Thoughts on Debugging

What *is* a Bug?

http://WhatWillWeUse.com

What *is* a Bug?

- Bugs?
- Defects?
- Errors!

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

1. 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

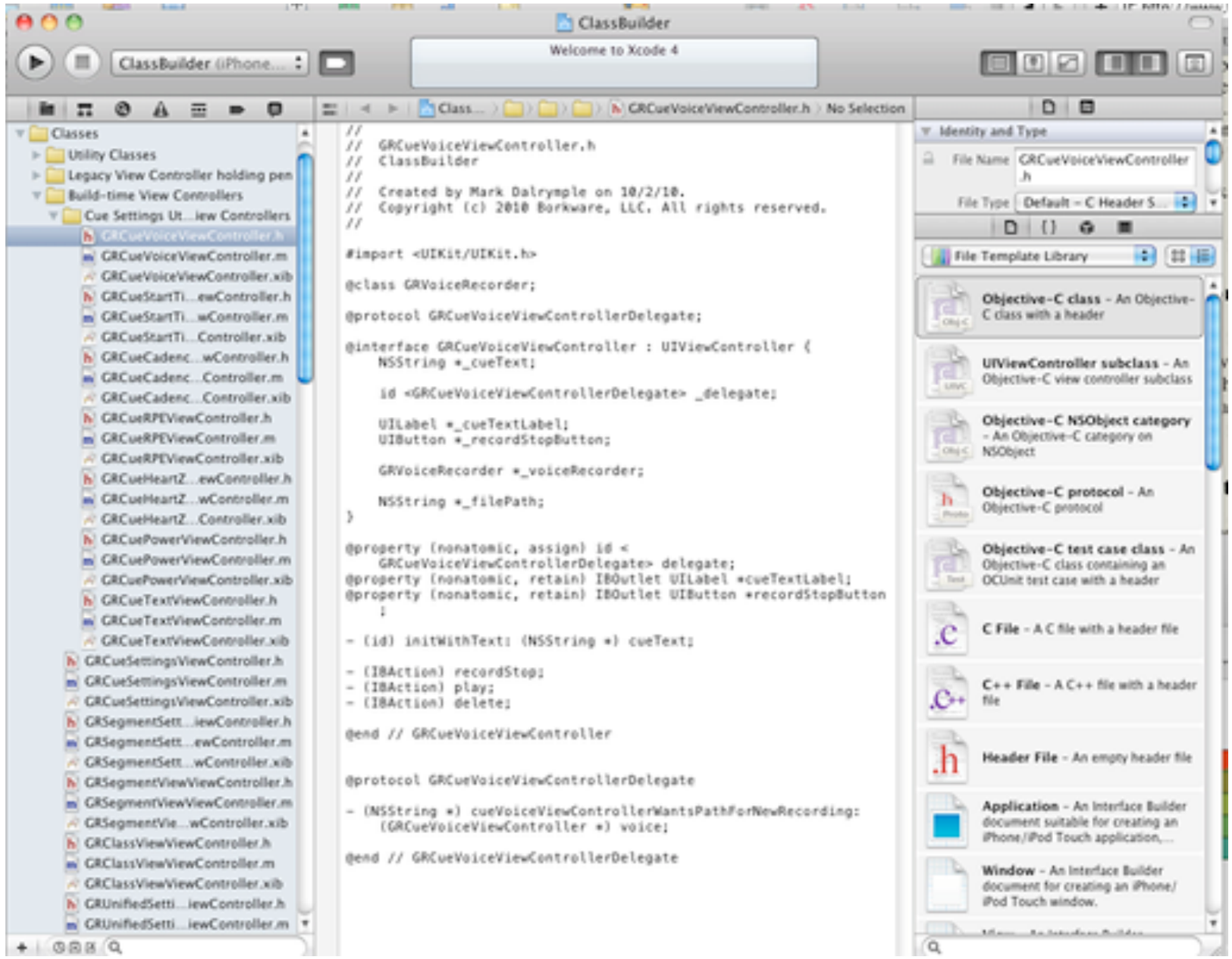
- Back up at-risk data
  - Especially configuration data
- Make sure your backups are good
- Know what a restore will entail

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

- Turn on Warnings (and fix them!)
  http://bit.ly/bored-zo-warnings
- Run the Static Analyzer
- Vacuum the database
- Remove Prefs / Application Support.
  I like RooSwitch.
- Check the hardware
- Wave the Dead Chicken

