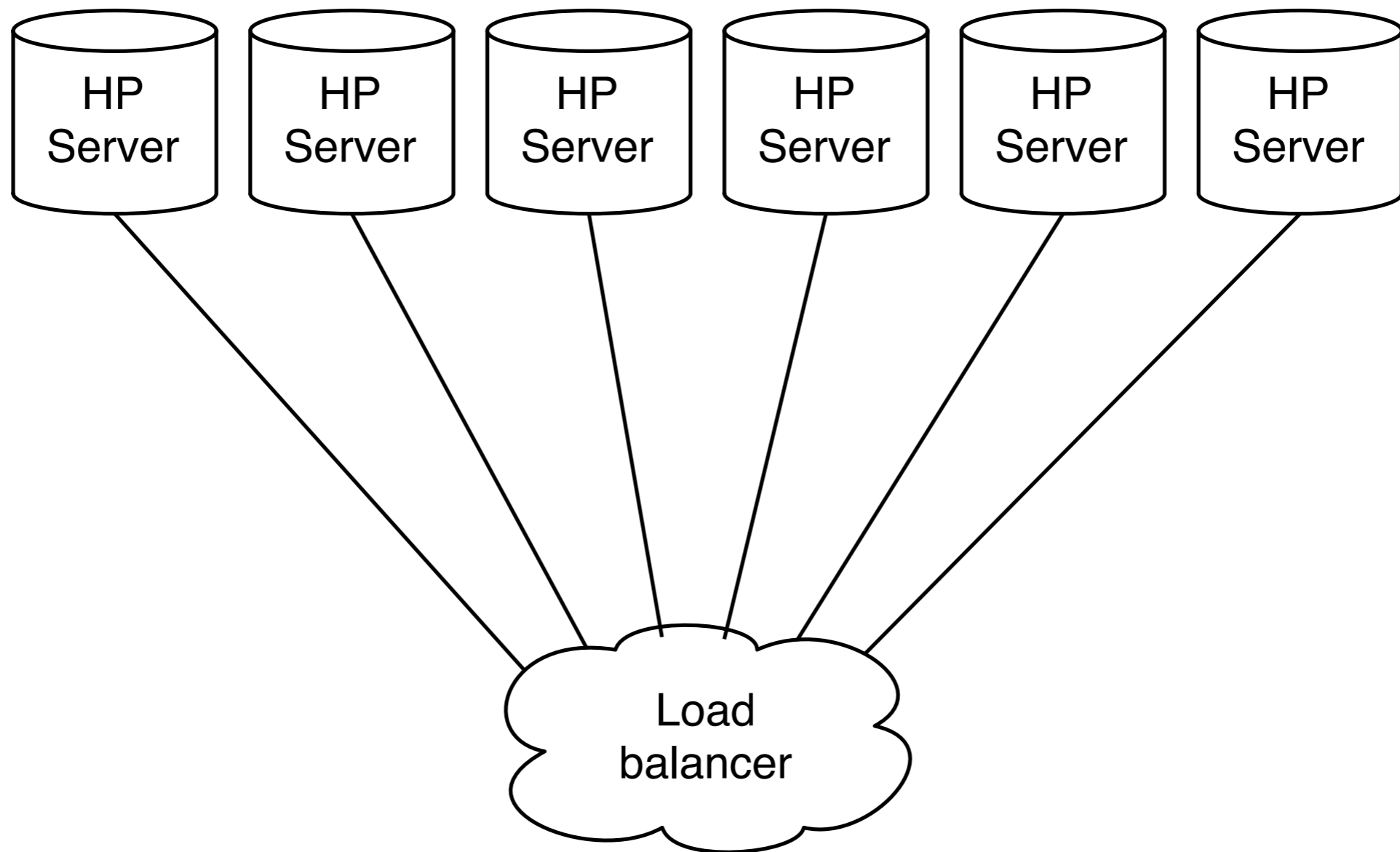
The Debugger

- Uncle Bob: “Debuggers are a Wasteful Timesink”
- But we in [Objective] C [++] Land do need them
- Don't get **too** debugger happy

Other uses for the Debugger

- Code Exploration
 - How's this work?
 - Where does this new feature go?
- Single-stepping to test new code
- Camping on Crashers

Web Server Crash



Novel uses

- Fake Breakpoints for intermittent problems:

```
static BOOL spin = YES;  
while (spin) ;
```

- Stochastic Profiling

Beware Convenience

```
TString *timestamp =  
    month + "/" + day + "/" + year + " "  
    + hours + ":" + minutes + ":"  
    + seconds;
```

Optimize Your Tools, and Yourself

The Command Line

Embrace the Command Line

- Finding that server bug would have been harder without the command-line
- Remote development and debugging
- One-off Test Cases

One-off test cases

```
#import <Foundation/Foundation.h>

// gcc -g -Wall -framework Foundation -o displayname displayname.m

int main (void) {
    [[NSAutoreleasePool alloc] init];

    NSString *blah = [[NSFileManager defaultManager]
                      displayNameAtPath: @"/Users/markd/Library"];
    NSLog(@"blah %@", blah);

    return 0;
} // main

% !g
gcc -g -Wall -framework Foundation -o displayname displayname.m

% !.
2010-10-21 16:26:17.972 displayname[2369:903] blah Library



---


% !.
2010-10-21 16:26:40.105 displayname[2375:903] blah Bibliothèque
```

Twiddle Sysprefs

Mac Apps and the Command Line

- `% xcodebuild -configuration Debug -target Groovin`
- You can run and debug Mac apps from the command line
- `% ./build/Debug/Groovin.app/Contents/MacOS/Groovin`
- `(gdb) attach -waitfor Groovin`
- `(lldb) process attach --waitfor -n Groovin`

