System Trace in Depth

Explore the deep end of the Instruments pool

Session 411

Chad Woolf Performance Tools Engineer
Joe Grzywacz Performance Tools Engineer
Last Year...

Session 412 - Time Profiling in Depth
Last Year...
Session 412 - Time Profiling in Depth

WWDC 2015
Multi-core
Multi-core

Get more done
Multi-core

Get more done

System load changes performance
Multi-core

Get more done
System load changes performance
High system load increases
• Preemption
• Lock contention
• Virtual memory activity
System Trace in Depth
Agenda
System tracing in depth
Agenda

System tracing in depth

System Trace for Apps
Agenda

System tracing in depth

System Trace for Apps
Using System Trace
• Threading
• Signposts
• Virtual Memory
• Best Practices
Choose a profiling template for: cwoolf3 > All Processes

- Standard
- Custom
- Recent

Blank  Activity Monitor  Allocations  Cocoa Layout  Core Animation  Core Data
Counters  Energy Log  File Activity Leaks  Metal System Trace  Network
OpenGL ES Analysis  System Trace  System Usage  Time Profiler  Zombies

System Trace
Provides comprehensive information about system behavior by showing when threads are scheduled, and showing all their transitions from user into system code via either system calls or memory operations.

Open an Existing File...  Cancel  Choose
System Trace
Provides comprehensive information about system behavior by showing when threads are scheduled, and showing all their transitions from user into system code via either system calls or memory operations.
System Trace
System Trace

Records a kernel trace
System Trace

Records a kernel trace

- Scheduling activity
System Trace

Records a kernel trace

• Scheduling activity
• System calls
System Trace

Records a kernel trace

• Scheduling activity
• System calls
• Virtual memory operations
System Trace

Records a kernel trace

• Scheduling activity
• System calls
• Virtual memory operations
System Trace

Records a kernel trace
• Scheduling activity
• System calls
• Virtual memory operations

Windowed Mode in Instruments 8
System Trace

Records a kernel trace
• Scheduling activity
• System calls
• Virtual memory operations

Windowed Mode in Instruments 8
• Keeps last ~5 sec of data
System Trace

Records a kernel trace
• Scheduling activity
• System calls
• Virtual memory operations

Windowed Mode in Instruments 8
• Keeps last ~5 sec of data
• Gives you more time to reproduce
Points of Interest
Points of Interest

You tell Instruments what’s interesting
Points of Interest

You tell Instruments what’s interesting

Signposts
Points of Interest

You tell Instruments what’s interesting

Signposts

Classic:

```c
syscall(SYS_kdebug_trace, ...
```
Points of Interest

You tell Instruments what’s interesting

Signposts

Classic:

```c
syscall(SYS_kdebug_trace, ...)
```

iOS 10 / macOS Sierra / tvOS 10 / watchOS 3:

```c
kdebug_signpost
kdebug_signpost_start
kdebug_signpost_end
```
Points of Interest

Events

Indicate an interesting point in time
Arbitrary code (0 - 16383)
4 uintptr_t arguments

// Point of Interest
func mouseDown(_ event: NSEvent) {
    // Emit a signpost for Instruments
    kdebug_signpost(5, 0, 0, 0, 0)
}

Points of Interest

Events

Indicate an interesting point in time

Arbitrary code (0 - 16383)

4 uintptr_t arguments

```swift
// Point of Interest
func mouseDown(_ event: NSEvent) {
    // Emit a signpost for Instruments
    kdebug_signpost(5, 0, 0, 0, 0, 0)
}
```
Points of Interest

Events

Indicate an interesting point in time

Arbitrary code (0 - 16383)

4 uintptr_t arguments

```swift
// Point of Interest
func mouseDown(_ event: NSEvent) {
    // Emit a signpost for Instruments
    kdebug_signpost(5, 0, 0, 0, 0)
}
```
Points of Interest

Events

Indicate an interesting point in time

Arbitrary code (0 - 16383)

4 uintptr_t arguments

```swift
// Point of Interest
func mouseDown(_ event: NSEvent) {
    // Emit a signpost for Instruments
    kdebug_signpost(5, 0, 0, 0, 0)
}
```
Points of Interest
Named codes
Points of Interest
Named codes

Options
- Color using last argument

Signpost Code Names
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
</table>

Match Signpost Intervals By:
- Code
- Code and First Argument
- Code and Thread
Points of Interest
Named codes
Regions of Interest
States or actions

