

Acknowledgments

Contents at a Glance

0. About this book
 1. Darwinism: The Evolution of MacOS and the *OS variants
 2. E Pluribus Unum: The Darwin Architecture
 3. Promenade: A tour of the *OS Filesystems
 4. Experience Points: UX and System Services
 5. Automatic for the People: Application Services
 6. Ex Machina: The Mach-O File Format
 7. In the Darkness, Bind Them: dyld internals
 8. Parts of the Process: Processes, Threads, and the Grand Central Dispatcher
 9. In Memoriam: Process and System Memory Management
 10. CFRUN - RunLoopRun: The Runtime Environments
 11. The Message is the Medium: Mach IPC
 12. Mecum Porto: Mach Primitives
 13. The Alpha & Omega - launchd
 14. X is not a Procedure Call: XPC Internals
 15. Follow Me: Process Tracing & Debugging
 16. Het-Work: Darwin Networking
- A. Spitting Image: MacOS & *OS Software Images
- ## Appendices

Table of Contents

○ XNU (at a glance)	55
■ Mach & BSD		• About This Book
■ libkern		• Darwin: The Evolution of *OS
■ IOKit		• The Origin of Speed
■ The Platform Expert and ml APIs	• Version Attributed
■ Kernel extensions		• Darwin and Open Source
■ Darwin Technologies (at a glance)		• Darwin, UNIX and POSIX
3. Promenade: A tour of the *OS Filesystems.....	• Filesystem: NextStep
○ Partitioning		• Mac OS
○ Filesystems		• 10.0 through 10.15 (beta)
○ Disk Images		• 10.1 and later
○ Disk Arbitration		• 10.2 through 10.15 (beta)
○ ACLs & Attributes	• 10.2 and later
■ File flags		• 10.2 through 10.15 (beta)
■ Extended Attributes		• 10.2 and later
■ Transparent file compression		• 10.2 and later
■ File forks (MacOS)		• 10.2 and later
○ Directories		• Journaled
4. Experience Points: UX and System Services.....	• Journaled
○ FSEvents		• Quota
○ Spotlight		• The I-Devices
○ QuickLook		• Device Driver
○ System Information	• Device Driver
■ Darwin: sysctl		• Device Driver
■ MacOS: AppleSystemInfo.framework & system_profiler		• Device Driver
■ *OS: MobileGestalt		• Device Driver
○ System Configuration		• Device Driver
○ Duet		• Device Driver
○ Printing		• Device Driver
○ Siri (assistantd) & Voice Control		• Device Driver
○ The User Interface		• Device Driver
5. Automatic for the People: Application Services.....	• Device Driver
○ Application Installation & LaunchServices		• Device Driver
○ Preferences		• Device Driver
○ Notifications	• Device Driver
■ Darwin Notifications (/usr/bin/notifyd)		• Device Driver
■ Distributed Notification Center (/usr/sbin/distributed)		• Device Driver
■ *OS: Notification proxying		• Device Driver
■ Apple Push Notifications (APN)		• Device Driver

o User Notifications	oc_once)	o PTrees
o emond (MacOS)		o Non-POSIX extensions
o Apple Events (MacOS)	▪ AppleScript	▪ Initialization
	▪ RAM	▪ Thread lifecycle
o Under the Hood	▪ Sending Events	▪ Threadspecification
6. Ex Machina: The Mach-O File format.....		▪ What's in a Process
o Overture: Fat Binaries		▪ Thread Specific Data (TSD)
o General Mach-O Concepts		▪ Signatures
o Mach-O FileType		▪ Thread Local Variables (TLV)
o Header Flags		▪ Thread Objects and OS flags
o Load Commands	▪ Load commands known to XNU	▪ Process task & thread policies
	▪ Load commands known to dyld	▪ Task Central Dispatcher
	▪ Load commands serving little or no purpose	▪ Block objects
o Objects	▪ Deprecated load commands	▪ Queue objects
7. In the Darkness, Bind Them: dyld internals.....		▪ Generic attributes
o Program startup		
o Linking		
o dyld opcodes	▪ Binding opcodes	
	▪ Advantages of the opcode scheme	
	▪ Rebasing opcodes	
	▪ arm64e and dyld-625 (Darwin 18) changes	
o Exports		
o CoreSymbolication.framework		▪ On the same base
o Interposing		▪ Memory Management APIs
o The Shared Library Cache		▪ POSIX BSD APIs
o Darwin 17: dyldv3		▪ mach_mmap
o Programmatic manipulation of Mach-O objects		▪ memory pages
o Remote inspection of dyld state		▪ Implications
o Debugging & tracing dyld operations		▪ The scissipole (deletant) zone
8. Parts of the Process: Threads and the Grand Central Dispatcher.....		▪ Custom Zones
o Processes		▪ In the Zone
o Metrics	▪ Process lifecycle	▪ Magazines
o Scenarios	▪ Process identifiers and grouping	▪ Hosted improvements
o Metrics	▪ Coalitions (Darwin 14+)	▪ The NSudo zone
o Metrics	▪ Credentials	▪ The NSudos zone (Darwin 18+)
o Security	▪ Personae (*OS)	▪ The Purgeable zone and Lifecycle
o The Mach-O Interface Generator	▪ Signals	▪ Debugging