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

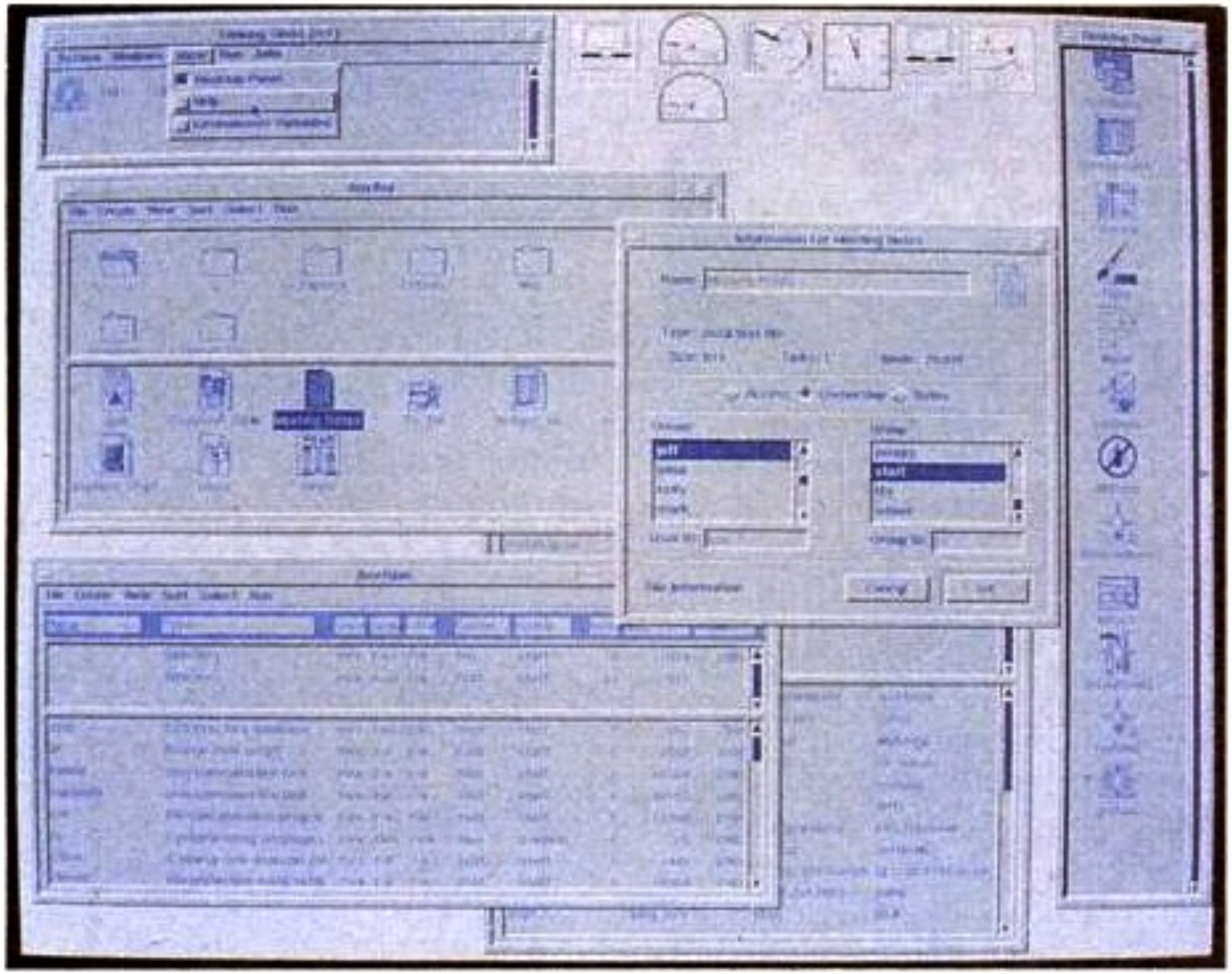
Keep After it

Caveman Debugging

Caveman Debugging

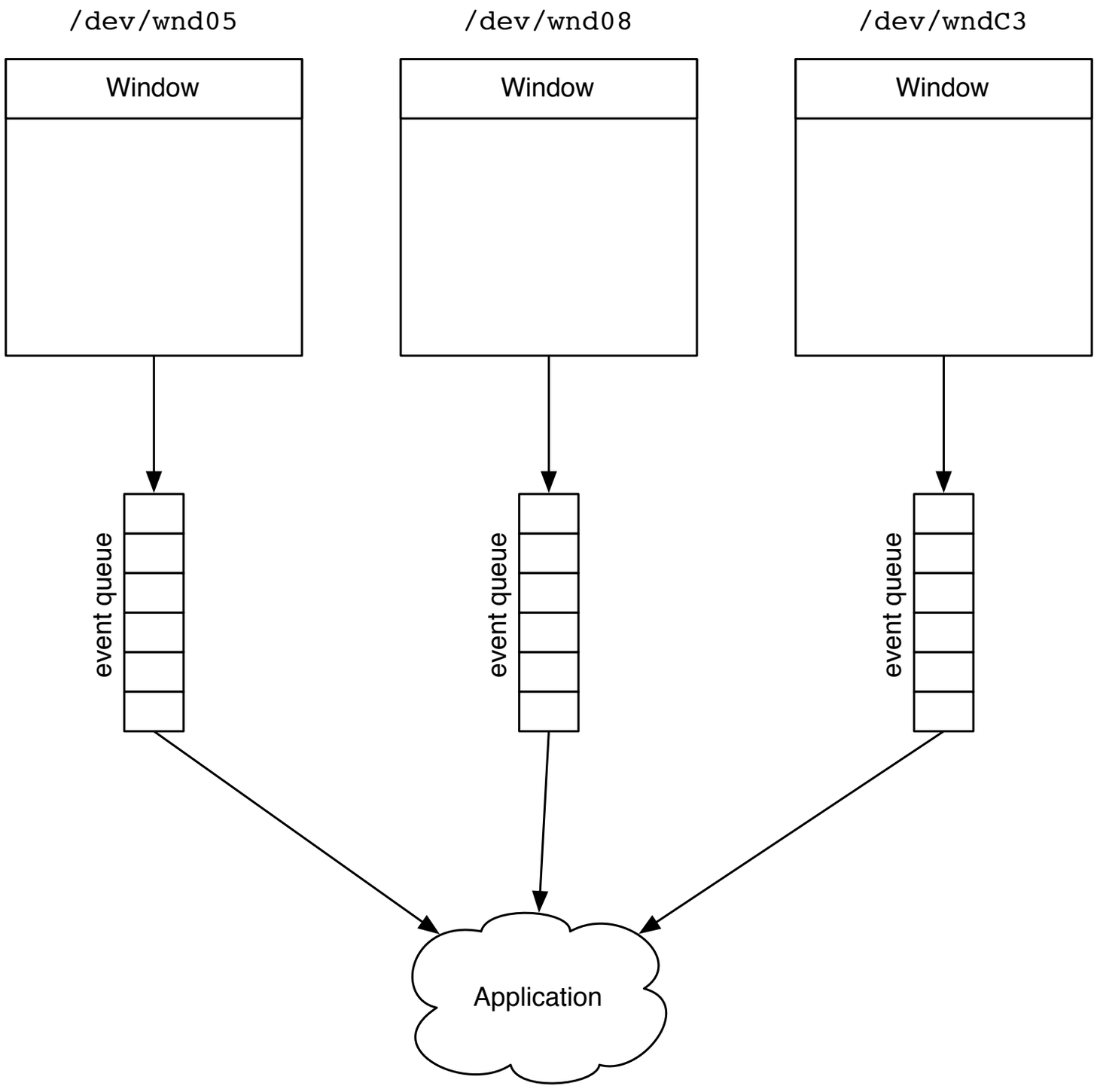
- Embrace the log
- Most useful if you have fast turnaround
- Handy for learning new API
- Don’t over-do it

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



The Debugger

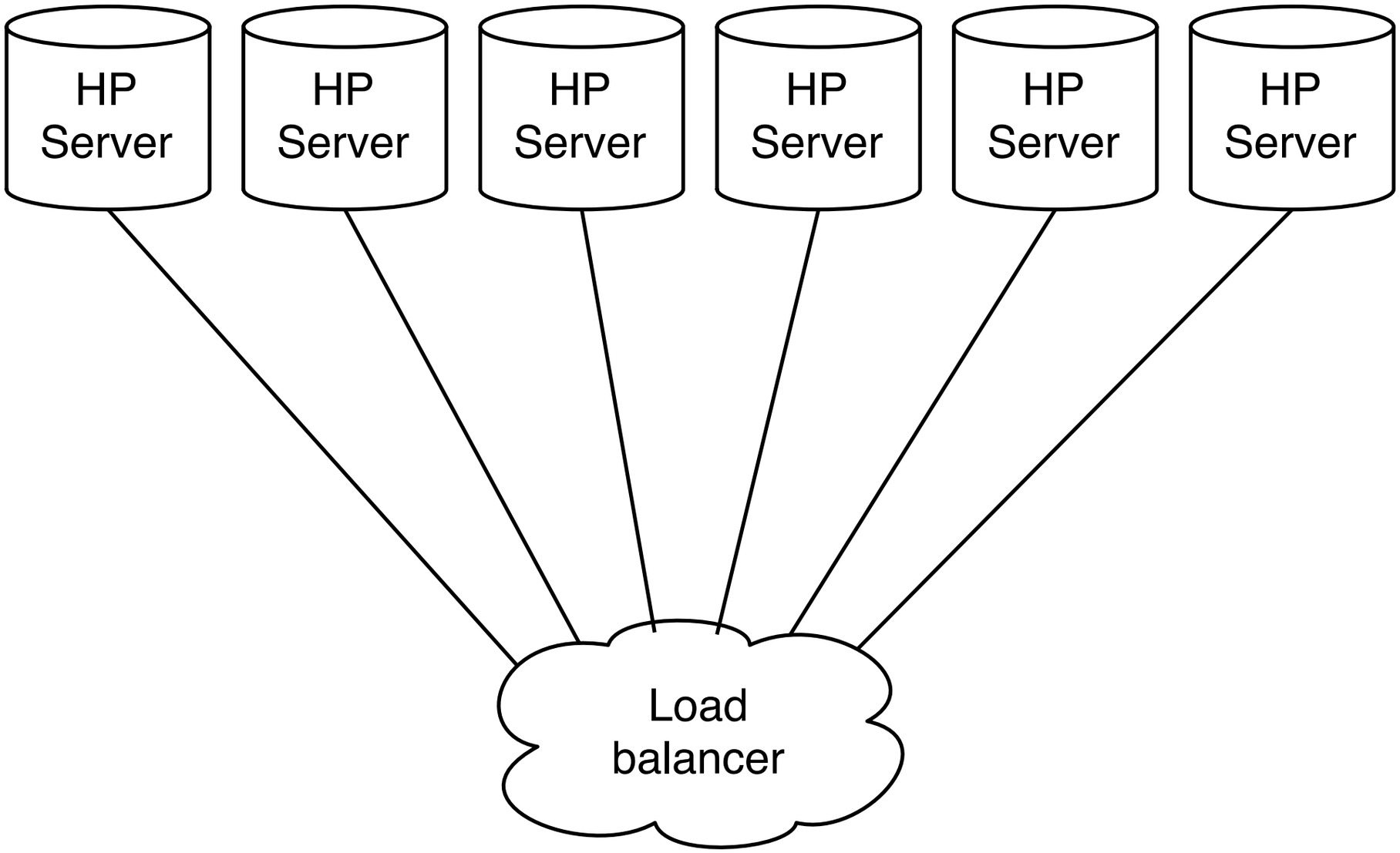
The Debugger

- 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

Server Crash #1



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
```

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`

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

gdb Command Line in Xcode

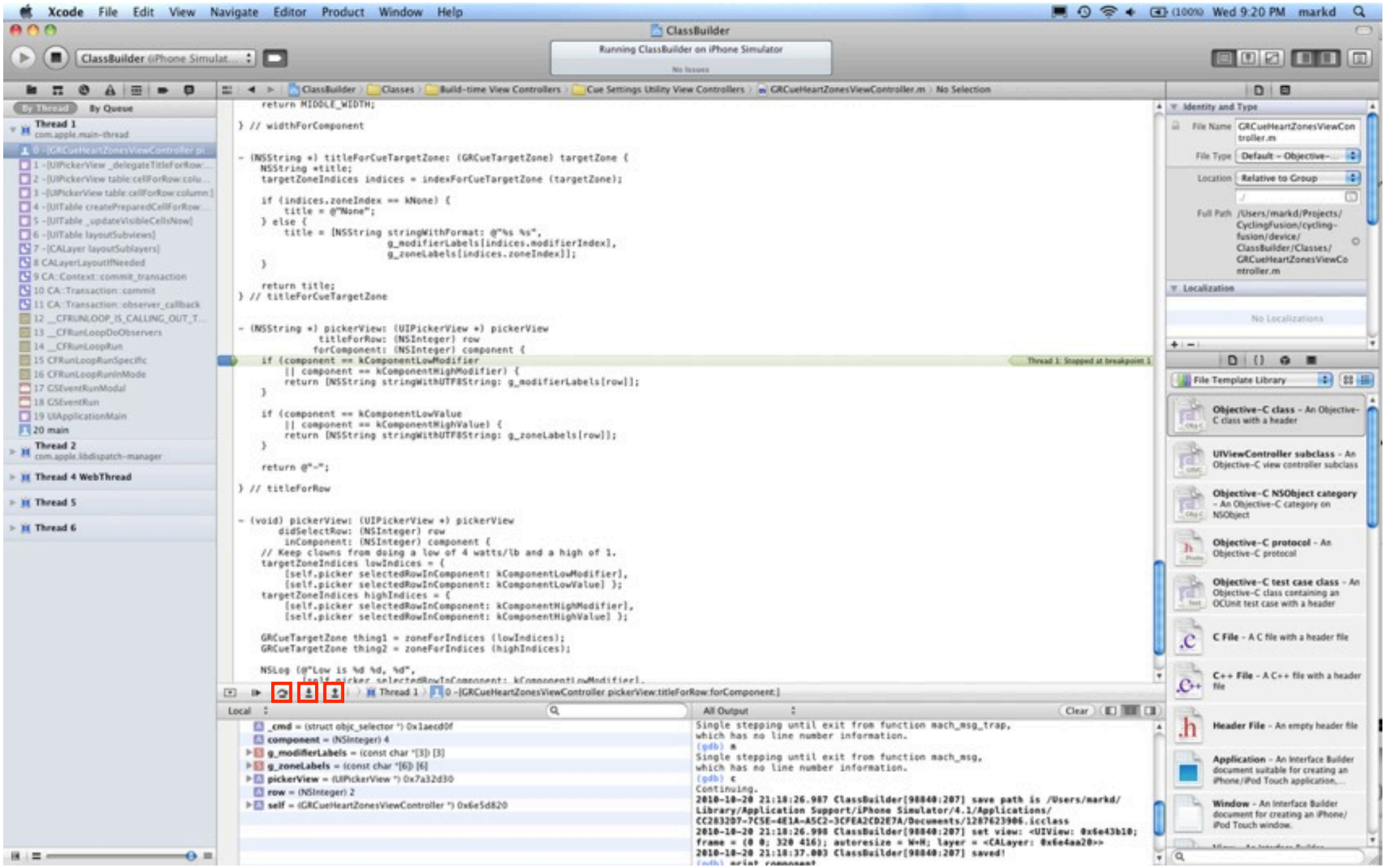
```
            forComponent: (NSInteger) component {
    if (component == kComponentLowModifier                    Thread 1: Stopped at breakpoint 1
        || component == kComponentHighModifier) {
        return [NSString stringWithUTF8String: g_modifierLabels[row]];
    }