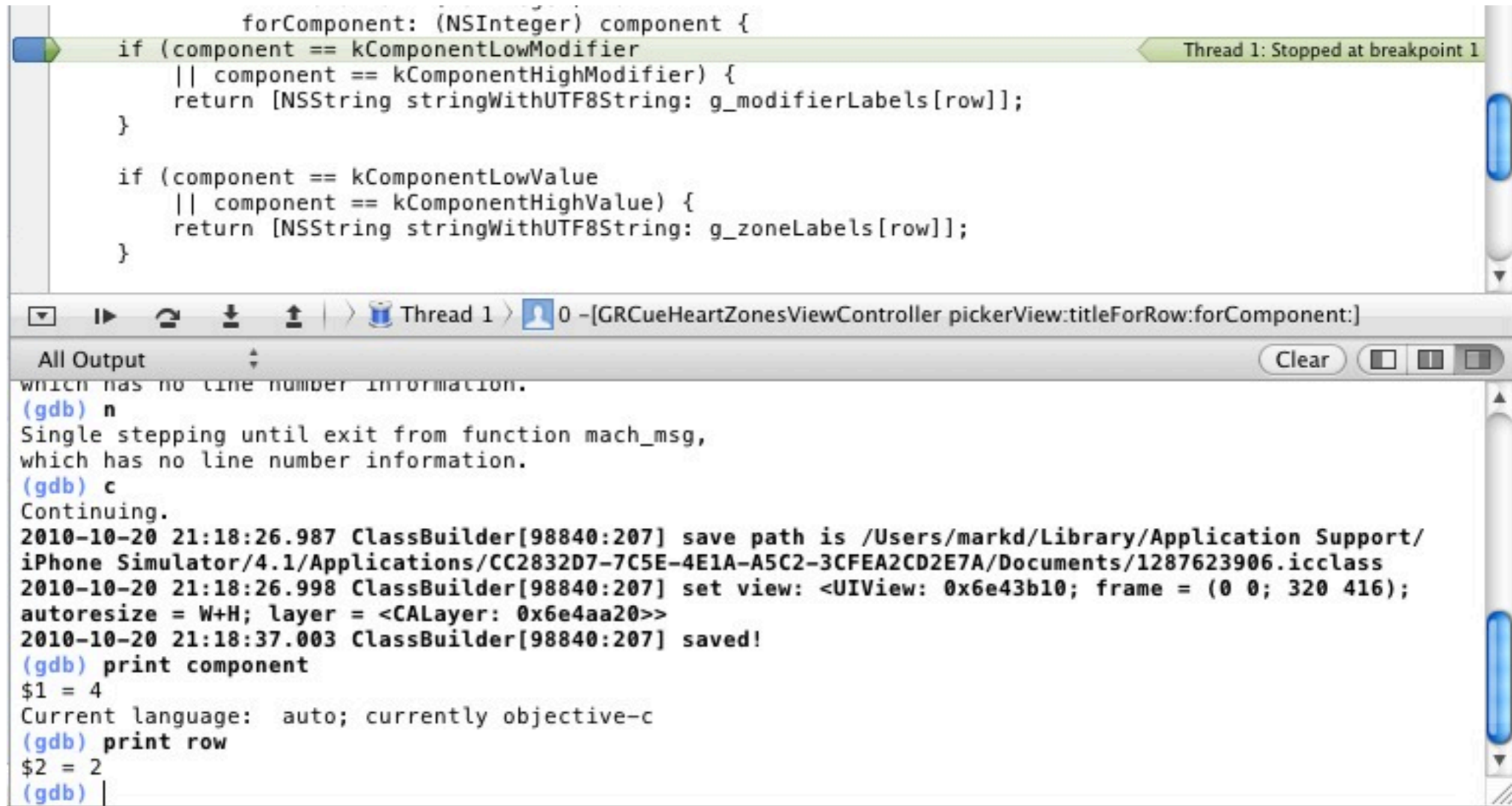
App Arguments

- `.... /Groovin -blah 10 -duckies "hello"`
- **NSUserDefaults for accessing arguments**
- `int blah = [defaults integerForKey: @"blah"];`
- `NSString *ducks =
[defaults stringForKey: @"duckies"];`
- **Knobs for QA and Debugging.**
- **Faster Turnaround**

Live Command-Line!

- (woo!)
- It's just emacs. Don't be afraid.
 - Jason Felice's talk next in Ballroom A

debugger Command Line in Xcode



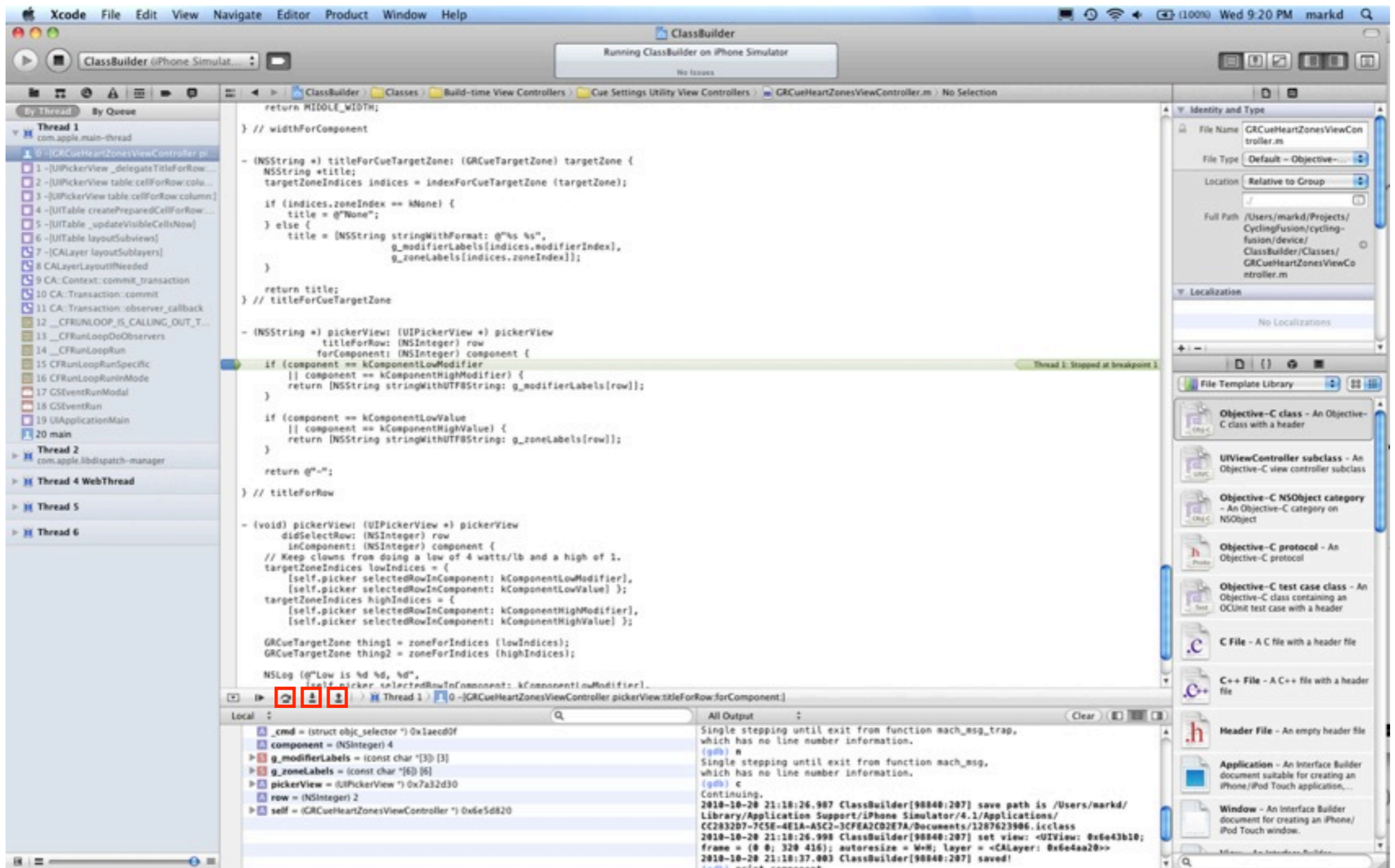
The screenshot shows the Xcode IDE with a debugger window open. The code being debugged is Objective-C, and the debugger is GDB. The code is as follows:

```
forComponent: (NSInteger) component {  
    if (component == kComponentLowModifier  
        || component == kComponentHighModifier) {  
        return [NSString stringWithUTF8String: g_modifierLabels[row]];  
    }  
  
    if (component == kComponentLowValue  
        || component == kComponentHighValue) {  
        return [NSString stringWithUTF8String: g_zoneLabels[row]];  
    }  
}
```

The debugger window shows the following GDB session:

```
(gdb) n  
Single stepping until exit from function mach_msg,  
which has no line number information.  
(gdb) c  
Continuing.  
2010-10-20 21:18:26.987 ClassBuilder[98840:207] save path is /Users/markd/Library/Application Support/  
iPhone Simulator/4.1/Applications/CC2832D7-7C5E-4E1A-A5C2-3CFEA2CD2E7A/Documents/1287623906.icclass  
2010-10-20 21:18:26.998 ClassBuilder[98840:207] set view: <UIView: 0x6e43b10; frame = (0 0; 320 416);  
autoresize = W+H; layer = <CALayer: 0x6e4aa20>>  
2010-10-20 21:18:37.003 ClassBuilder[98840:207] saved!  
(gdb) print component  
$1 = 4  
Current language: auto; currently objective-c  
(gdb) print row  
$2 = 2  
(gdb) |
```

Why would you want to?