o PThreads (glance)	<ul style="list-style-type: none"> ▪ Non-POSIX extensions ▪ Initialization ▪ Thread lifecycle ▪ Introspection ▪ What's in a <code>pthread_t</code> ▪ Thread Specific Data (TSD) 	User Notifications
3. Promenade: A tour of the OS filesystems	<ul style="list-style-type: none"> o Partitioning ▪ Thread Local Variables (TLV) o Filesystems ▪ Thread priorities and QoS class o Disk Images ▪ Process, task & thread policies o Interlude: KEvents and KQueues o Grand Central Dispatcher <ul style="list-style-type: none"> ▪ Blocks ▪ Queues <ul style="list-style-type: none"> ▪ Attributes ▪ Queue attributes ▪ Queue maintenance ▪ Dispatching blocks o Directories <ul style="list-style-type: none"> ▪ Continuations 	Features: Fast Binary Events General Mach-O Concepts Mach-O Files Header Flags Load Commands Load Commands Known to XNU Load Commands Known to dyld Load Commands Known to the kernel Description load commands In the Dispatcher Build Thread: dyld interfaces Blocksize switch Tracing Glib objects Binding objects Advances of the object scheme Response objects Swapfile and dyld-625 (Darwin 18) changes
4. Experience Points: UX and System Services	<ul style="list-style-type: none"> o FSEvents ▪ Dispatch Objects o Spotlight ▪ Dispatch Sources o QuickLook ▪ Introspection o System Information ▪ kdebug codes 	Core Application Framework Interposing the system framework The Shared Library Cache Darwin 17: dyld Protectionistic manipulation of Mach-O objects Remote inspection of dyld state Dependencies & shared dyld operations Deployment & scaling dyld operations Darwin 18: dyld
9. In Memoriam: Process Memory Management	<ul style="list-style-type: none"> o On the same page o Memory Management APIs <ul style="list-style-type: none"> ▪ POSIX/BSD APIs o System Configuration ▪ mach <code>vm_map</code> o Printing ▪ memory tags o libmalloc <ul style="list-style-type: none"> ▪ The scalable (default) zone o The User Interface <ul style="list-style-type: none"> ▪ Custom Zones 	SystemInfo.framework & system_profiler Core Application Framework Interposing the system framework The Shared Library Cache Darwin 17: dyld Protectionistic manipulation of Mach-O objects Remote inspection of dyld state Dependencies & shared dyld operations Darwin 18: dyld
5. Automation for the People: Application Services	<ul style="list-style-type: none"> ▪ Zone APIs o Application Installation & LaunchServices ▪ In the Zone o Preferences <ul style="list-style-type: none"> ▪ Magazines o Notifications <ul style="list-style-type: none"> ▪ Hoard improvements ▪ Darwin Notifications (/usr/bin/notify) (Darwin 14+) ▪ Distributed Notification Center (/usr/sbin/distributednotifications) (Darwin 14+) ▪ *OS: Notification proxying ▪ The Nano zone ▪ The Nanov2 zone (Darwin 18+) ▪ The Purgeable zone and libcache ▪ Debugging 	Paths of the Process: Threads and the Global Context Processes Process Lifecycle Process Identifiers and Grouping Configuration (Darwin 14+) Dependencies (Darwin 14+) Persistence (*OS) Signals Apple Push Notifications (APN)