Indicate an interesting range of time

Arbitrary code

Four integer/pointer arguments at start and end

```c
// Timing an activity (code 10 - "Start Up")
-(void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
    kdebug_signpost_start(10, 0, 0, 0, 0);
    [self loadAssets];
    kdebug_signpost_end(10, 0, 0, 0, 0);
}
```
Regions of Interest
States or actions

Indicate an interesting range of time
Arbitrary code
Four integer/pointer arguments at start and end

// Timing an activity (code 10 - "Start Up")
-(void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
    kdebug_signpost_start(10, 0, 0, 0, 0, 0);
    [self loadAssets];
    kdebug_signpost_end(10, 0, 0, 0, 0, 0);
}
Regions of Interest
States or actions

Indicate an interesting range of time
Arbitrary code
Four integer/pointer arguments at start and end

// Timing an activity (code 10 - "Start Up")
-(void)applicationDidFinishLaunching:(NSNotification *)aNotification
{
    kdebug_signpost_start(10, 0, 0, 0, 0);
    [self loadAssets];
    kdebug_signpost_end(10, 0, 0, 0, 0);
}````
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Mouse down</td>
</tr>
<tr>
<td>10</td>
<td>Loading Assets</td>
</tr>
</tbody>
</table>

Options
- Color using last argument

Match Signpost Intervals By:
- Code
- Code and First Argument
- Code and Thread
### Signpost Code Names

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Mouse down</td>
</tr>
<tr>
<td>10</td>
<td>Loading Assets</td>
</tr>
</tbody>
</table>

### Match Signpost Intervals By:

- Code
- Code and First Argument
- Code and Thread
Points of Interest

Matching rule: Code and First Argument

Concurrent
Asynchronous

// Start the download (code 20 - "URL Download")
-(NSURLSessionDownloadTask *)startURLDownload: (NSURL *) url {
    NSURLSessionDownloadTask *dlTask = [self.downloadSession downloadTaskWithURL:url];
    kdebug_signpost_start(20, (uintptr_t)dlTask, 0, 0, 0);
    [dlTask resume];
    return dlTask;
}

-(void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)dlTask
didCompleteWithError:(nullable NSError *)error {
    kdebug_signpost_end(20, (uintptr_t)dlTask, 0, 0, 0);
}
Points of Interest

Matching rule: Code and First Argument

Concurrent

Asynchronous

// Start the download (code 20 - "URL Download")
-
(NSURLSessionDownloadTask *)startURLDownload: (NSURL *) url {
  NSURLSessionDownloadTask *dlTask = [_urlSession downloadTaskWithURL:url];
  kdebug_signpost_start(20, (uintptr_t)dlTask, 0, 0, 0);
  [dlTask resume];
  return dlTask;
}

- (void)NSURLSession:(NSURLSession *)session task:(NSURLSessionTask *)dlTask
didCompleteWithError:(nullable NSError *)error {
  kdebug_signpost_end(20, (uintptr_t)dlTask, 0, 0, 0);
}
// Timing concurrent "loops" (code 30 - "Loading Chunk")
- (void)loadAssets {
    dispatch_apply(4, dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0), ^(size_t i) {
        kdebug_signpost_start(30, 0, 0, 0, 0);
        _loadAssetChunk(i);
        kdebug_signpost_end(30, 0, 0, 0, 0);
    });
}
Matching rule: Code and Thread

Concurrent

"Loop" timing

```c
// Timing concurrent "loops" (code 30 - "Loading Chunk")
- (void)loadAssets {
    dispatch_apply(4, dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0), ^(size_t i) {
        kdebug_signpost_start(30, 0, 0, 0, 0);
        _loadAssetChunk(i);
        kdebug_signpost_end(30, 0, 0, 0, 0);
    });
}
```
Points of Interest
Matching rule: Code and Thread

Concurrent

"Loop" timing

// Timing concurrent "loops" (code 30 - "Loading Chunk")
- (void)loadAssets {
    dispatch_apply(4, dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0), ^(size_t i) {
        kdebug_signpost_start(30, 0, 0, 0, 0);
        _loadAssetChunk(i);
        kdebug_signpost_end(30, 0, 0, 0, 0);
    });
}
**Signpost Code Names**

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Mouse down</td>
</tr>
<tr>
<td>10</td>
<td>Loading Assets</td>
</tr>
</tbody>
</table>

**Options**
- Color using last argument

**Match Signpost Intervals By:**
- Code
- Code and First Argument
- Code and Thread
### Options
- Color using last argument

