Measuring Performance Using Logging

Signposts and Instruments

Session 405

Shane Owara, Darwin Runtime
Chad Woolf, Instruments
Your mission: Improve performance
Introducing Signposts

Signposts
- Part of the os_log family
- Performance-focused time markers

Instruments
- Aggregate and analyze signpost data
- Visualize activity over time
let logHandle = OSLog(subsystem: "com.example.widget", category: "Setup")

os_log(.info, log: logHandle, "Hello, %{public}s!", world)

Our new logging system was introduced at WWDC 2016
• Built for debugging with efficiency and privacy in mind
let logHandle = OSLog(subsystem: "com.example.widget", category: "Setup")

os_log(.info, log: logHandle, "Hello, %{public}s!", world)

Our new logging system was introduced at WWDC 2016

• Built for debugging with efficiency and privacy in mind

Signposts created for investigating performance

• Built for performance use case and integration with developer tools

---

Unified Logging and Activity Tracing

WWDC 2016

Adopting signposts
Overlapping operations
Adding metadata
Controlling signposts
Investigating with Instruments
Measuring Intervals with Signposts
Fetch Asset X
Fetch Asset Y
Fetch Asset Z
Fetch Asset X

Fetch Asset Y

Fetch Asset Z

Time
Fetch Asset X

Fetch Asset Y

Fetch Asset Z

os_signpost(.begin, ...)

os_signpost(.end, ...)
for element in panel.elements {

    fetchAsset(for: element)
}

import os.signpost

for element in panel.elements {
    fetchAsset(for: element)
}

import os.signpost

let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

for element in panel.elements {
    fetchAsset(for: element)
}

Category: Use for grouping
import os.signpost

let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAsset(for: element)
    os_signpost(.end, log: refreshLog, name: "Fetch Asset")
}
Fetch Asset X

Fetch Asset Y

Fetch Asset Z
import os.signpost

let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAsset(for: element)
    os_signpost(.end, log: refreshLog, name: "Fetch Asset")
}
import os.signpost

let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

os_signpost(.begin, log: refreshLog, name: "Refresh Panel")
for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAsset(for: element)
    os_signpost(.end, log: refreshLog, name: "Fetch Asset")
}

os_signpost(.end, log: refreshLog, name: "Refresh Panel")

A different signpost name for this different interval
Fetch Asset X
Fetch Asset Y
Fetch Asset Z
Fetch Asset X
Fetch Asset Y
Fetch Asset Z
Measuring Asynchronous Intervals
Fetch Asset X
Fetch Asset Y
Fetch Asset Z
Signpost Names

```swift
os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
os_signpost(.end, log: refreshLog, name: "Fetch Asset")
```

The string literal identifies signpost intervals

The name must match at `.begin` and `.end`
Signpost IDs

```swift
let spid = OSSignpostID(log: refreshLog)
os_signpost(.begin, log: refreshLog, name: "Fetch Asset", signpostID: spid)
os_signpost(.end, log: refreshLog, name: "Fetch Asset", signpostID: spid)
```

Use signpost IDs to tell overlapping operations apart.

While running, use the same IDs for each pair of `.begin` and `.end`.
Making Signpost IDs

```swift
let spid = OSSignpostID(log: refreshLog)

let spid = OSSignpostID(log: refreshLog, object: element)
```

Signpost IDs are process-scoped

Making from object is convenient if you have the same object at `.begin` and `.end`
Asynchronous Operation X
Fetch Asset X

Asynchronous Operation Y
Fetch Asset Y

Asynchronous Operation Z
Fetch Asset Z
Asynchronous Operation X

Asynchronous Operation Y

Asynchronous Operation Z

Fetch Asset X

Fetch Asset Y

Fetch Asset Z

Signpost IDs:

- X
- Y
- Z

Time
let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

os_signpost(.begin, log: refreshLog, name: "Refresh Panel")

for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAsset(for: element)
    os_signpost(.end, log: refreshLog, name: "Fetch Asset")
}

os_signpost(.end, log: refreshLog, name: "Refresh Panel")
let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

os_signpost(.begin, log: refreshLog, name: "Refresh Panel")

for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAssetAsync(for: element) {
        os_signpost(.end, log: refreshLog, name: "Fetch Asset")
    }
}
notifyWhenDone {
    os_signpost(.end, log: refreshLog, name: "Refresh Panel")
}
let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

let spidForRefresh = OSSignpostID(log: refreshLog)

os_signpost(.begin, log: refreshLog, name: "Refresh Panel")

for element in panel.elements {
    let spid = OSSignpostID(log: refreshLog, object: element)
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAssetAsync(for: element) {
        os_signpost(.end, log: refreshLog, name: "Fetch Asset")
    }
}

notifyWhenDone {
    os_signpost(.end, log: refreshLog, name: "Refresh Panel")
}
let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

let spidForRefresh = OSSignpostID(log: refreshLog)

os_signpost(.begin, log: refreshLog, name: "Refresh Panel", signpostID: spidForRefresh)

for element in panel.elements {
    let spid = OSSignpostID(log: refreshLog, object: element)
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset", signpostID: spid)
    fetchAssetAsync(for: element) {
        os_signpost(.end, log: refreshLog, name: "Fetch Asset", signpostID: spid)
    }
}

notifyWhenDone {
    os_signpost(.end, log: refreshLog, name: "Refresh Panel", signpostID: spidForRefresh)
}
## Organizing Signposts: A Hierarchy

```swift
log = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

os_signpost(.begin, log: log, name: "Fetch Asset", signpostID: spid)
```