HP-16C

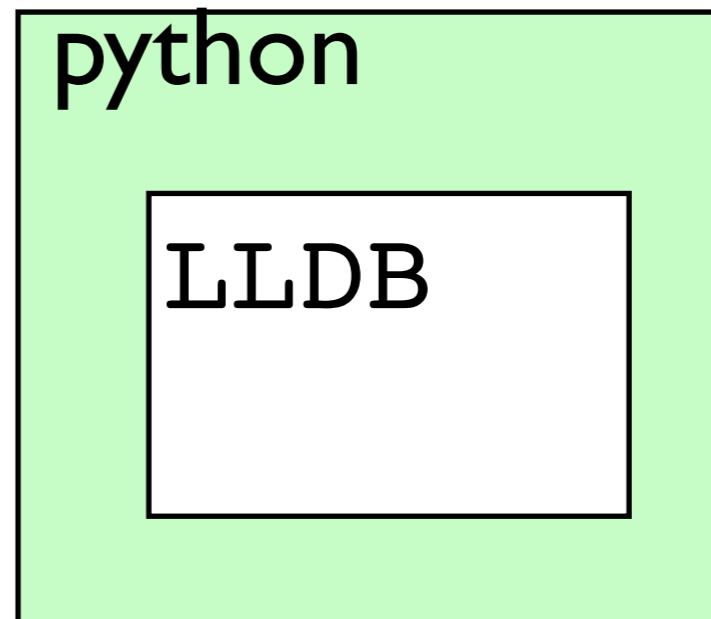
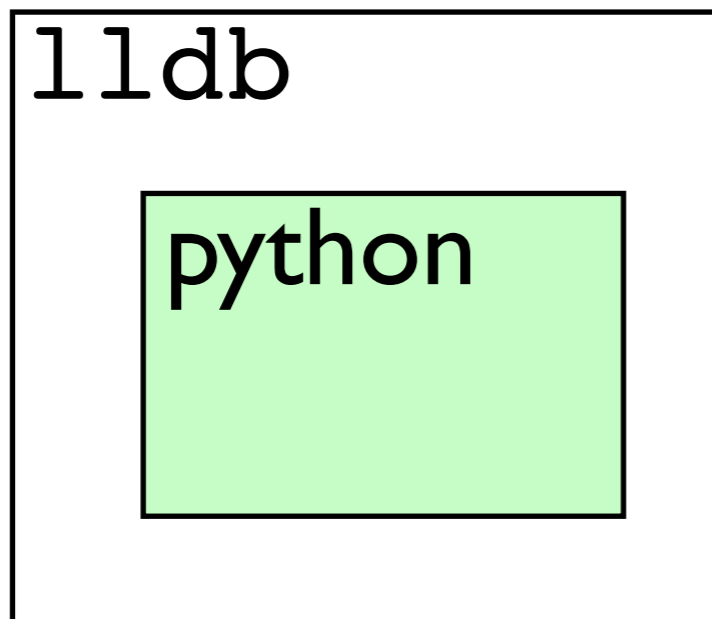
- (gdb) po
- (gdb) print i \$1 = 17263812
- (gdb) print/x i \$2 = 0x1076cc4
- (gdb) print/t i \$3 = 1000001110110110011000100

Displaying And Changing Data

- (gdb) print *node->next
\$5 = {
 theChar = 95 'b',
 next = 0x30AB0
}
- (gdb) set node->next->theChar = 'q'
- (gdb) print \$5->next

lldb / LLDB scripting

- You've got Python in my debugger!
- You've got debugger in my Python!



Breakpoints n'at

Other Tools

Other Tools

- Crash Logs / Core Files
- NSZombieEnabled
- DTrace
- Using Profilers for breakpoint locations
- Don't forget: it might be the hardware

Inspectors

The screenshot displays several tool windows from the Nutron application:

- Dictionary Inspector:** A table with columns 'key' and 'set'. It lists various system keys and their corresponding values.
- Managed Object Context Inspector:** A table with columns 'first name', 'last name', 'address', 'email', 'phone', and 'salary'. It contains data for several employees like Phil Berry, John Smith, and Britany Holder.
- Inspector (top left):** Shows the structure of an `NSConcreteTextStorage` object, including properties like `NSFont`, `NSLink`, and `NSParagraphStyle`.
- Inspector (middle left):** Shows the structure of an `NSMenu` object, including `Title`, `SupersMenu`, and `Items`.
- Inspector (center):** A detailed view of a `MandelbrotView` object, showing its `context`, `parser`, `a`, `b`, `symbols`, `isa`, `progressTextField`, `refreshButton`, `_nuivars`, and `progressBox`.
- Inspector (right):** Shows the class hierarchy for `MandelbrotView`, including `NSView`, `ivars` (like `id`, `progressTextField`, `refreshButton`), `properties`, `class methods`, and `instance methods` (like `didEnd:returnCode:con...`, `saveImage:`, `refreshImage:`, etc.).
- Console (bottom):** Shows a sequence of commands and their outputs:

```
1 > (set a '(1 2 3 4 5))
(1 2 3 4 5)
2 > (set b (a map:(do (n) (* n n))))
(1 4 9 16 25)
3 > (set v (select-view))
<MandelbrotView:19f1f0>
4 >
```

<http://bit.ly/trynutron>

When I'm Having Problems

- Look for Code Smells
- Keep a Log. I like VoodooPad.



Analogs

3
10/13/97

HP performance ⁷⁷

App Server has some bad performance problems w.r.t.
something like Apache (see pg 75) [At least it's
more consistent!] Baseline is ~~118 conn/sec~~¹⁰

105

105 conn / 10 sec
(w/lossing on)

Good Lord, where to start?

Just for fun, see what impact ^{mutex} locking has. Just
returning in Ns-Lock Mutex & Ns-Unlock Mutex.



Hmm, ~~got~~ not getting any connections reported on the
log, or any connections possible. ugh.

What if we take out the critical section stuff?

It works, & got these #'s:

125 97 65 65 80 83 97 109

90.13

avg of 90.13 / 10 sec

A Typical Debugging Log

ProxyFun

so it looks like the easiest point of contact is whacking
the NSURLRequest. What can be done there?

Change headers
HTTP Body Stream
whether should handle cookies

suck

yeah

where does the NSURLHandle jazz live?

nowhere :-(
Actually, NSURLHandle has been deprecated, use NSURLConnection instead

Is thre NSURLConnection jazz?

[GTMHTTPFetcher](#) uses it

What's useful in the NSURL connection |API?

"The interface for NSURLConnection is very sparse providing only the controls tos tart and cancel asynchronous loads"

Takes an NSURLRequest
Has a delegate
- connection:willSendRequest:redirectResponse: lets you make a new request.
??? does it get called with the first request?

What about other APIs



When I'm Still Having Problems

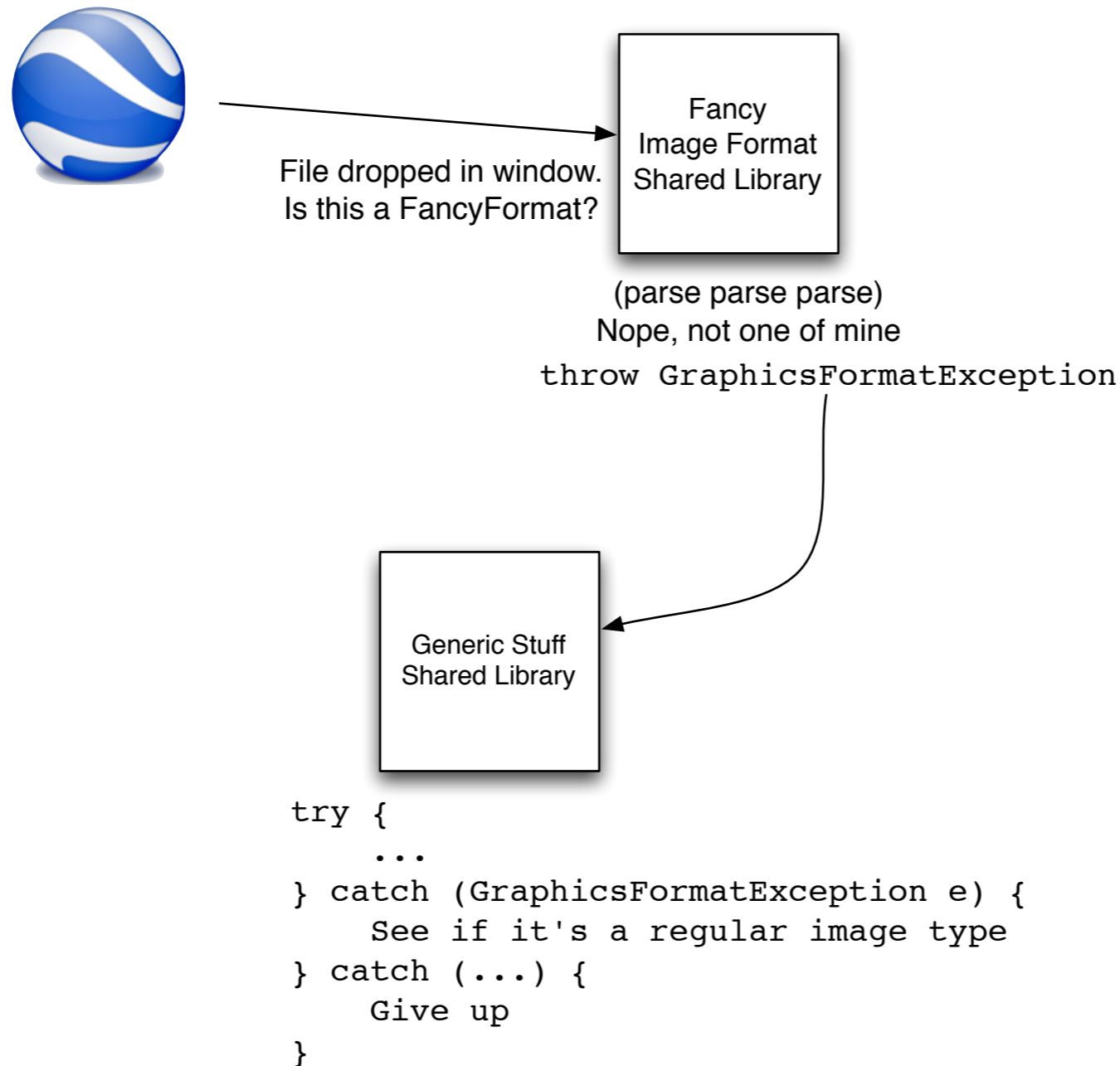
- Rubber Duck
- Sleep on it
- Find some smarter friends