### Signpost Code Names
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Mouse down</td>
</tr>
<tr>
<td>10</td>
<td>Loading Assets</td>
</tr>
</tbody>
</table>

### Match Signpost Intervals By:
- Code
- Code and First Argument
- Code and Thread
// Color by last argument
// 0 - Blue, 1 - Green, 2 - Purple, 3 - Orange, 4 - Red
-(void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(nullable NSError *)error {
    kdebug_signpost_end(20, (uintptr_t)task, 0, 0, (error) ? 4 : 1);
}

Points of Interest

Color using last argument

Pass/Fail

Frame overrun

Differentiation
Points of Interest

Color using last argument

Pass/Fail

Frame overrun

Differentiation

// Color by last argument

// 0 - Blue, 1 - Green, 2 - Purple, 3 - Orange, 4 - Red

-(void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(nullable NSError *)error {
    kdebug_signpost_end(20,(uintptr_t)task, 0, 0, (error) ? 4 : 1);
}
Points of Interest

Color using last argument

Pass/Fail
Frame overrun
Differentiation

// Color by last argument

// 0 - Blue, 1 - Green, 2 - Purple, 3 - Orange, 4 - Red

-(void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(nullable NSError *)error {
    kdebug_signpost_end(20,(uintptr_t)task, 0, 0, (error) ? 4 : 1);
}

NEW
### Points of Interest

#### Correlation

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>00:00:200</td>
<td>SignpostGame (1659)</td>
</tr>
<tr>
<td>00:00:250</td>
<td>__thread_selfid 0x108b6b</td>
</tr>
<tr>
<td>00:00:300</td>
<td>SignpostGame (1659)</td>
</tr>
<tr>
<td>00:00:350</td>
<td>__thread_selfid 0x108b88</td>
</tr>
<tr>
<td>00:00:400</td>
<td>SignpostGame (1659)</td>
</tr>
<tr>
<td>00:00:450</td>
<td>__thread_selfid 0x108b83</td>
</tr>
<tr>
<td>00:00:500</td>
<td>SignpostGame (1659)</td>
</tr>
</tbody>
</table>

#### Regions
- Launching Start: (0x0 0x0 0x0 0x0), End: (...)
- Load Asset Start: (0x0 0x0 0x0 0x0), End: ...
- Download Start: 0x7fcc9041263...
- Load Asset Start: (0x1 0x0 0x0 0x0), End: ...
- Download Start: 0x7fcc904...
- Load Asset Start: (0x2 0x0 0x0 0x0), End: ...
- Load Asset Start: (0x3 0x0 0x0 0x0), End: ...

#### Points
Points of Interest

Correlation
Graphasaurus 2
A legacy, reborn

Real world problems
New graphing style
Time profiled
Needs parallelism
• 5 ms per row
• Four rows
• 20 ms > 16 ms (60 fps)
Demo
Graphasaurus 2

Joe Grzywacz
Lock Contention
A side effect of system load
Lock Contention

Running (42.88 μs)
Lock Contention

Blocking
Lock Contention
Runnable

unlock_wait (13.75 µs)
Runnable (6.96 µs)
Lock Contention

Overview
Lock Contention

Overhead

Only 82% in Running
Lock Contention

Fixed
Lock Contention
Fixed

100% in Running
Preempted
Preempted

Involuntary

- Priority decayed
- High priority work runnable
Preempted

Involuntary
- Priority decayed
- High priority work runnable

Voluntary
- Spin locks
- thread_switch
Preempted

Involuntary
- Priority decayed
- High priority work runnable

Voluntary
- Spin locks
- thread_switch
Preempted

Involuntary
- Priority decayed
- High priority work runnable

Voluntary
- Spin locks
- thread_switch
Interrupted

Interrupt handler
Priority doesn’t matter
Brief
System Load
System Load

Details Active Threads

<table>
<thead>
<tr>
<th>Priority</th>
<th>Process</th>
<th>Thread</th>
<th>State</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>mach_kernel</td>
<td>0x711f</td>
<td>Running</td>
<td>CPU 0</td>
</tr>
<tr>
<td>47</td>
<td>Graphasaurus (419)</td>
<td>_dispatch_worker_thread3 0x8906</td>
<td>Running</td>
<td>CPU 1</td>
</tr>
<tr>
<td>47</td>
<td>Graphasaurus (419)</td>
<td>_dispatch_worker_thread3 0x8907</td>
<td>Runnable</td>
<td>n/a</td>
</tr>
</tbody>
</table>
System Load
System Load