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

Thread 1 〉 0 -[GRCueHeartZonesViewController pickerView:titleForRow:forComponent:]

All Output Clear

```
which has no line number information.
(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?



Useful Commands

- `(gdb) continue`
- `(gdb) step`
- `(gdb) next`
- `(gdb) return`
- `(gdb) until`

More Useful Commands

- `(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 = 0x0
}
```

- (gdb) set node->next->theChar = ‘q’

Displaying Arrays

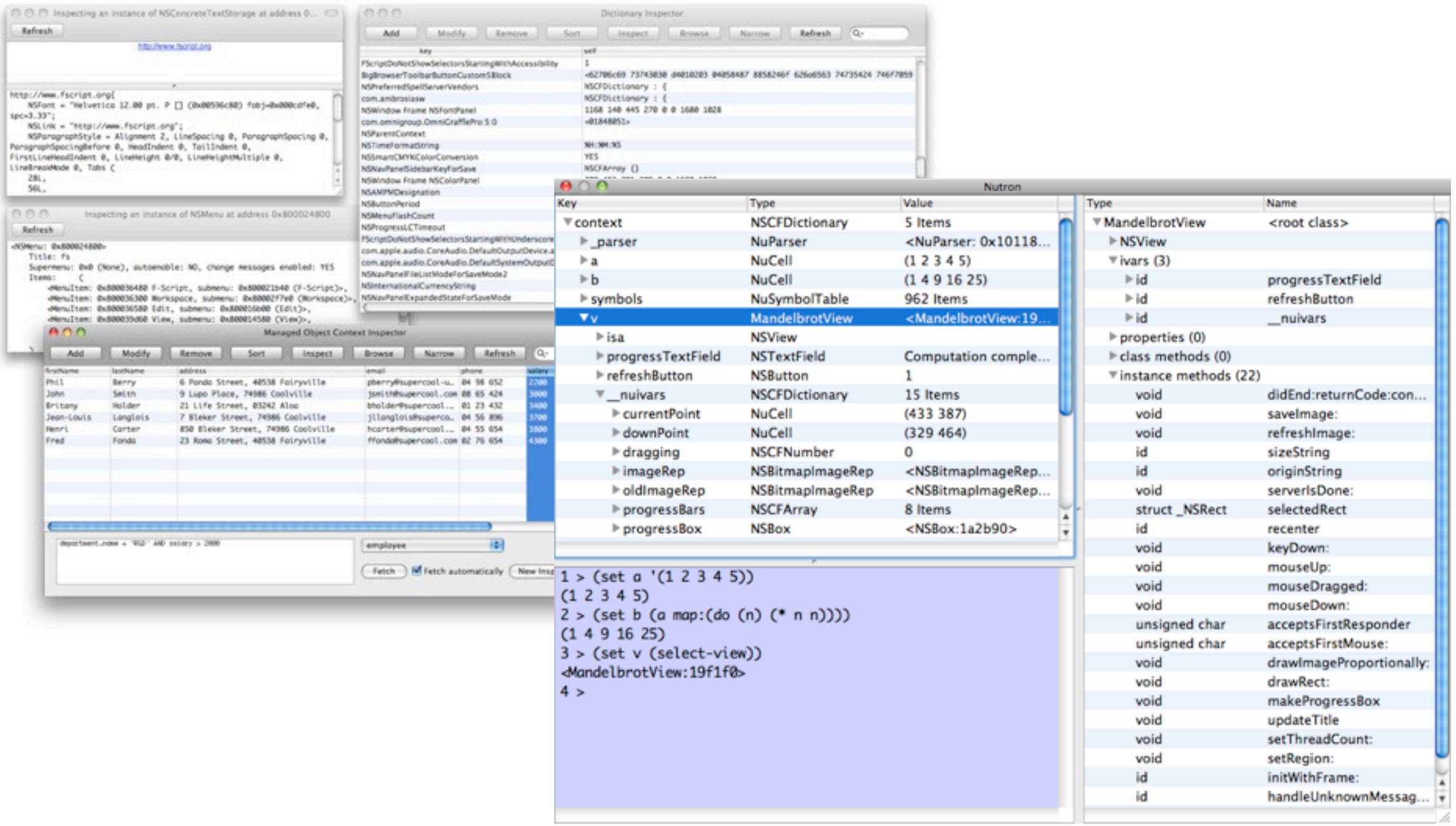
```
int imsg[] = { 78, 111, 119,
               32, 72, 105, 114, 105, 110, 103, 0};

(gdb) print {int} imsg @ 10

$2 = {78, 111, 119, 32, 72, 105, 114, 105, 110, 103}

(gdb) print {int}(imsg + 3)@2
$3 = {32, 72}

(gdb) print/c {int} imsg @ 10
$4 = {78 'N', 111 'o', 119 'w', 32 ' ',
      72 'H', 105 'i', 114 'r',
      105 'i', 110 'n', 103 'g'}
```