The Case of the Thrown Images

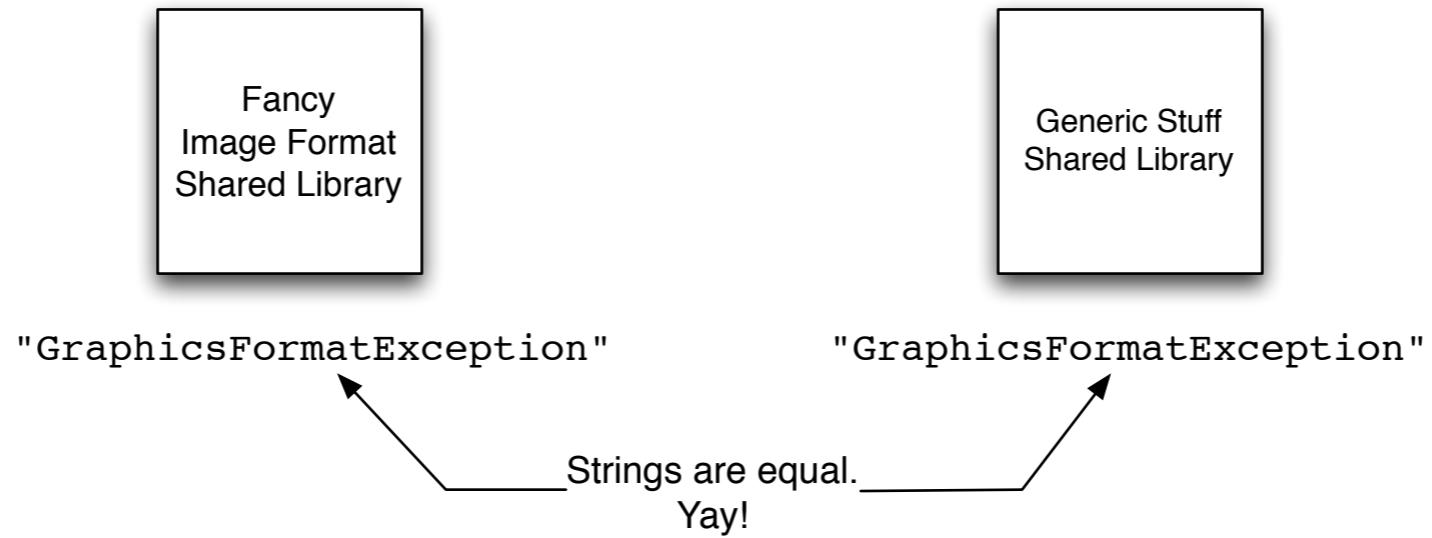


What, but not the Why

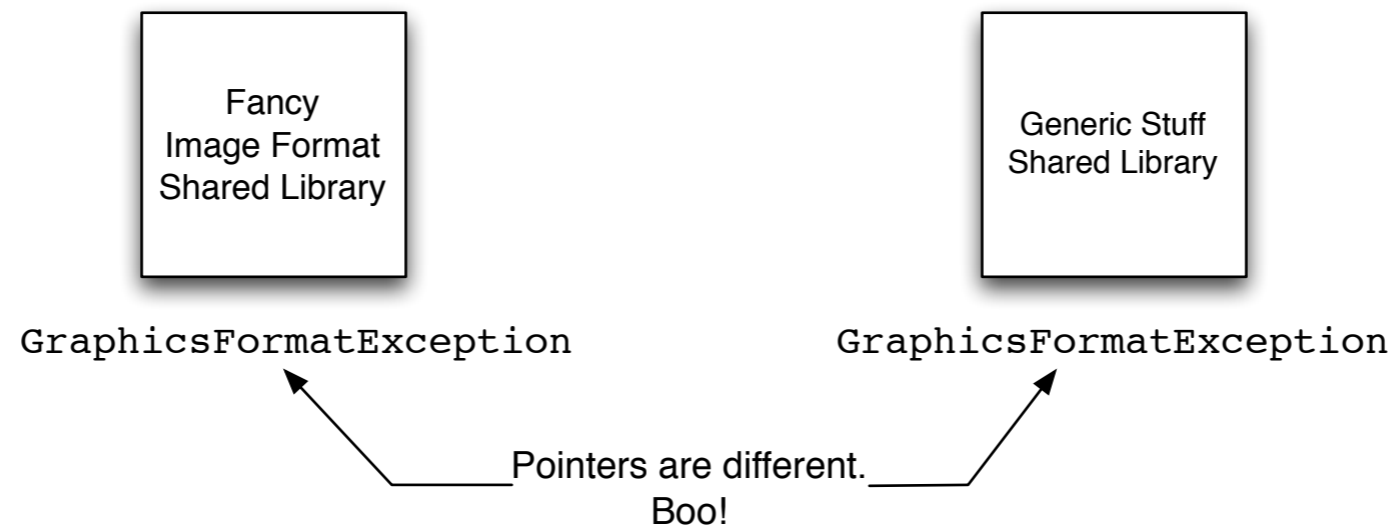


Why

GCC 3

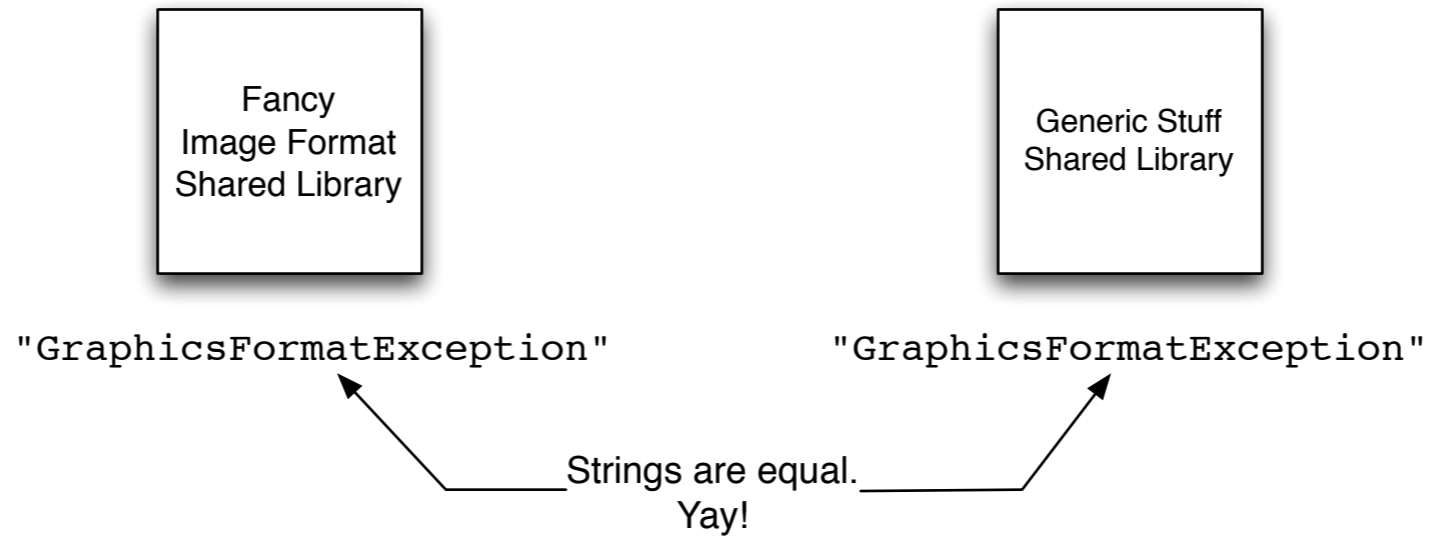


GCC 4

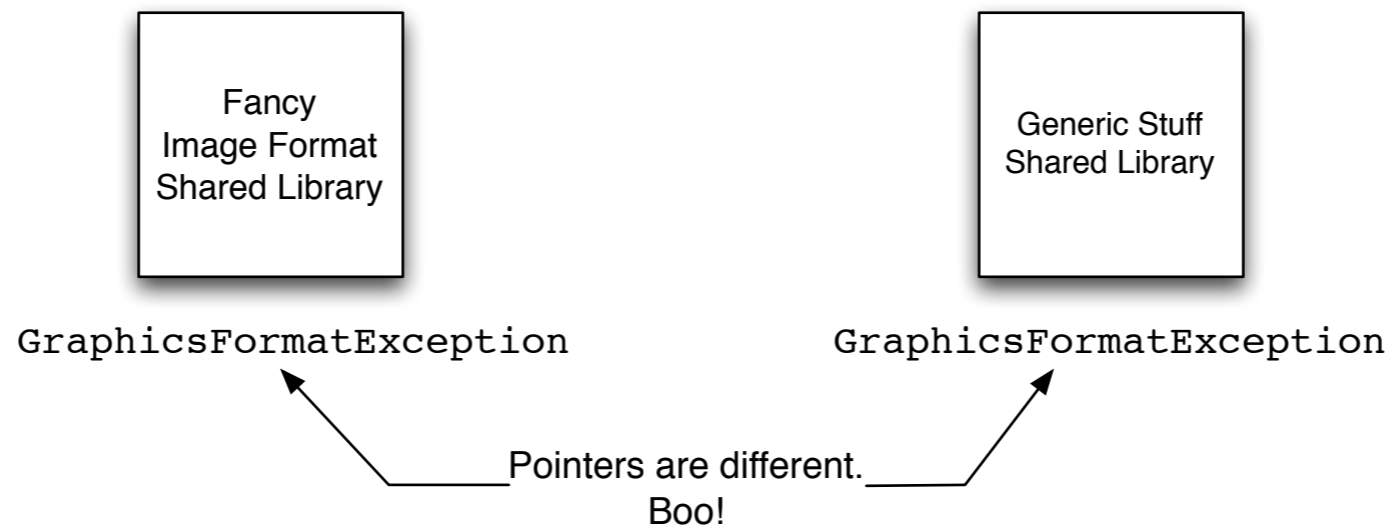


Why

GCC 3



GCC 4