Details Active Threads

<table>
<thead>
<tr>
<th>Priority</th>
<th>Process</th>
<th>Thread</th>
<th>State</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>mach_kernel</td>
<td>0x711f</td>
<td>Running</td>
<td>CPU 0</td>
</tr>
<tr>
<td>47</td>
<td>Graphasaurus (419)</td>
<td>_dispatch_worker_thread3 0x8906</td>
<td>Running</td>
<td>CPU 1</td>
</tr>
<tr>
<td>47</td>
<td>Graphasaurus (419)</td>
<td>_dispatch_worker_thread3 0x8907</td>
<td>Runnable</td>
<td>n/a</td>
</tr>
</tbody>
</table>
User Interactive Load Average

Average active threads over a 10 ms period
Priority $\geq 33$
User Interactive Class (QoS)
Orange when load exceeds hardware
Demo
Priorities

Joe Grzywacz
Quality of Service

Prioritizing your threads

<table>
<thead>
<tr>
<th>Process / Name</th>
<th>Count</th>
<th>Duration</th>
<th>Min Duration</th>
<th>Avg Duration</th>
<th>Std Dev Dur.</th>
<th>Max Durati...</th>
</tr>
</thead>
<tbody>
<tr>
<td>* All *</td>
<td>3,443</td>
<td>12.29 s</td>
<td>1.33 µs</td>
<td>3.57 ms</td>
<td>3.46 ms</td>
<td>18.83 ms</td>
</tr>
<tr>
<td>Grphasaurus (454)</td>
<td>3,443</td>
<td>12.29 s</td>
<td>1.33 µs</td>
<td>3.57 ms</td>
<td>3.46 ms</td>
<td>18.83 ms</td>
</tr>
<tr>
<td>GenTooltips</td>
<td>1,060</td>
<td>2.73 s</td>
<td>1.33 µs</td>
<td>2.57 ms</td>
<td>2.61 ms</td>
<td>18.83 ms</td>
</tr>
<tr>
<td>UpdateGraph</td>
<td>1,060</td>
<td>879.86 ms</td>
<td>461.71 µs</td>
<td>830.06 µs</td>
<td>401.43 µs</td>
<td>2.72 ms</td>
</tr>
<tr>
<td>RenderGraph</td>
<td>1,059</td>
<td>5.33 s</td>
<td>3.95 ms</td>
<td>5.03 ms</td>
<td>489.68 ms</td>
<td>8.00 ms</td>
</tr>
<tr>
<td>CADisplayLink</td>
<td>264</td>
<td>3.36 s</td>
<td>10.46 ms</td>
<td>12.73 ms</td>
<td>758.19 µs</td>
<td>14.61 ms</td>
</tr>
</tbody>
</table>
Quality of Service
Prioritizing your threads
# Quality of Service

Prioritizing your threads

<table>
<thead>
<tr>
<th>Priority</th>
<th>Process</th>
<th>Thread</th>
<th>State</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Graphasaurus</td>
<td>_dispatch_worker_thread3</td>
<td>Running</td>
<td>CPU 1</td>
</tr>
<tr>
<td>38</td>
<td>Graphasaurus</td>
<td>_dispatch_worker_thread3</td>
<td>Running</td>
<td>CPU 0</td>
</tr>
<tr>
<td>31</td>
<td>locationd</td>
<td>_dispatch_worker_thread3</td>
<td>Preempted</td>
<td>n/a</td>
</tr>
<tr>
<td>4</td>
<td>Graphasaurus</td>
<td>_dispatch_worker_thread3</td>
<td>Runnable</td>
<td>n/a</td>
</tr>
</tbody>
</table>
## Quality of Service

### Prioritizing your threads

<table>
<thead>
<tr>
<th>Priority</th>
<th>Process</th>
<th>Thread</th>
<th>State</th>
<th>Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Graphasaurus (454)</td>
<td>_dispatch_worker_thread3 0xe25d</td>
<td>Running</td>
<td>CPU 1</td>
</tr>
<tr>
<td>38</td>
<td>Graphasaurus (454)</td>
<td>_dispatch_worker_thread3 0xe251</td>
<td>Running</td>
<td>CPU 0</td>
</tr>
<tr>
<td>31</td>
<td>locationd (68)</td>
<td>_dispatch_worker_thread3 0xe039</td>
<td>Preempted</td>
<td>n/a</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>Graphasaurus (454)</td>
<td>_dispatch_worker_thread3 0xe24f</td>
<td>Runnable</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Quality of Service