Other Tools

Other Tools

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

Inspectors

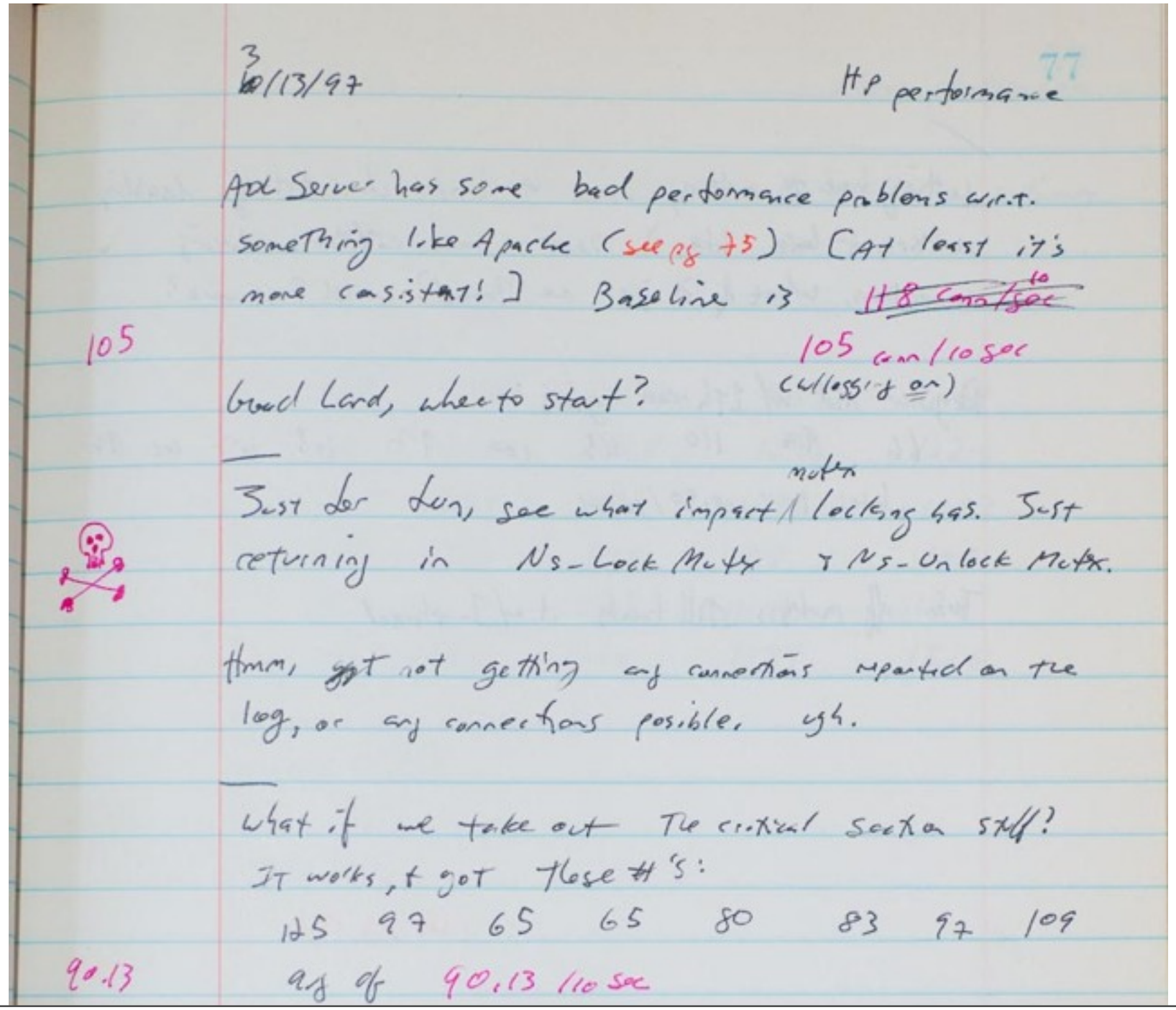


http://bit.ly/trynutron

When I'm Having Problems

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

Analogs



3
10/13/97 HP performance 77

Apache Server has some bad performance problems w.r.t.
something like Apache (see pg 75) [At least it's
more consistant!] Baseline is ~~HP Connects~~ to
105 105 conn/10sec
Good Lord, where to start? (w/keepalives on)

Just for fun, see what impact locking has. Just mutex
returning in Ns_LockMutex & Ns_UnlockMutex.

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

What if we take out the critical sockets 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"

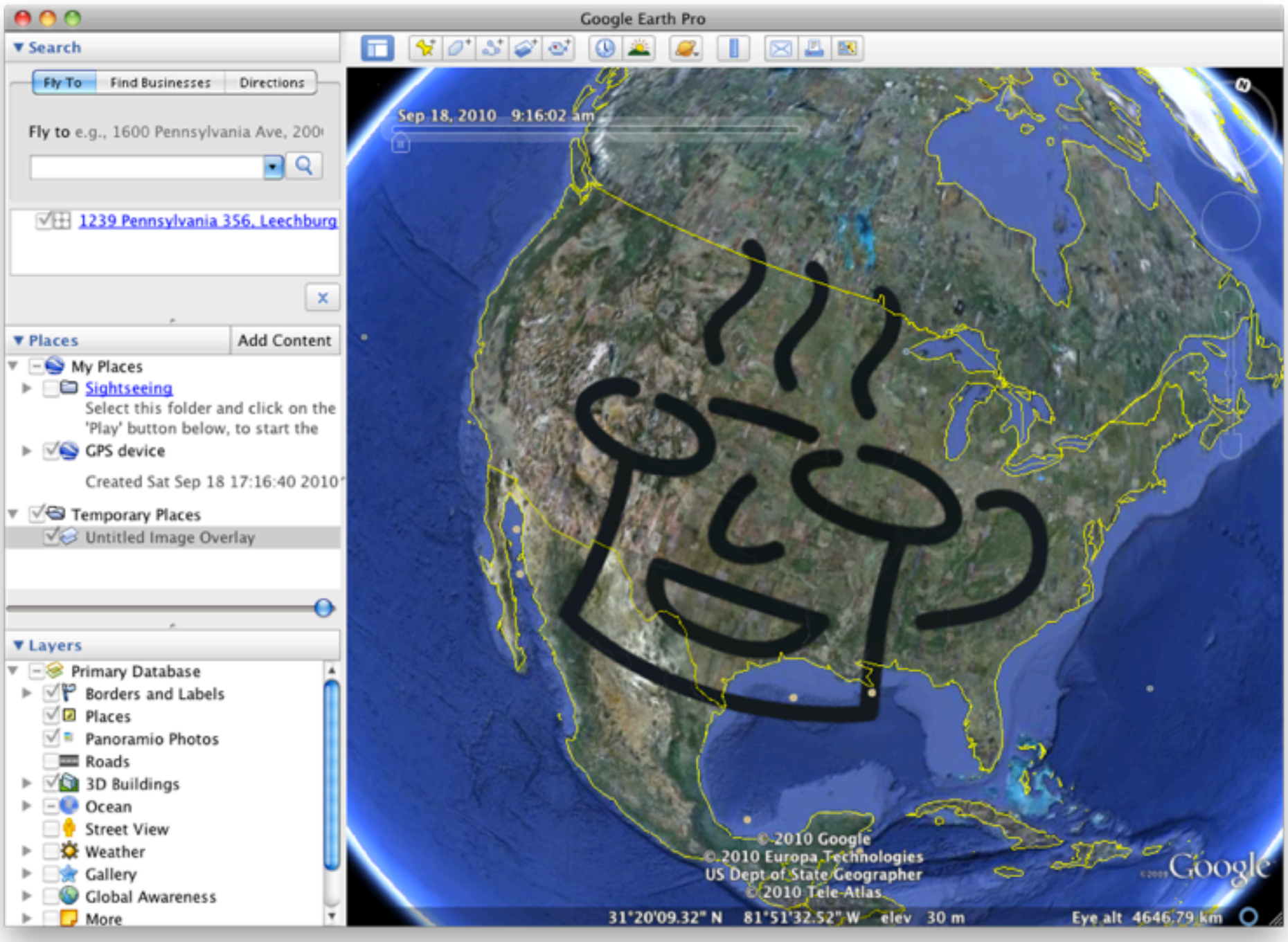
Takes an NSURLRequest
Has a delegate
- connection:willSendRequest:redirectResponse: lets you make a new request.
    ??? does it get called with the first request?