<table>
<thead>
<tr>
<th>Example</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log category</td>
<td>&quot;RefreshOperations&quot;</td>
</tr>
<tr>
<td>Signpost name</td>
<td>&quot;Fetch Asset&quot;</td>
</tr>
<tr>
<td>Signpost ID</td>
<td>spid</td>
</tr>
</tbody>
</table>

- Log category: Represents related operations.
- Signpost name: An operation to measure.
- Signpost ID: Single interval.
Adding Metadata to Signposts
Custom Metadata in Signpost Arguments

```swift
os_signpost(.begin, log: log, name: "Compute Physics")
```
Custom Metadata in Signpost Arguments

os_signpost(.begin, log: log, name: "Compute Physics", "for particle")

Add context to the .begin and .end
Custom Metadata in Signpost Arguments

```
os_signpost(.begin, log: log, name: "Compute Physics",
            "%d %d %d %d",
            x1, y1, x2, y2)
```

Add context to the `.begin` and `.end`

Pass arguments with `os_log` format string literal
Custom Metadata in Signpost Arguments

```python
os_signpost(.begin, log: log, name: "Compute Physics",
            "%.1f  %.1f  %.2f  %.1f  %.1f",
            x1, y1, m, x2, y2)
```

Add context to the `.begin` and `.end`

Pass arguments with `os_log` format string literal

Pass many arguments with different types
Custom Metadata in Signpost Arguments

```python
os_signpost(.begin, log: log, name: "Compute Physics",
            "%%{public}s  %.1f  %.1f  %.2f  %.1f  %.1f",
            description, x1, y1, m, x2, y2)
```

Add context to the `.begin` and `.end`

Pass arguments with `os_log` format string literal

Pass many arguments with different types

Pass dynamic strings
Custom Metadata in Signpost Arguments

```go
os_signpost(.begin, log: log, name: "Compute Physics",
    "for %{public}s at (%.1f, %.1f) with mass %.2f and velocity (%.1f, %.1f)",
    description, x1, y1, m, x2, y2)
```

Add context to the `.begin` and `.end`

Pass arguments with `os_log` format string literal

Pass many arguments with different types

Pass dynamic strings

The format string is a fixed cost, so feel free to be descriptive!
Adding Independent Events
Fetch Asset

os_signpost(.begin, ...)

Time

os_signpost(.end, ...)
Signpost Events

```
os_signpost(.event, log: log, name: "Fetch Asset",
            "Fetched first chunk, size %u", size)
os_signpost(.event, log: log, name: "Swipe",
            "For action 0x%x", actionCode)
```

Marking a single point in time
Event: Connected to service
Event: Fetched first chunk
Event: Swipe to update
Event: Swipe to update

Events: Swiping to update over and over and over!
Conditionally Enabling Signposts
Signposts Are Lightweight

Built to minimize observer effect

Built for fine-grained measurement in a short time span
Enabling and Disabling Signpost Categories

**OSLog.disabled**

Take advantage of special log handle

Just change the handle—can leave calling sites alone
let refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")

os_signpost(.begin, log: refreshLog, name: "Refresh Panel")
for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAsset(for: element)
    os_signpost(.end, log: refreshLog, name: "Fetch Asset")
}
os_signpost(.end, log: refreshLog, name: "Refresh Panel")
let refreshLog: OSLog
if ProcessInfo.processInfo.environment.keys.contains("SIGNPOSTS_FOR_REFRESH") {
    refreshLog = OSLog(subsystem: "com.example.your-app", category: "RefreshOperations")
} else {
    refreshLog = .disabled
}

os_signpost(.begin, log: refreshLog, name: "Refresh Panel")
for element in panel.elements {
    os_signpost(.begin, log: refreshLog, name: "Fetch Asset")
    fetchAsset(for: element)
    os_signpost(.end, log: refreshLog, name: "Fetch Asset")
}

os_signpost(.end, log: refreshLog, name: "Refresh Panel")
Instrumentation-Specific Code

```swift
if refreshLog.signpostsEnabled {
    let information = copyDescription()
    os_signpost(..., information)
}
```

For additional expensive code that is only useful for the signpost
Signposts in C
# Signposts in C

<table>
<thead>
<tr>
<th>Swift</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>import os.signpost</code></td>
<td><code>#include &lt;os/signpost.h&gt;</code></td>
</tr>
<tr>
<td>OLog</td>
<td><code>os_log_t, os_log_create()</code></td>
</tr>
<tr>
<td><code>.disabled</code></td>
<td><code>OS_LOG_DISABLED</code></td>
</tr>
<tr>
<td><code>os_signpost(.begin, ...)</code></td>
<td><code>os_signpost_interval_begin()</code></td>
</tr>
<tr>
<td><code>os_signpost(.end, ...)</code></td>
<td><code>os_signpost_interval_end()</code></td>
</tr>
<tr>
<td><code>os_signpost(.event, ...)</code></td>
<td><code>os_signpost_event_emit()</code></td>
</tr>
<tr>
<td>OSSignpostID</td>
<td><code>os_signpost_id_t</code></td>
</tr>
</tbody>
</table>
Instruments 10
Instruments 10

os_signpost
Instruments 10

os_signpost

Points of Interest
Instruments 10

os_signpost

Points of Interest

Custom instruments
Demo
Visualizing signpost data
Summary

Annotate code with signposts
• Easily mark intervals
• Capture metadata of interest

Use Instruments to view signpost data
• Visualize where time is spent
• Understand what program is doing
<table>
<thead>
<tr>
<th>Event Title</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating Custom Instruments Lab</td>
<td>Technology Lab 8</td>
<td>Wednesday 3:00PM</td>
</tr>
<tr>
<td>Creating Custom Instruments</td>
<td>Hall 1</td>
<td>Thursday 11:00AM</td>
</tr>
</tbody>
</table>

More Information

https://developer.apple.com/wwdc18/405