Prioritizing your threads

Attribute of blocks, queues, threads
Constrains the priority range
Throttles I/O
Throttles CPU frequency
Virtual Memory

Faults

Affect performance
Worse under a load
Manageable
System Trace
Has the tools
## System Trace

**Has the tools**

<table>
<thead>
<tr>
<th>Duration</th>
<th>Self Duration</th>
<th>Symbol Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>178.12 ms 95.4% 0 s</td>
<td></td>
<td>Grphasaurus (419)</td>
</tr>
<tr>
<td>175.59 ms 94.1% 0 s</td>
<td></td>
<td>Copy On Write</td>
</tr>
<tr>
<td>2.45 ms 1.3% 0 s</td>
<td></td>
<td>Zero Fill</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>Page Cache Hit</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>Main Thread 0x88e8</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>start libdyld.dylib</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>main Grphasaurus</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>UIApplicationMain UIKit</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>-[UIApplication _run] UIKit</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>GSEventRunModal GraphicsServices</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>CFRunLoopRunSpecific CoreFoundation</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>_CFRunLoopRun CoreFoundation</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>_CFRunLoopDoSource1 CoreFoundation</td>
</tr>
<tr>
<td>78.96 µs 0.0% 0 s</td>
<td></td>
<td>_CFRunLoop_IS_CALLING_OUT_TO_A_SOURCE1_PERFORM_FUNCTION</td>
</tr>
<tr>
<td>72.96 µs 0.0% 0 s</td>
<td></td>
<td>migHelperRecievePortCallout AppSupport</td>
</tr>
<tr>
<td>72.96 µs 0.0% 0 s</td>
<td></td>
<td>_XReceivedStatusBarDataAndActions UIKit</td>
</tr>
<tr>
<td>72.96 µs 0.0% 0 s</td>
<td></td>
<td>-[UIScreenStatusBar statusBarServer:didReceiveStatusBarData:withAction: source: onThread::] UIKit</td>
</tr>
<tr>
<td>72.96 µs 0.0% 0 s</td>
<td></td>
<td>-[UIScreenStatusBar _foregroundView setStatusBarData:actions:] UIKit</td>
</tr>
<tr>
<td>72.96 µs 0.0% 0 s</td>
<td></td>
<td>-[UIScreenStatusBar _foregroundView_setStatusBarData:actions:] UIKit</td>
</tr>
<tr>
<td>72.96 µs 0.0% 0 s</td>
<td></td>
<td>-[UIScreenStatusBarLayoutManager updateItemsWithData:actions:] UIKit</td>
</tr>
</tbody>
</table>
| 72.96 µs 0.0% 0 s | | -[UIScreenStatusBarLayoutManager_updateItemViewViewWithD
Fault on Access

Allocations are quick
First access causes fault
Resolved Inline

No explicit call

Access any byte in the page

Just-in-time mapping to physical memory
Mitigation
Mitigation

Absorb them

• Leave room for faulting in your budget
• More resilient under a load
Mitigation

Absorb them
• Leave room for faulting in your budget
• More resilient under a load

Fault pages on a background thread
• dispatch_async
• Avoids stutters when showing new content
Summary

Companion to the Time Profiler

Applications that scale well under heavy loads

Try it out on your app

Many new features in Instruments 8
More Information

https://developer.apple.com/wwdc16/411
<table>
<thead>
<tr>
<th>Related Sessions</th>
<th>Location</th>
<th>Date Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizing App Startup Time</td>
<td>Mission</td>
<td>Wednesday 10:00AM</td>
</tr>
<tr>
<td>Using Time Profiler in Instruments</td>
<td>Nob Hill</td>
<td>Friday 3:00PM</td>
</tr>
<tr>
<td>Concurrent Programming with GCD in Swift 3</td>
<td>Pacific Heights</td>
<td>Friday 4:00PM</td>
</tr>
<tr>
<td>Labs</td>
<td>Location</td>
<td>Time</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>System Trace Q&amp;A Lab</td>
<td>Fort Mason</td>
<td>Thursday 10:00PM</td>
</tr>
<tr>
<td>Xcode Open Hours</td>
<td>Developer Tools Lab C</td>
<td>Thursday 12:00PM</td>
</tr>
<tr>
<td>Profiling and Debugging Lab</td>
<td>Developer Tools Lab C</td>
<td>Friday 3:00PM</td>
</tr>
<tr>
<td>Xcode Open Hours</td>
<td>Developer Tools Lab B</td>
<td>Friday 3:00PM</td>
</tr>
</tbody>
</table>