Launchd

At Your Service!

Jonathan Levin, j (-at-) Technologeeks.com

The Presenter

- Author:
  - "Mac OS X and iOS Internals" (Wiley, 2012)
    - http://newosxbook.com/
    - 2nd Edition (Updates to 10.10/iOS 8) Due March 2015
      - Taking reader requests: http://newosxbook.com/forum/index.php
  - "Android Internals: A Confectioner's Cookbook"
    - First book to cover Android Internals: http://newandroidbook.com

- Trainer and Consultant
  - CTO of Technologeeks.com (@Technologeeks)

The Presentation

- Launchd – functional overview
  - Rehash of well documented stuff (RTFM, launchd.info, Wiki)
  - The much more useful, albeit not-so-documented features

- Presentation: http://technologeeks.com/docs/launchd.pdf

- Bonus material: Chapter 7 of MOXiI (1st edition)
  - http://newosxbook.com/articles/Ch07.pdf
    - Also discusses iOS SpringBoard and OS X Finder

Launchd Roles

- Launchd is the very first process to startup
  - PID 1, started directly by the kernel
- Will refuse to be started manually
  - Can start per-user instances of itself
  - One instance per logged on user (pre 10.10)
    - Including system services such as _spotlight
    - Post 10.10 – only one instance (uses XPC)
- Mission Statement:
  - Launch (start) jobs (processes) by/with specified criteria

Launchd Roles

init

- Launchd takes over the traditional role of UN*X init

TABLE 7-2: init vs. launchd

RESPONSIBILITY	TRADITIONAL INIT	LAUNCHD
Function as PID 1, great ancestor of all processes	init is the first process to emerge into user mode, and forks other processes (which in turn may fork others). Resource limits it sets for itself are inherited by all of its descendants.	Same. launchd also sets Mach exception ports, which are used by the kernel internally to handle exception conditions and generate signals (see Chapter 8).
Support "run levels"	Traditional init supports run levels: 0 – poweroff 1 – single user 2 – multi-user 3 – multi-user + NFS 5 – halt 6 – reboot	launchd does not recognize run levels and allows only for individual per-daemon or per-agent files. There is, however, a distinction for single-user mode.
Start system services	init runs services in order, per files listed in `/etc/rc?.d` (corresponding to run level), in lexicographic order.	launchd runs both system services (daemons), and per-user services (agents).
System service specification	init runs services as shell scripts, unaware and oblivious to their contents.	launchd processes property list files, with specific keywords.
Restart services on exit	init recognizes the `respawn` keyword in `/etc/inittab` for restart.	launchd allows a KeepAlive key in the daemon or agent's property list.
Default user	Root.	Root, but launchd allows a username key in the property list.

Source: Mac OS X/iOS Internals (1st), pg. 230

Daemons and Agents

- Daemons:
  - Background services
  - No UI (stdin/stdout/stderr redirected to /dev/null or file)
  - Not dependent on user logon

- Agents*
  - Run in the context of a user session
  - May present a User Interface

* No Agents in iOS

Startup Paths

TABLE 7-1: Launch Daemon locations

DIRECTORY	USED FOR
/System/Library/LaunchDaemons	Daemon plist files, primarily those belonging to the system itself.
/Library/LaunchDaemons	Daemon plist files, primarily third party.
/System/Library/LaunchAgents	Agent plist files, primarily those belonging to the system itself.
/Library/LaunchAgents	Other agent plist files, primarily third party. Usually empty.
~/Library/LaunchAgents	User-specific launch agents, executed for this user only.

Source: Mac OS X/iOS Internals (1st), pg. 229

- Tip: Monitor these directories frequently
  - Malware seeking persistence will likely leave traces there

Segue: Property Lists

- Apple favors this weird XML syntax. God knows why*.
- Terrible to read, easy to serialize (to a CFDictionary)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
...
</dict>
</plist>
```

- No attributes. <key> elements, values follow.

* - Actually, the people of NeXT do. This is yet another legacy of NeXTSTEP 3.3..

Segue: Property Lists

- Some property lists are compacted to the BPList form
  - Especially in iOS, to save on precious XML-parser memory
  - “file” recognizes these as “Apple binary property list”

- Use the plutil utility for quick conversions:

```
$ cat shell.plist | plutil -convert xml1 - -o -
```

- Also useful to test if your plist is valid:

```
$ plutil some.test.plist
OK
```

Launchd plist format

- Required keys are straightforward:

Key	Type	
Label	String	Unique Job identifier
Program or ProgramArguments	String	Path to executable
	Array of strings	argv[0]...argv[argc-1]

... But the optional keys unlock the true functionality.

Launchd Roles

Resource Throttling

- Before forking the job, launchd can `setrlimit(2)`:
  - Define limits as array under Soft/HardResourceLimits

Limit key	ulimit(1) equivalent	
Core	-c	Maximum Core size
CPU	-t	CPU time (in seconds)
Data	-d	Size (in bytes) of Data segment
FileSize	-f	
MemoryLock	-l	Size (in bytes) of Maximum mlock(2)
NumberOfFiles	-n	# of open files
NumberOfProcesses	-u	# of processes forked
ResidentSetSize	-m	Size (in bytes) of max RAM usage
Stack	-s	Size (in bytes) of stack segment

Launchd Roles

Resource Throttling

- Additional throttling keys include:

Limit key	Purpose
Nice	Call nice(3) to adjust process priority
LowPriorityIO	Throttle I/O of this job
~~TimeOut~~	~~Idle timeout, in seconds~~
ExitTimeOut	Interval (in seconds) between kill -TERM and -KILL
ThrottleInterval	Spawn job not more than once every n seconds
EnablePressuredExit	(10.10/iOS 8): Exit if memorystatus detects pressure

10.10

- Undocumented (in iOS): JetsamProperties

Limit key	Restricts/Sets
JetsamMemoryLimit	Memory limit, in MB
JetsamPriority	Priority band. Implies app killability on low memory

Launchd Roles

Controlling launched jobs

- In addition to the command line, launchd controls:

Key	Value	Purpose
EnvironmentVariables	Dictionary (strings)	Pass environment to job
StandardInPath StandardOutPath StandardErrorPath	String	Redirects stdin/stdout/stderr
Umask	Integer	Default umask(3) of process
RootDirectory	String	Chroot(2) to here
WorkingDirectory	String	Chdir(2) to here
User	String	Username to run job as
DisableASLR	Boolean	Address Space Randomization
WaitForDebugger	Boolean	Waits for gdb/lldb attachment
LegacyTimers	Boolean	[Non]-Coalesce job timers **10.9**

Launchd Roles

Resource Throttling

- OS X 10.9+ and iOS support setting ProcessType:

ProcessType value	Purpose
Standard	Default Settings
Background	Background process , low priority (~4), low limits
Interactive	UI/interactive process , preferred priority (~47)
Adaptive	Dynamic priority and limit adjustments

- Parameter passed to posix_spawnattr_setprocesstype_np
- Deprecates (the undocumented) POSIXSpawnType

Launchd Roles

Immortality (or, at least, reincarnation)

- Launchd can keep your job alive (KeepAlive)
  - Use <true/> to keep alive always, or specify dictionary:

KeepAlive Condition	Expects...
SuccessfulExit	Boolean value – restart on [un]Successful exit ($? = 0)
~~NetworkState~~	~~Boolean – keep job alive if network is [un]available~~ **10.10**
PathState	Dictionary of filesystem paths and boolean values
OtherJobEnabled OtherJobActive	Dictionary of other job labels and boolean values
Crashed	Boolean – restart if crashed

- Deprecates older OnDemand key

Launchd Roles

atd/crond

- Supports periodic/scheduled execution:

Key	
RunAtLoad	Run job immediately when plist file is loaded
StartInterval	Interval (min 10 seconds) to run job
StartCalendarInterval	Specify WeekDay/Hour/Minute (emulates UN*X crontab)
LaunchOnlyOnce	Run job once, and never try again

Launchd Roles

inetd

- Incorporates classic inetd/xinetd functionality:
  - <sockets> expects dictionary[2] or dictionary or arrays
  - By convention, use <listeners> dictionary

Sockets dictionary key	Values
SockType	Stream (TCP) or Dgram (TCP)
SockProtocol	TCP or UDP (redundant if using SockType)
SockFamily	IPv4 or IPv6
SockServiceName	Entry in /etc/services to resolve port
SockNodeName	IP Address to bind to (default: 0.0.0.0 – INADDR_ANY)
SockPassive	True if listen(2)ing, false if connect(2)ing

  - Bonus: Register with Bonjour, and/or join MulticastGroup

Launchd Roles

can your inetd do this?

- Sockets also extended to UN*X domain
  - Same key <sockets> , same values, but:

Sockets dictionary key	Values
SockType	Stream , Dgram , or SeqPacket
SockPathName	Path to socket representation on filesystem
SockFamily	Unix
SockPathOwner	Ownership of socket respresentation on filesystem
SockPathMode	Permissions **in decimal** (e.g. 511 = 0777)
SecureSocketWithKey	Random, secure name for socket, inherited via env. var

LimitLoad[To/From]..

- SessionType: May restrict daemon or agent to/from
  - Aqua: GUI login
  - LoginWindow: Pre-Login agent
  - Background: background daemon, no UI
  - System: root launchd context only

10.9+:

- Hardware: Restrict daemon/agent to/from machine
  - Specify machines in “machine” dict
  - As per Apple nomenclature (e.g. MacBook5,1)

Triggers

- The documented trigger keys include:

Key	Purpose
WatchPaths	Array of filenames to watch, and start job on
QueueDirectories	Array of dirnames to watch, and start job on
StartOnMount	Boolean – Start on any filesystem mount (autorun!)

Triggers

- The **un**documented* LaunchEvents is FAR more potent:
  - VFS Notifications (e.g. low disk)
  - Network state notifications
  - I/O kit notifications (device matching)
  - Generic notifications

.. And therein lies the true power of launchd-based automation

(q.v. LibNotify's notify_keys.h for a partial references)

* - Apple finally acknowledges LaunchEvents in the 10.10 launchd.plist man, but mostly as an obiter

VFS Notifications

com.apple.cache_delete.plist