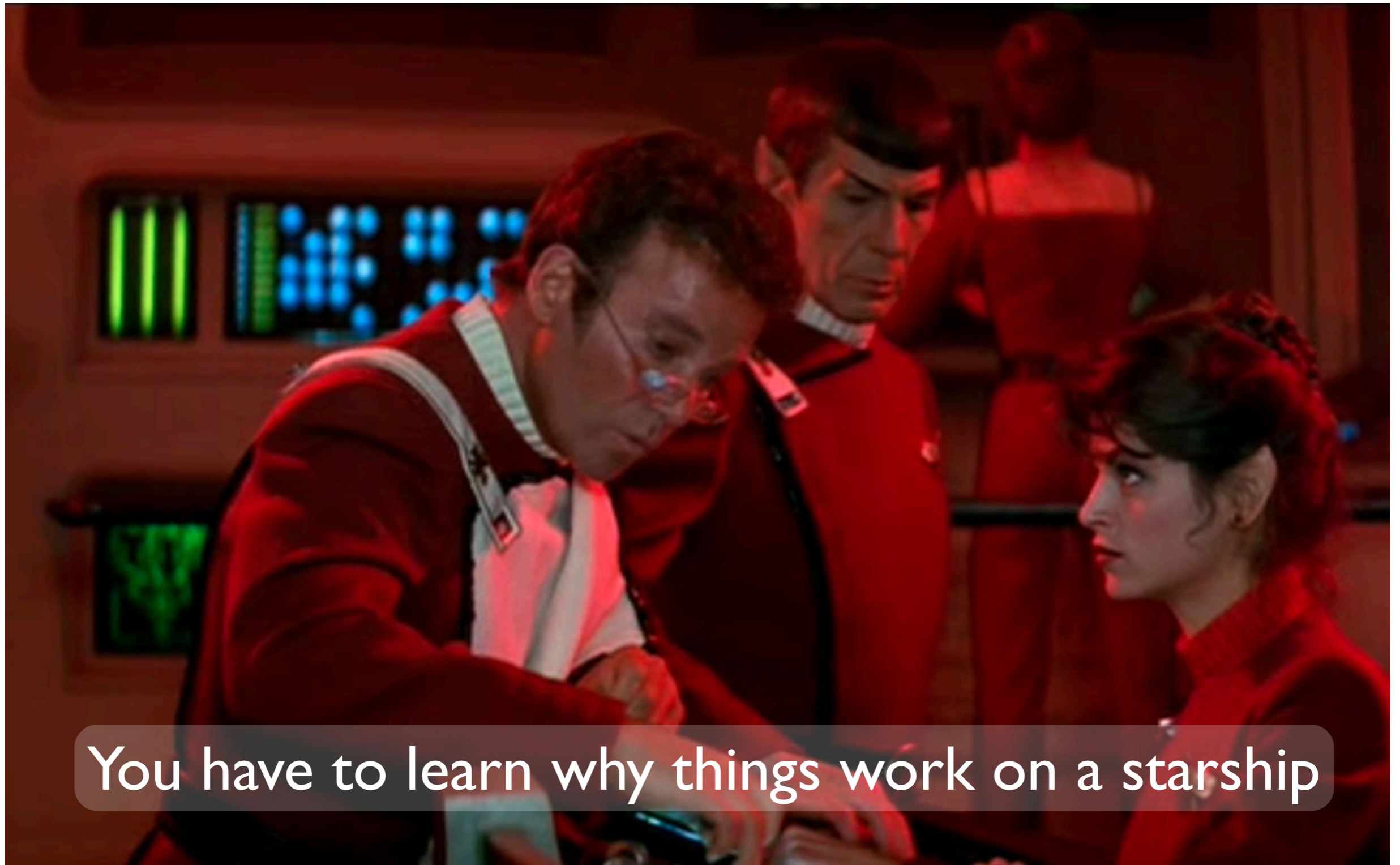
`dlopen (so_path, RTLD_LOCAL) -> dlopen (so_path, RTLD_GLOBAL)`

Optimizing your build time

I hate waiting

- Fast turnaround is vital for debugging
- Compilation and linking are bottlenecks
- Build-time dependency checks are not perfect
- Knowledge is power