What about other APIs

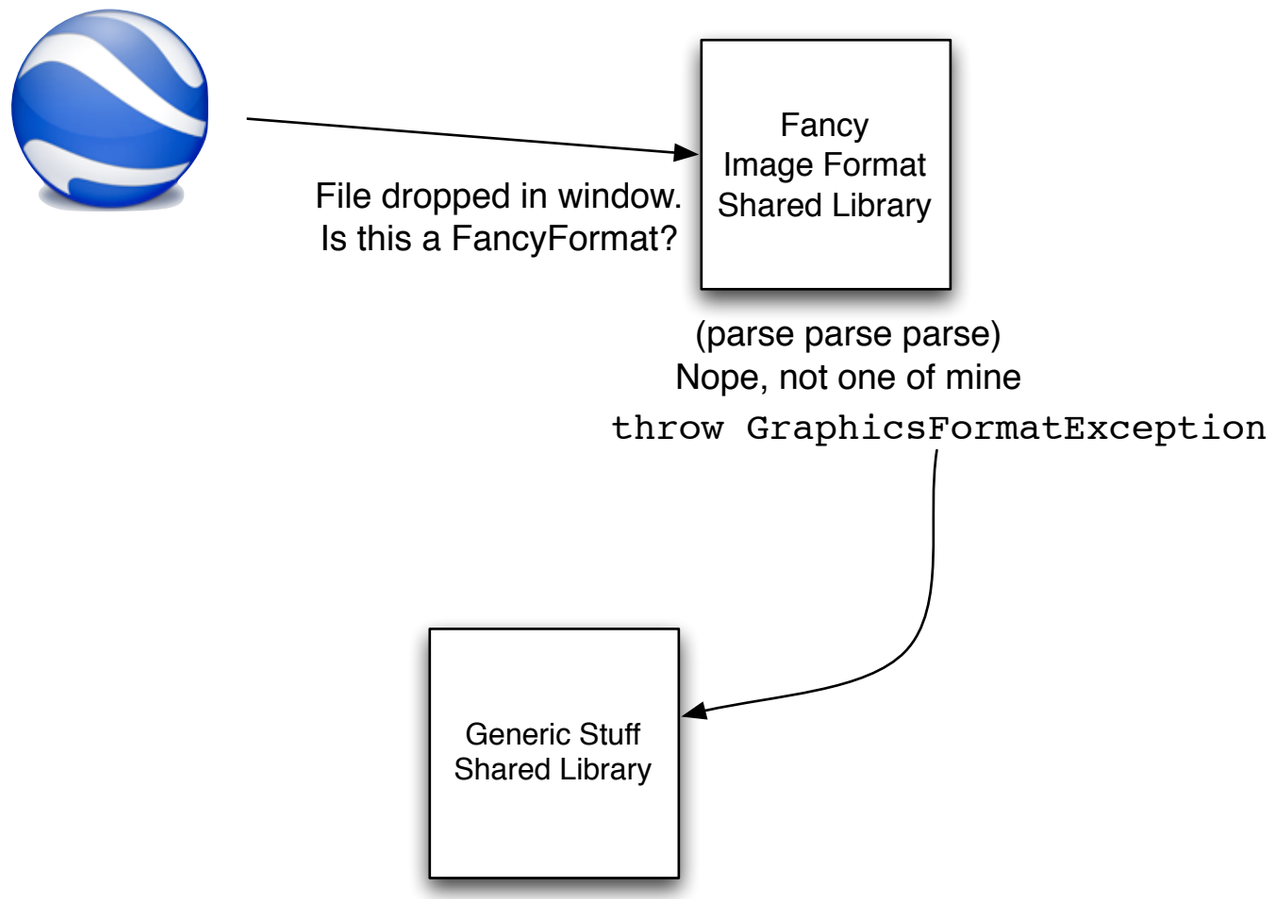
When I'm Still Having Problems

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

The Case of the Thrown Images

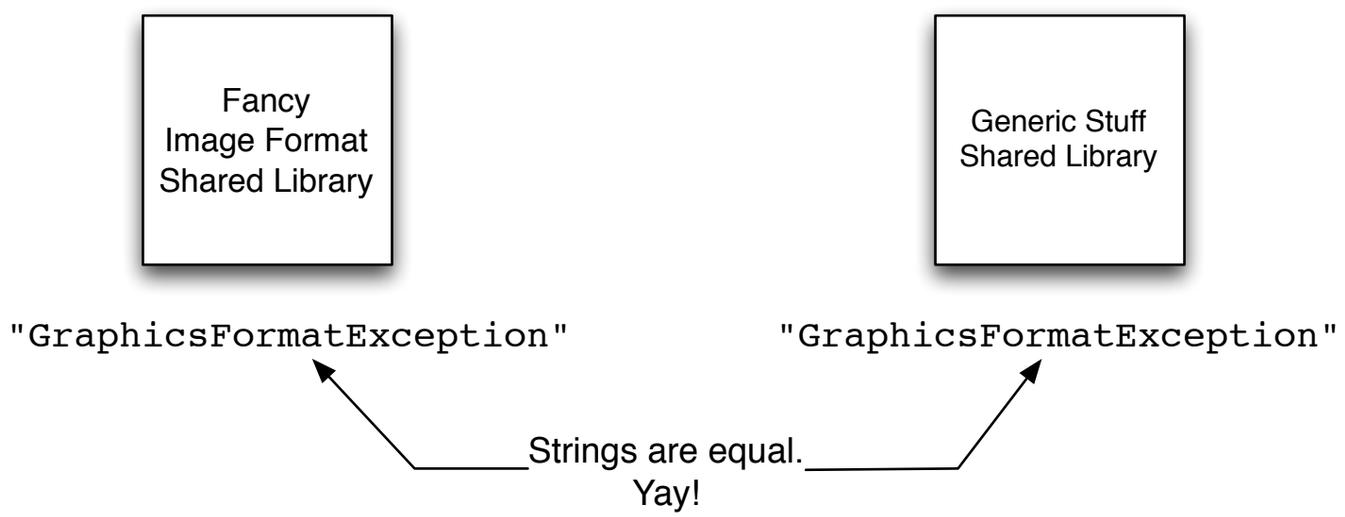


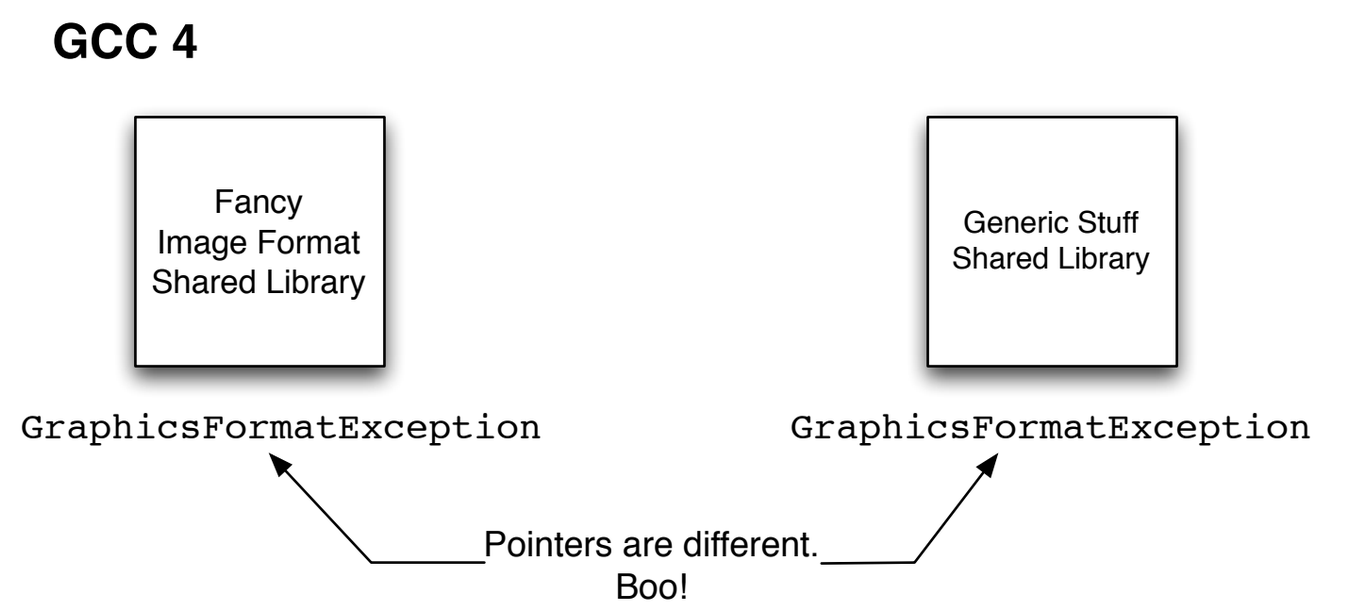
What, but not the Why




```
try {
    ...
} catch (GraphicsFormatException e) {
    See if it's a regular image type
} catch (...) {
    Give up
}
```

Why


GCC 3
Fancy
Image Format
Shared Library
Generic Stuff
Shared Library
"GraphicsFormatException"
"GraphicsFormatException"
Strings are equal.
Yay!


GCC 4
Fancy
Image Format
Shared Library
Generic Stuff
Shared Library
GraphicsFormatException
GraphicsFormatException
Pointers are different.
Boo!