```
        <key>LaunchEvents</key>
        <dict>
                <key>com.apple.dispatch.vfs</key>
                <dict>
                        <key>Monitor Low Disk Conditions</key>
                        <dict>
                                <key>LowDisk</key>
                                <integer>30</integer>
                                <key>VeryLowDisk</key>
                                <integer>0</integer>
                        </dict>
                </dict>
...
```

Network State Notifications

com.apple.networkd.plist

```
<key>LaunchEvents</key>
  <dict>
        <key>com.apple.notifyd.matching</key>
        <dict>
           <key>com.apple.airport.userNotification</key>
           <dict>
                 <key>Notification</key>
                 <string>com.apple.airport.userNotification</string>
           </dict>
        </dict>
    </dict>
```

Network State Notifications

Location or network change

com.apple.icloud.findmydeviced.plist

```
<key>LaunchEvents</key>
        <dict>
                <key>com.apple.notifyd.matching</key>
                <dict>
                        <key>com.apple.locationd/Prefs</key>
                        <dict>
                                <key>Notification</key>

        <string>com.apple.locationd/Prefs</string>
                        </dict>
                        <key>com.apple.system.hostname</key>
                        <dict>
                                <key>Notification</key>

        <string>com.apple.system.hostname</string>
                        </dict>
                </dict>
        </dict>
```

I/O Kit Notifications

Respond to any USB device

com.apple.usbd.plist

```
<key>LaunchEvents</key>
    <dict>
        <key>com.apple.iokit.matching</key>
        <dict>
            <key>com.apple.usb.device</key>
            <dict>
                <key>IOProviderClass</key>
                <string>IOUSBDevice</string>
                <key>idProduct</key>
                <string>*</string>
                <key>idVendor</key>
                <string>*</string>
                <key>IOMatchLaunchStream</key>
                <true/>
            </dict>
        </dict>
    </dict>
```

I/O Kit Notifications

Respond to BlueTooth devices

com.apple.blued.plist

```
<key>LaunchEvents</key>
    <dict>
        <key>com.apple.iokit.matching</key>
        <dict>
            <key>com.apple.bluetooth.hostController</key>
            <dict>
                <key>IOProviderClass</key>
                <string>IOBluetoothHCIController</string>
                <key>IOMatchLaunchStream</key>
                <true/>
            </dict>
        </dict>
    </dict>
```

Also check: com.apple.bluetoothaudiod.plist

Generic Notifications

com.apple.softwareupdated.plist

```
<key>LaunchEvents</key>
    <dict>
        <key>com.apple.notifyd.matching</key>
        <dict>
            <key>ManualBackgroundTrigger</key>
            <dict>
                <key>Notification</key>
          <string>com.apple.SoftwareUpdate.TriggerBackgroundCheck</string>
            </dict>
            <key>CheckForCatalogChange</key>
            <dict>
                <key>Notification</key>
           <string>com.apple.SoftwareUpdate.CheckForCatalogChange</string>
            </dict>
        </dict>
...
```

Launchd Roles

mach_init

- Supports Mach bootstrap server functionality
- Registers Mach services:
  - well known (i.e. special) Mach ports
    - HostSpecialPort directive
  - Ephemeral (arbitrary) service ports
    - MachServices directive
- Outside the scope of this talk, but SUPER powerful
  - Esp. HostSpecialPort (override amfid, sandboxd, etc)
  - ExceptionServer (e.g. CrashReporter)

Launchd Roles

XPC domain enforcement

- As of 10.7: Launchd serves as XPC focal point
- Idea: Limits visibility of Mach services to subdomains
  - If you can’t see the service, you can’t use/abuse it
- Full discussion of XPC is also outside our scope..

Fun with launchd

- Control the beast with launchctl
  - Command line, with optional shell (pre 10.10)
  - Run as root to access global context
  - Otherwise you can only see your own context

  - Useful: load/unload, start/stop

Fun with launchd (10.10)

- Launchctl in 10.10/iOS 8 no longer interactive
- Old commands (somewhat) supported (~~bstree,bslist~~)
- New commands
  - procinfo prints detailed information on ***any*** pid
    - Entitlements
    - Environment
    - *some* Mach Ports (finally!)
  - hostinfo prints host special ports

The Bad news...

- Launchd is an ongoing, highly evolving project
  - 10.8: 442
  - 10.9: 842
  - 10.10: moved to libxpc 559 (560 in iOS 8)
    - Source not available yet – and may not ever be
    - Libxpc is a closed source project
- Open source days may be nearing their end
  - Also true for XNU – Apple refactoring more into kec KEXTs

- Launchd & XPC to be fully “out”-ed in 2nd Ed of MOXiL