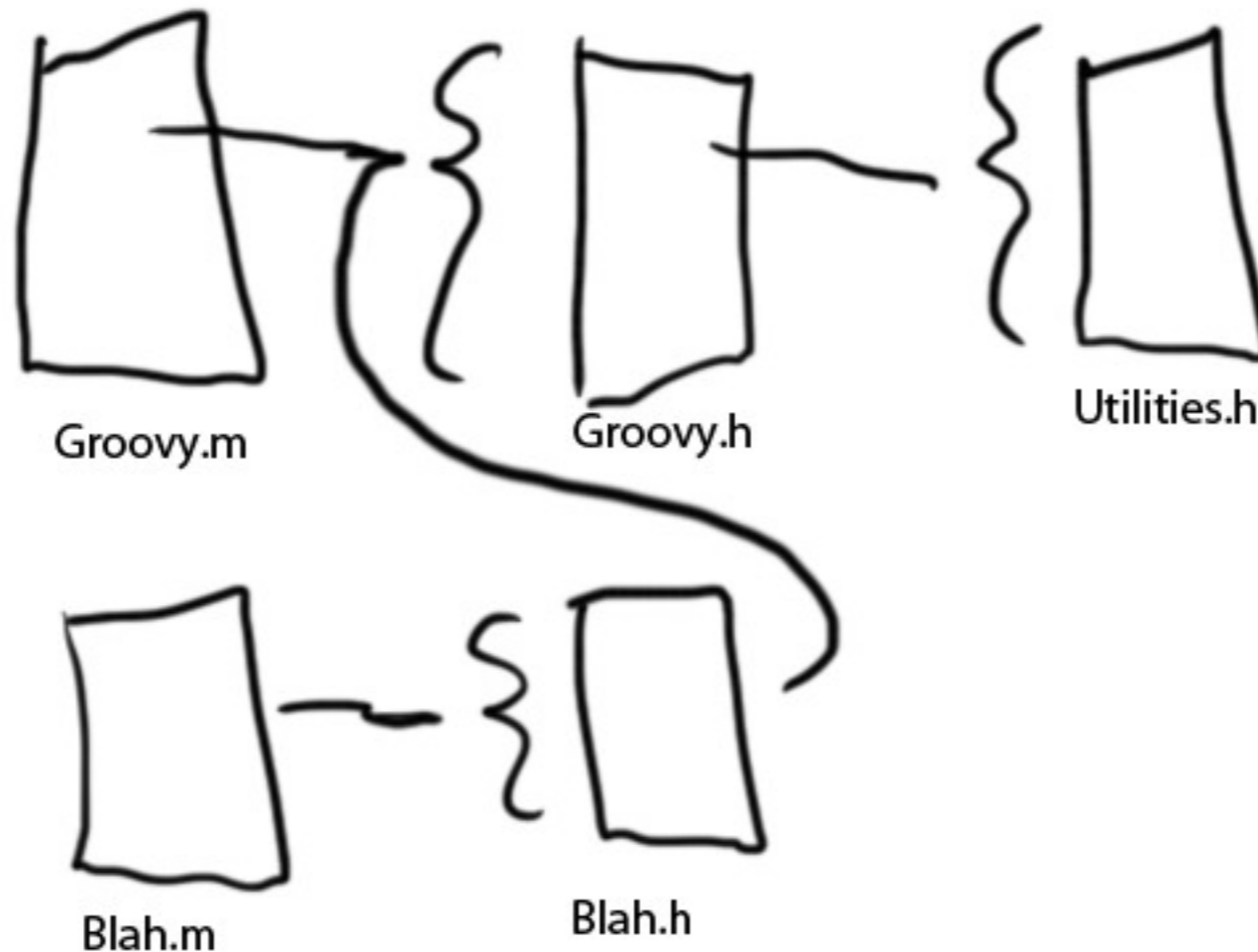
How Things Work



You have to learn why things work on a starship

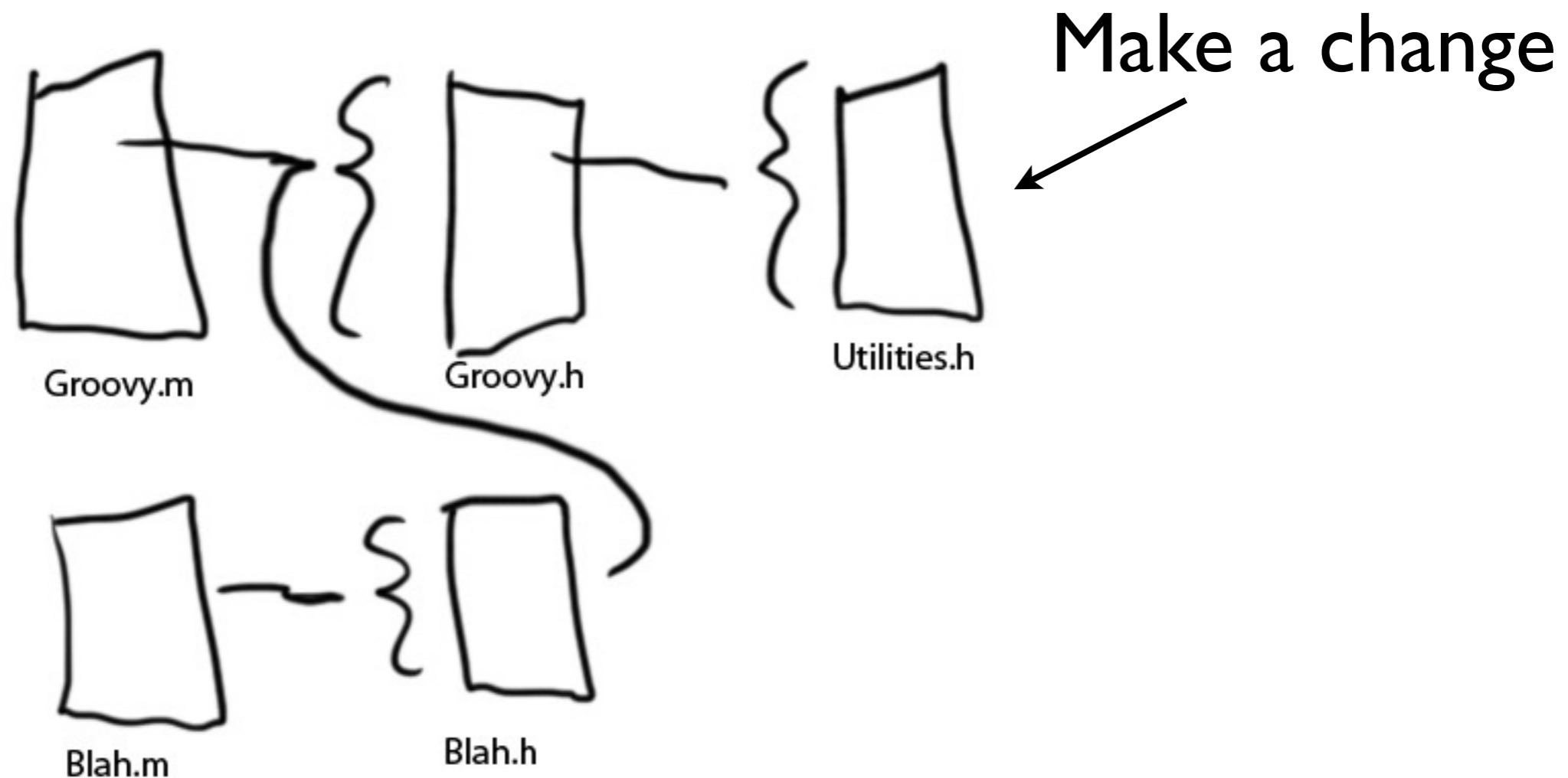
Compile Time Dependencies

- You pay a price for including headers you don't need



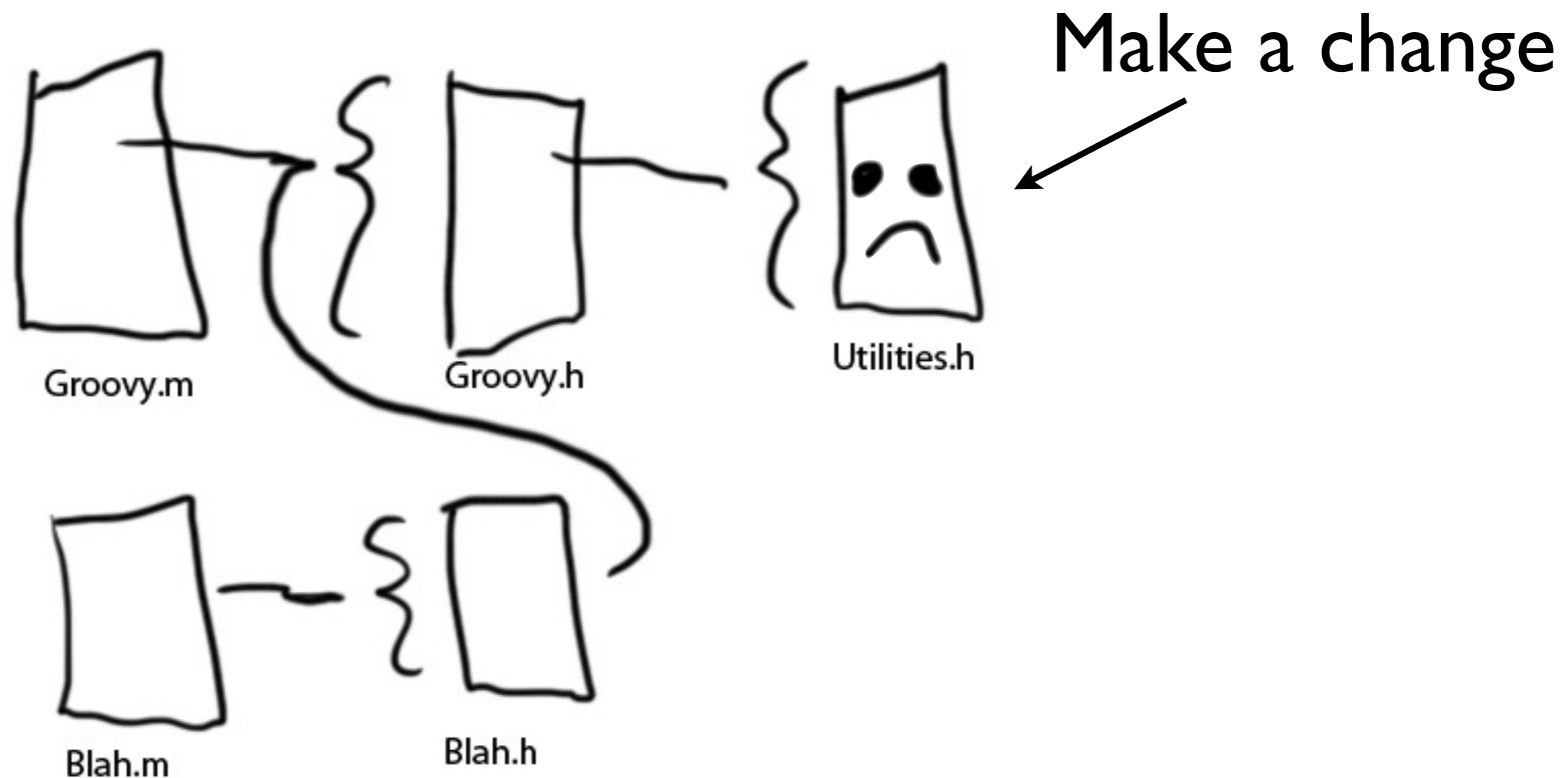
Compile Time Dependencies

- You pay a price for including headers you don't need



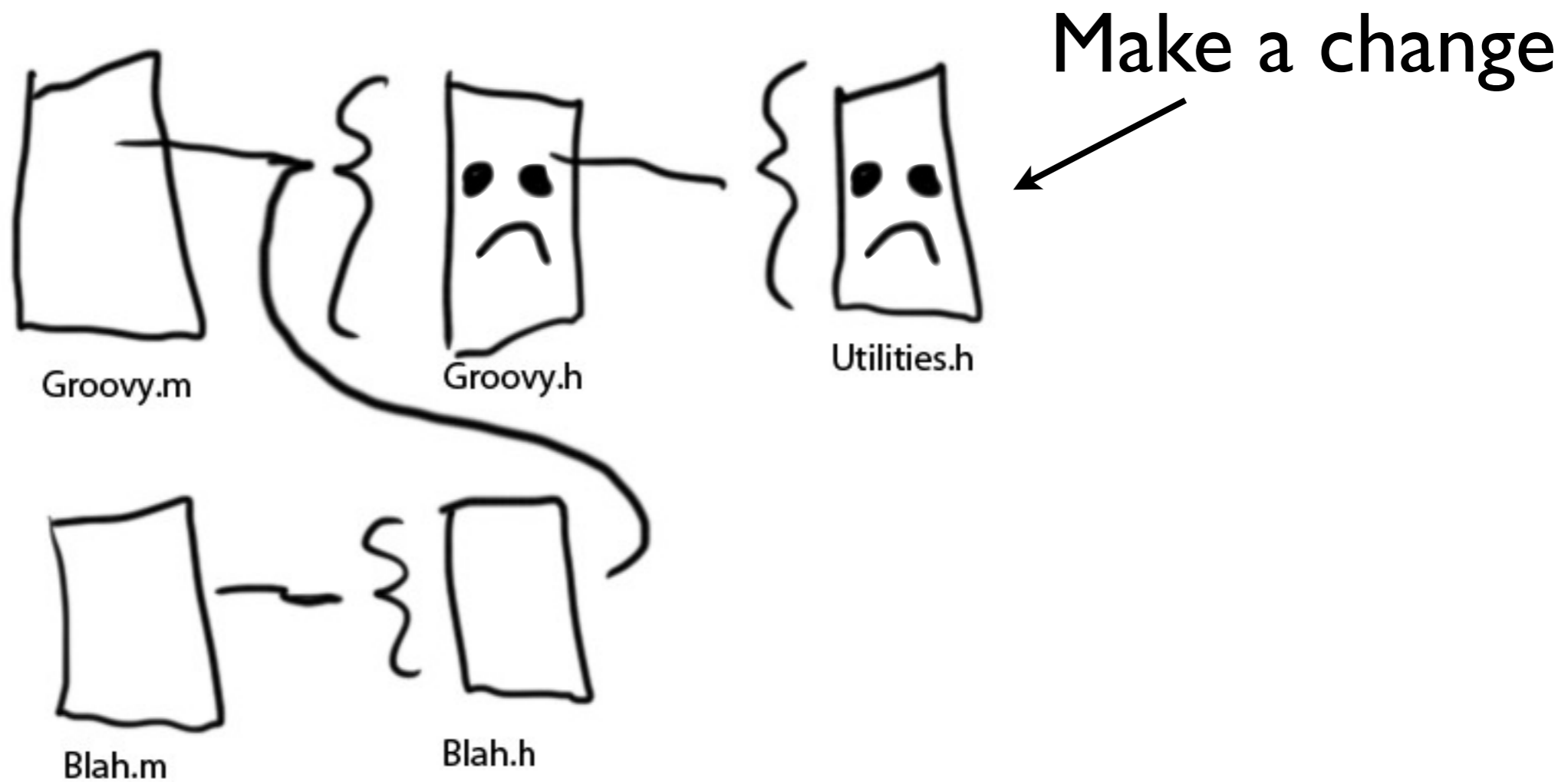
Compile Time Dependencies

- You pay a price for including headers you don't need