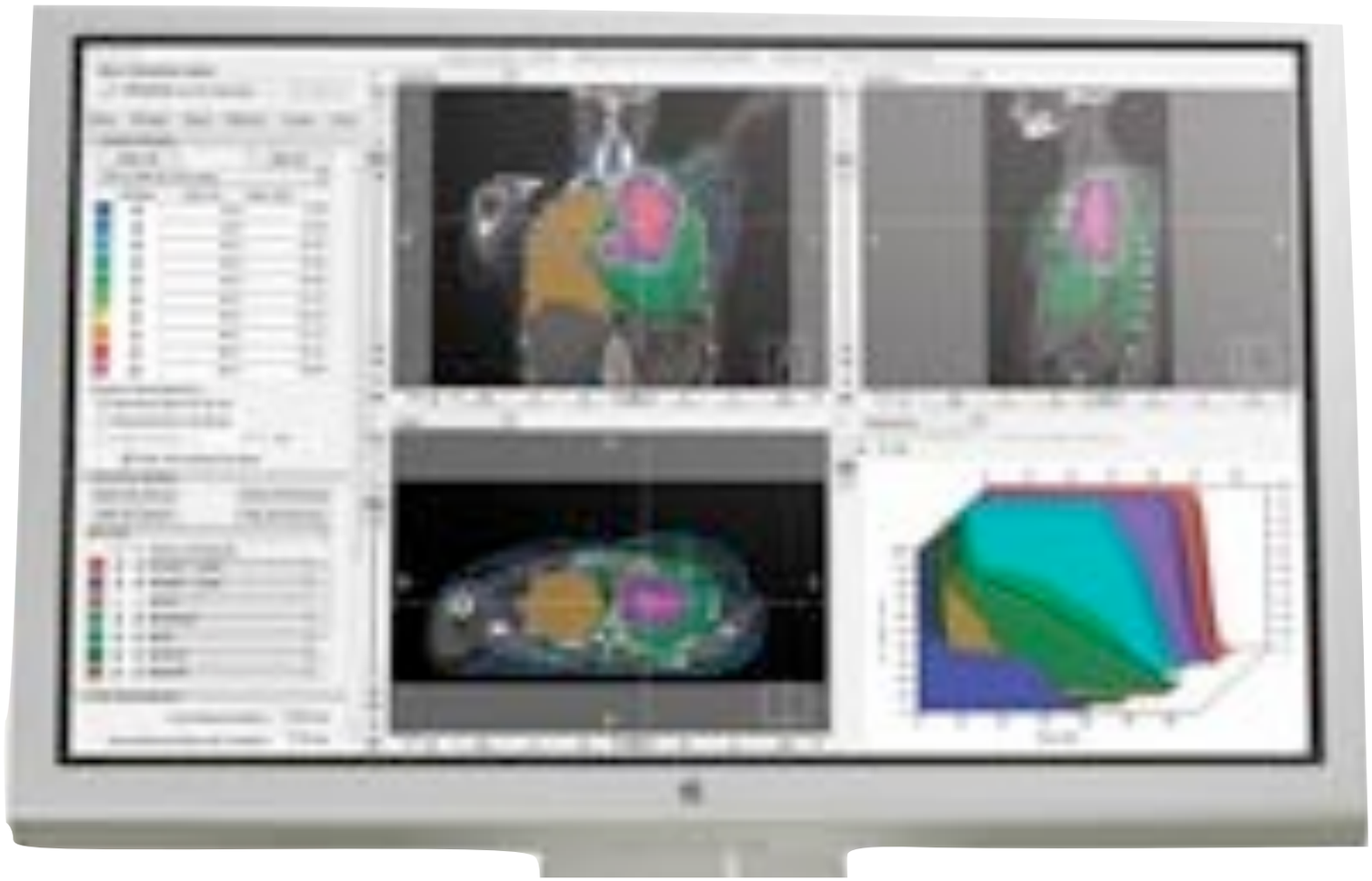
Optimizing your build time

I hate waiting

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

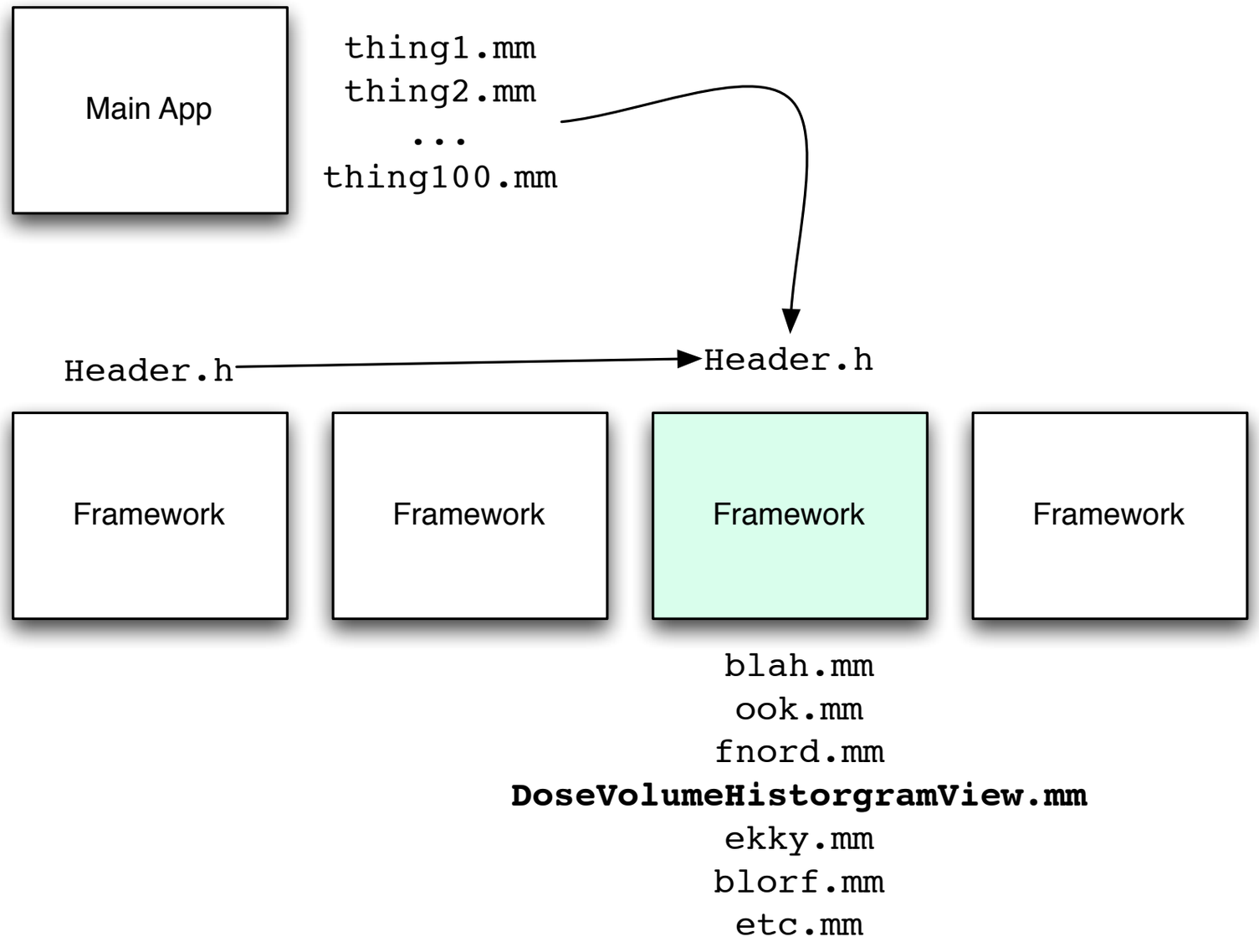
The App

- Here's a Big App you've just met
- Here's a Bug

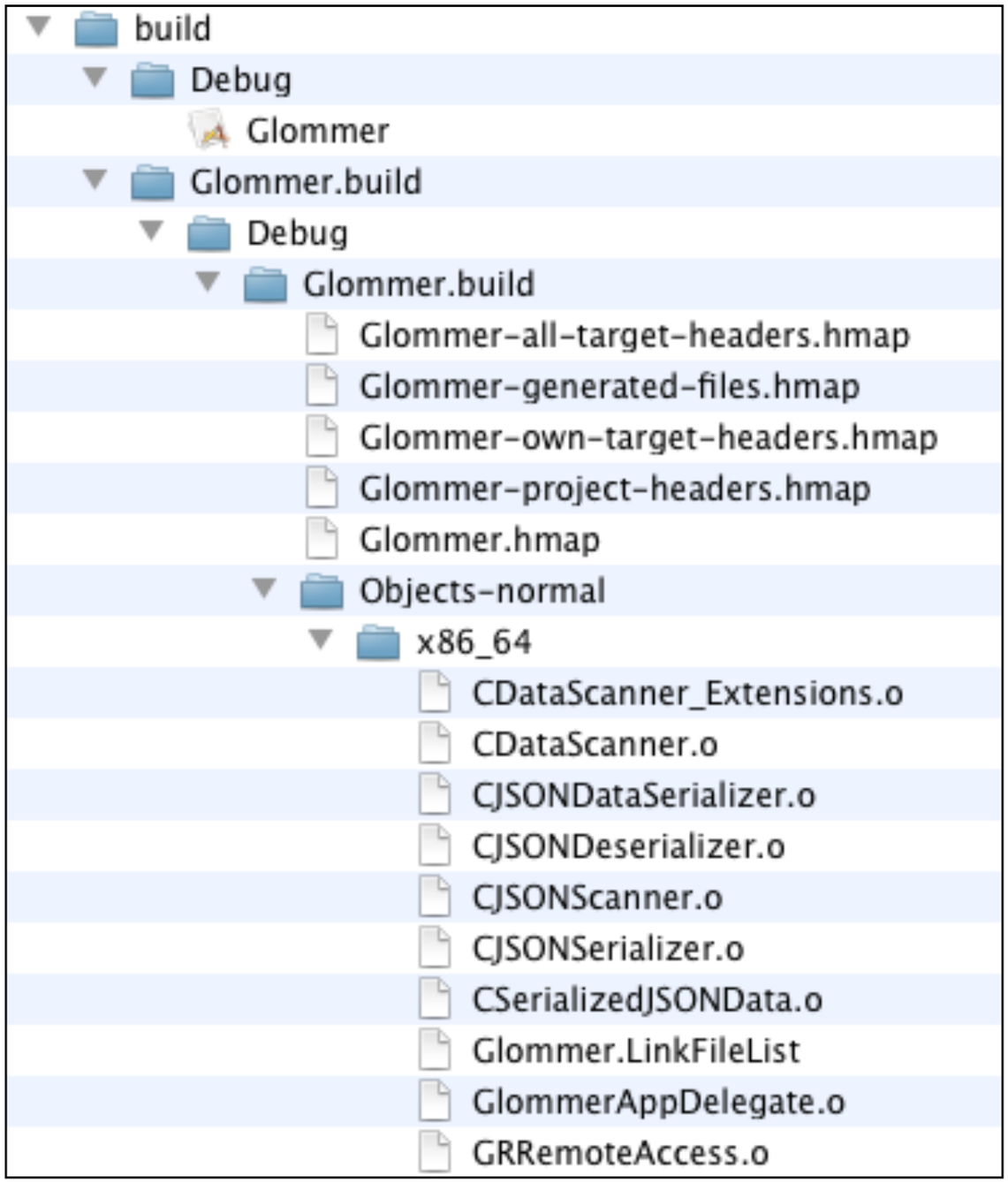


The Problem

- Build times were horrible



Inside the build directory



The Solution

Edit the code. Change the header file.

```
% find build/Glommer.build -name “*.o” -exec touch {} \;
```

Prevents everything from rebuilding

```
% touch Framework/DoseVolumnHistorgramView.m

% xcodebuild -configuration Debug -target Framework

% ./build/Debug/Glommer.app/Contents/MacOS/Glommer
```

!f !t !x !.

Second Problem - gdb Cooldown

```
Reading symbols for shared libraries ..................... done
Reading symbols for shared libraries ............... done
Reading symbols for shared libraries .......... done
Reading symbols for shared libraries ........................... done
Reading symbols for shared libraries ................ done
Reading symbols for shared libraries ..................... done
Reading symbols for shared libraries . done
Reading symbols for shared libraries ... done
Reading symbols for shared libraries . done
Reading symbols for shared libraries ................. done
Reading symbols for shared libraries ....................... done
Reading symbols for shared libraries ...... done
Reading symbols for shared libraries . done
Reading symbols for shared libraries ..................... done
Reading symbols for shared libraries . done
Reading symbols for shared libraries ............. done
Reading symbols for shared libraries ................ done
Reading symbols for shared libraries ............. done
Reading symbols for shared libraries ..................... done
```

The Solution

- Keep gdb running
- Rebuild in another shell
- `(gdb) run`

```
`/Glommer.app/Contents/Frameworks/
Snork.framework' has changed; re-reading
symbols.
```

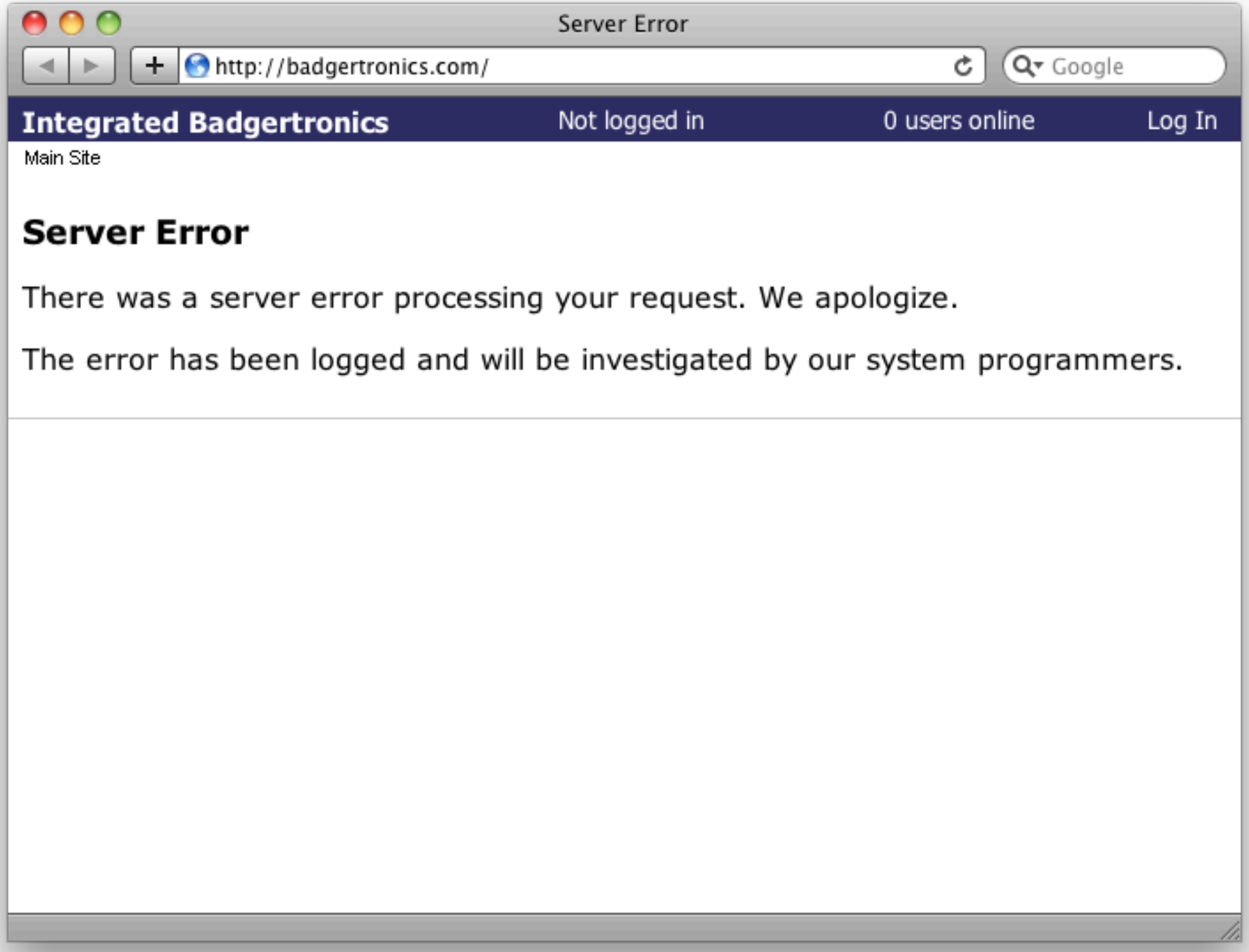
The Most Important Questions

What Changed?

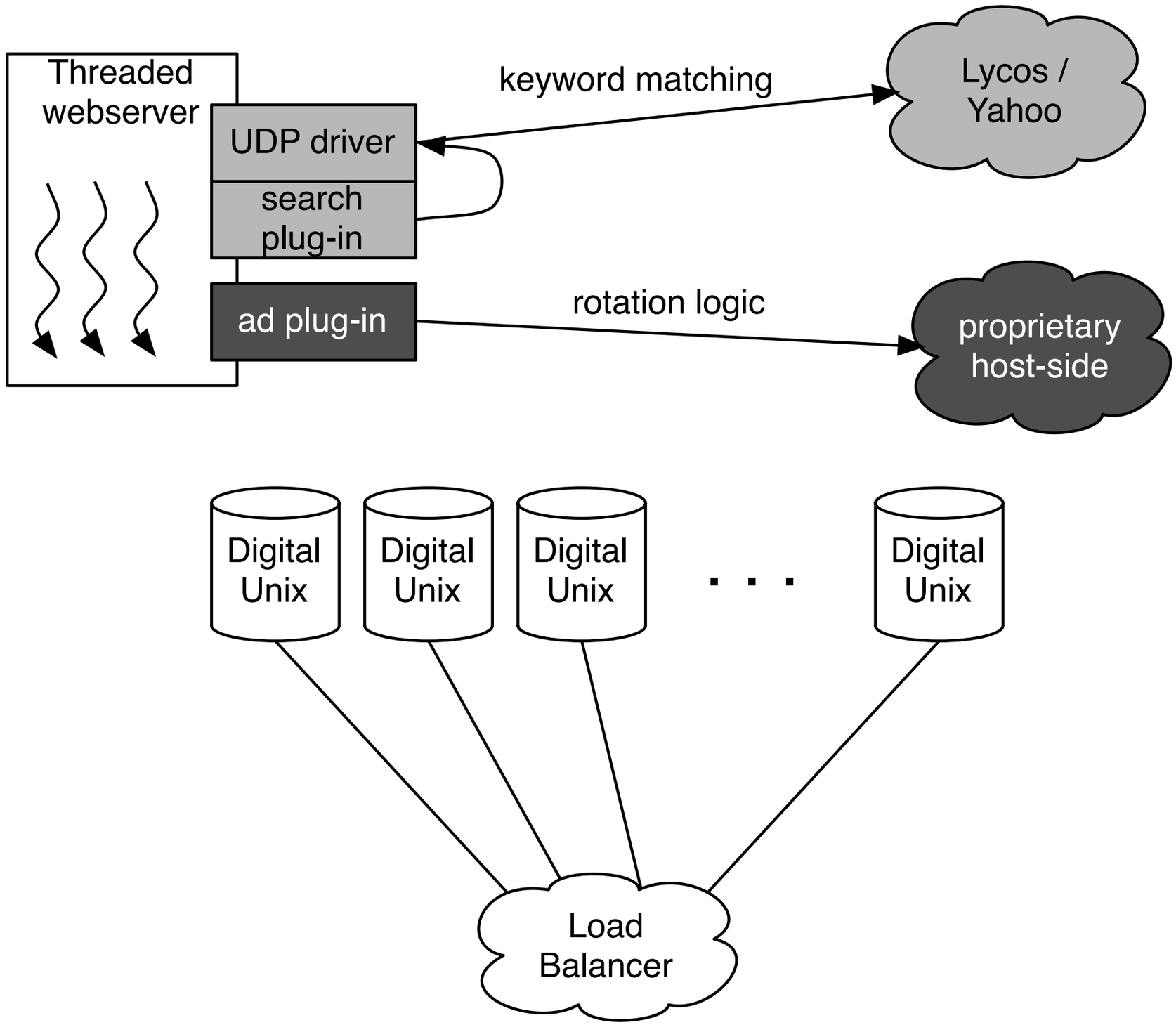
- What's new?
- What's changed?
- What's different?

The LSD*

*Lurking Software Defect



Server Crash #2



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

NSCoder Night

Your time to code!