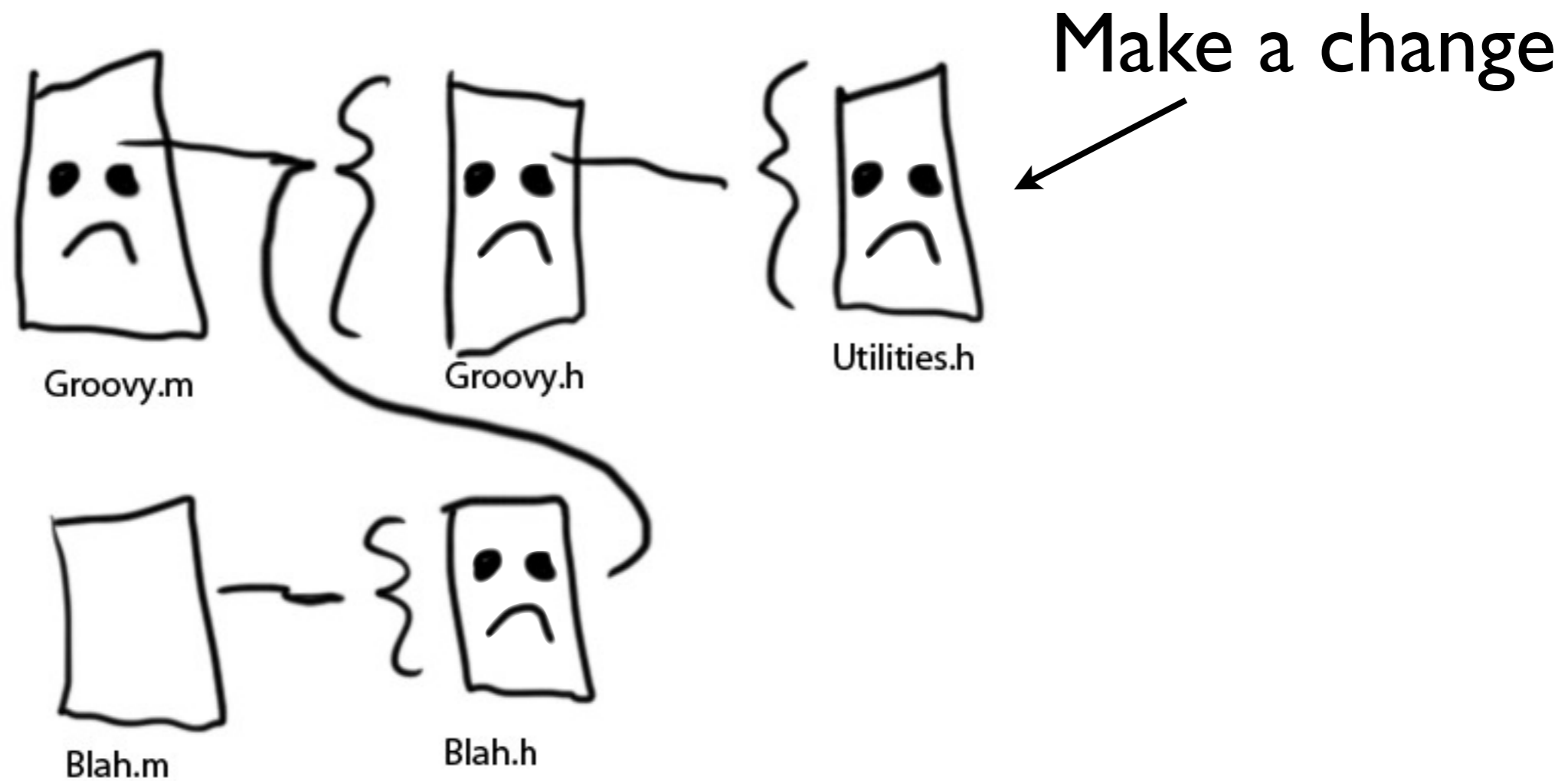
Compile Time Dependencies

- You pay a price for including headers you don't need



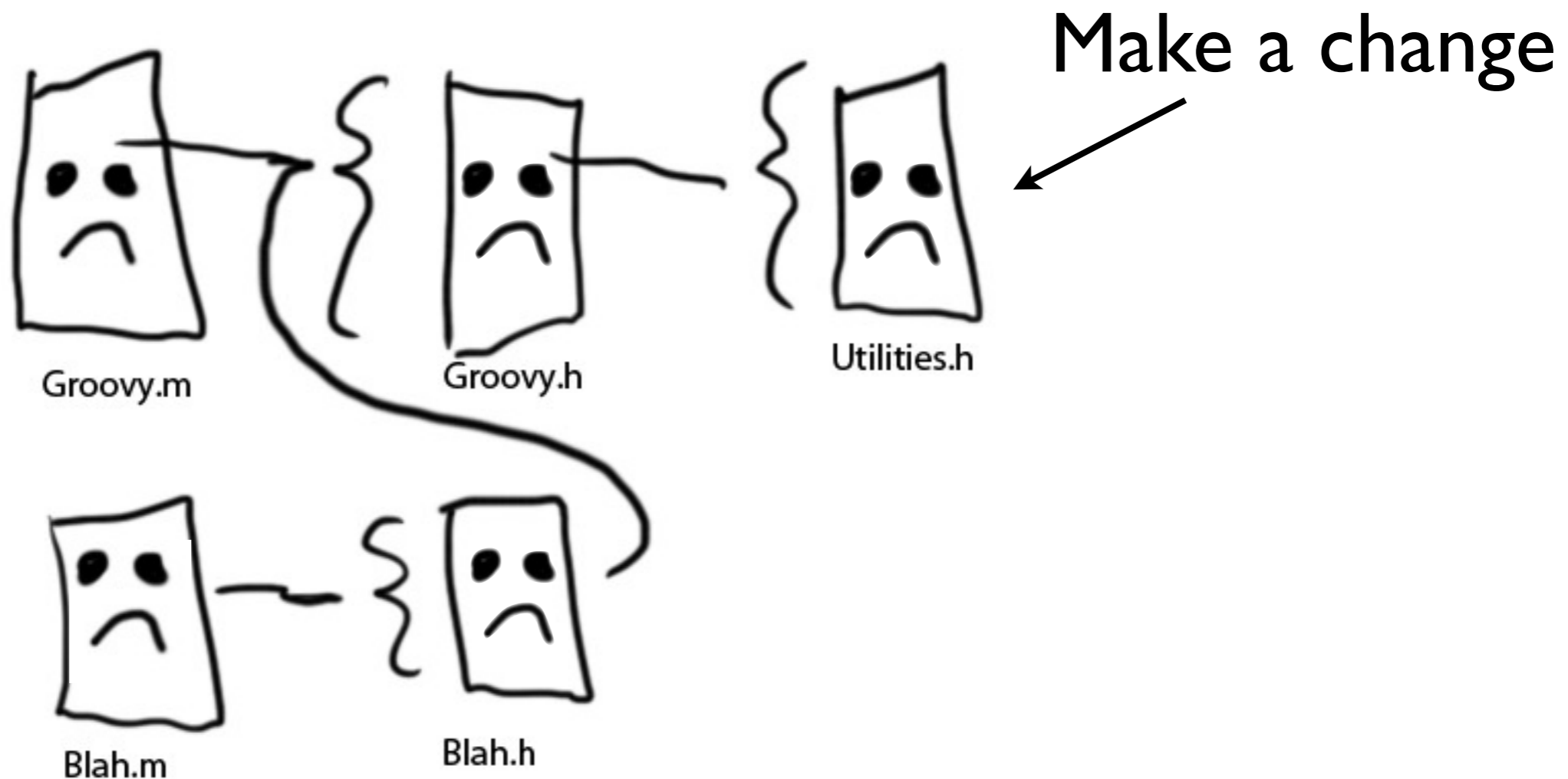
Compile Time Dependencies

- You pay a price for including headers you don't need



Compile Time Dependencies

- You pay a price for including headers you don't need



The Most Important Questions

What Up?

- What's new?
- What's changed?
- What's different?

Be Excellent to Each
Other

Be nice to your Opsen

- They're cranky, and they're fun
- They can save your butt

Get to Know your Community



cocoacconf

the conference for iPhone, iPad, and Mac developers

NSCConference

NSCoder Night

Your time to code!



Pittsburgh TechFest

2012



Bridging Pittsburgh's software communities



The
seattle
coders

2c SecondConf

iOSDevCamp



R.I.P :-)

The New York iPhone Software Developers Meetup

So, Wha'happened?

- Don't Panic
- Own the problem
- Keep after it
- Optimize your tools and yourself
- Be Excellent to each other
- Practice, learn, and reflect