WWDC

2C SecondConf

NSConference

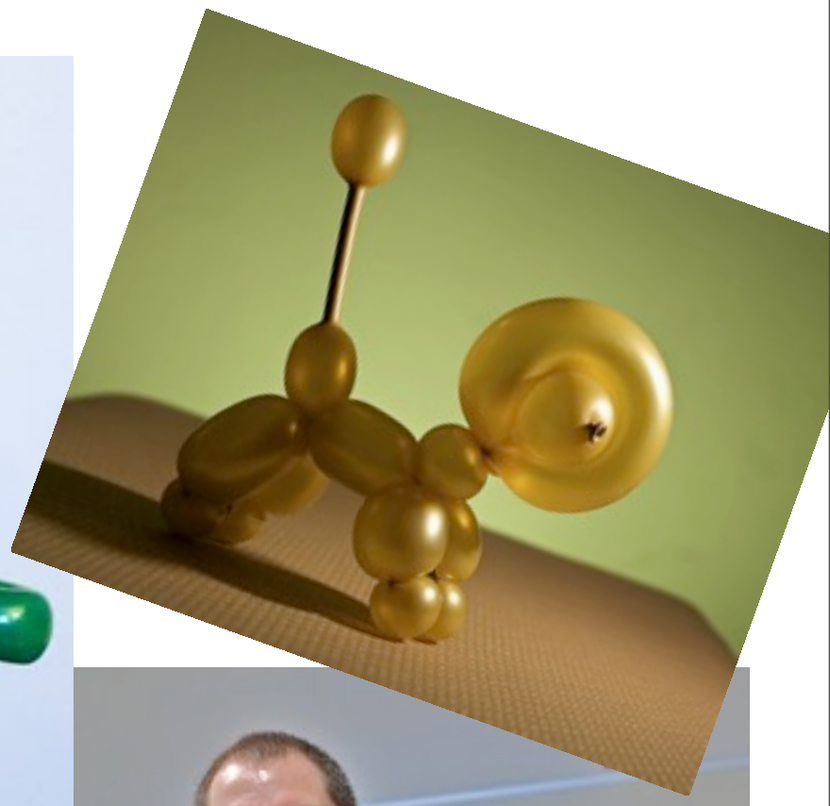
The New York iPhone Software Developers Meetup

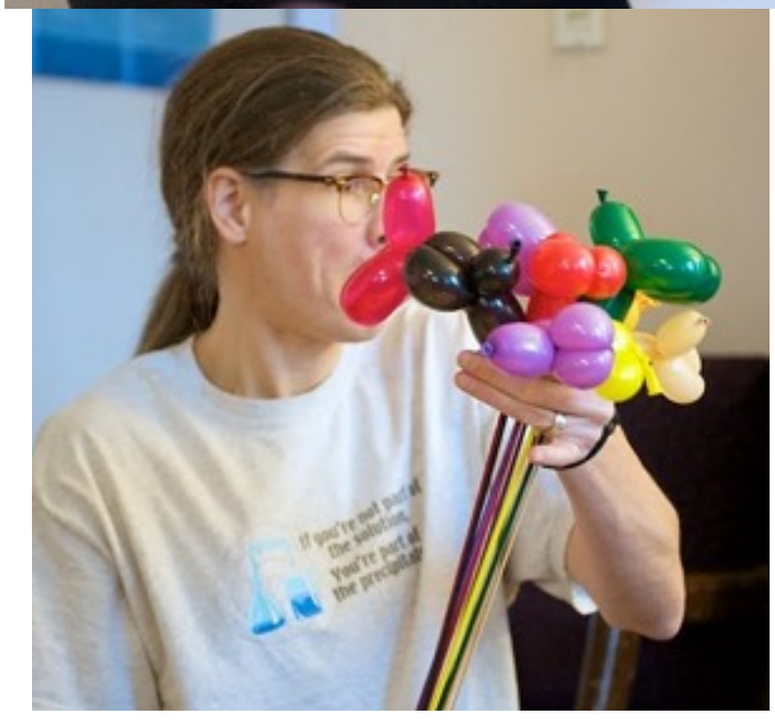
What does it all mean?

A little bit of philosophy

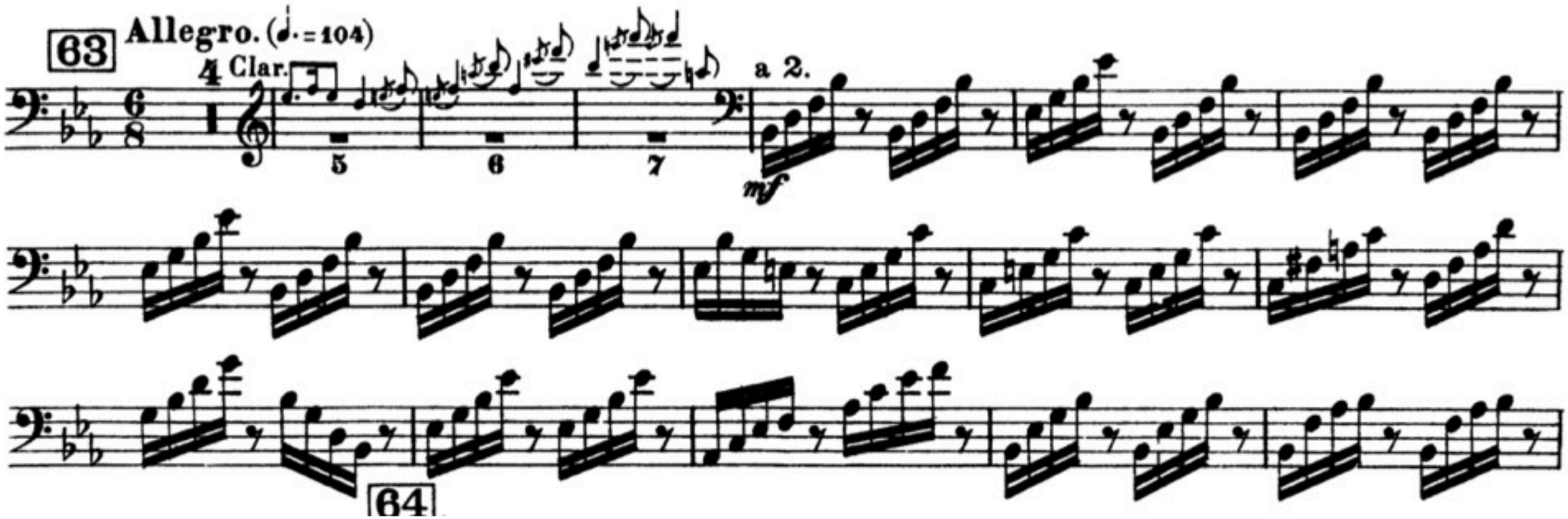
- Don't Panic
- Own the problem
- Keep after it
- Optimize your tools and yourself
- Be Excellent to each other
- Practice, learn, and reflect

A little bit of silliness

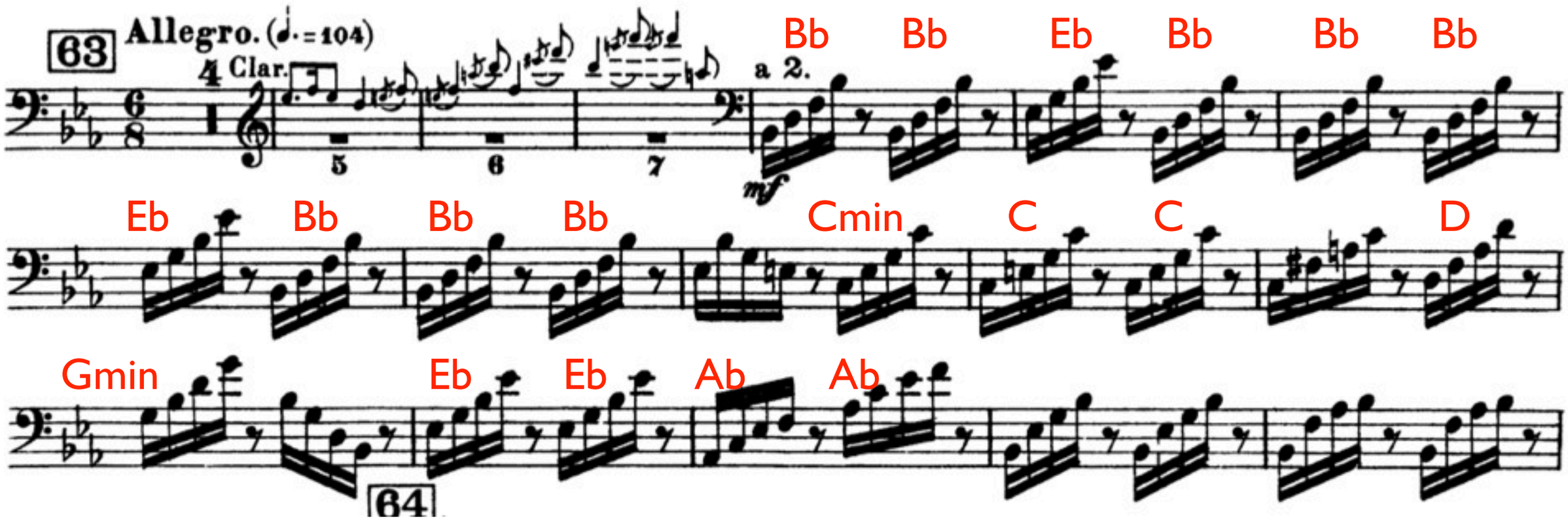




A little bit of music



A little bit of music



That's it.

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

Thank you.