o libplatform (os_alloc_once)	12. Memory Protection: Mach Privileges
o Swap	<ul style="list-style-type: none"> ▪ Dynamic paging (MacOS) ▪ Compressed RAM
o Under pressure:	<ul style="list-style-type: none"> ▪ MacOS: MemoryStatus ▪ *OS: Jetsam ▪ Programmatic API ▪ Detecting and responding to memory pressure ▪ *OS: mmaintained
10. CFRun - RunLoopRun: The Runtime Environments.....	307
o CoreFoundation	<ul style="list-style-type: none"> ▪ CF* Objects ▪ CFRUNLoop Internals
o Objective-C	<ul style="list-style-type: none"> ▪ Objective-C library initialization ▪ Runtime support information ▪ Type encoding ▪ Classes and instances ▪ Sending messages ▪ Message passing internals ▪ Other compiler directives ▪ Reflection ▪ Method swizzling ▪ Tracing Objective-C
o Swift	<ul style="list-style-type: none"> ▪ Compiling ▪ Mach-O sections ▪ Reverse engineering Swift binaries ▪ Closures ▪ Mangling
11. The Message is the Medium: Mach IPC (the user mode view)	337
o A little history	<ul style="list-style-type: none"> ▪ History of the BSD Socket APIs
o Mach-isms	<ul style="list-style-type: none"> ▪ Configuration
o Scenarios	<ul style="list-style-type: none"> ▪ Statistics
o Mach Port APIs	<ul style="list-style-type: none"> ▪ Overview
o Message format	<ul style="list-style-type: none"> ▪ and PF
o Sending and receiving messages	
o The Mach Interface Generator	<ul style="list-style-type: none"> ▪ (NECP)
o SkyWalk	

12. Mecum Porto: Mach Primitives.....	367
◦ Ports as object abstractions	
◦ The host initialization	
◦ The host_privfecycle	
◦ The task inspection	
◦ The thread : in a pthread_t	
◦ The lesser objects [ic Data (TSD)]	
▪ The clock synchronization objects	
▪ The processor and processor set	
◦ Exception ports	
13. The Alpha & Omega - Launchd	389
◦ Born Again: Launchd's reincarnation in libxpc	
◦ The many faces of launchd	
▪ atd (scheduled execution)	
▪ crond (scheduled recurring execution)	
▪ inetd/xinetd (socket handoff)	
▪ Triggered execution	
◦ Interlude: UserEventAgent	
◦ Agents & Daemons	
▪ The <u>TEXT.</u> <u>_bs.plist</u> property list	
◦ Domains	
◦ launchctl	
◦ APIs kdebug codes	
▪ The XPC bootstrap pipe	
9. In Memoriam: Process Memory Management.....	275
◦ On the page	
◦ ServiceManagement.framework	
14. X is not a Procedure Call: XPC internals	411
◦ Design Rationale	
◦ Implementation details	
◦ XPC Object APIs	
▪ XPC data types (fault) zone	
▪ Object representation	
◦ XPC Transport APIs	
▪ XPC pipes	
▪ NSXPC*magazines	
▪ GCD Integration	
◦ Remote XPC	
▪ The Nano2 zone (Darwin 18+)	
▪ The Purgeable zone and libcache	
▪ Debugging	

- o XPC services
 - `xpc_main`
 - Transactions
 - Starting services
 - XPC caches

o XPC activities

- o Tracing XPC

15. Follow Me: Process Tracing and Debuggingwork on "Mac OS X Internals" has been 439

- o `proc_info`, Mac OS has dramatically morphed from a fringe operating system to one that is widely gaining acceptance. Owing much to the revolution that was the iPhone, more people than ever became "Mac people", leaving the PC behind or confining it to a VM. There is still no substitute for the hegemony of Windows - optimistic market share estimates have only one in three years being a Mac. And yet, no longer can developers afford to think only about Windows as a viable alternative, and Mac OS a force to be reckoned with.
- o `os_log(3)`
 - `logd`

I "grew up" on this book, and though it boasts uncanny resistance to the tests of time, it has begun fading to obsolescence, with a second edition nowhere in sight. Aside from the "Mac Hacks" reference - certainly not any of Apple's - has even begun to scratch the surface of the operating systems. Thus, six years ago, I decided to pick up the gauntlet and author a book myself, and a year later the first edition of this book - "Mac OS and iOS Internals" (known as MOXII) - was published.

o DTrace (Mac OS)

- o `kdebug`
 - Controlling `kdebug`
 - producing `kdebug` messages from user mode
- o `ktrace`

Not so any more. The second edition doesn't demur from tackling all of these head on, including providing examples of using the private framework APIs directly. And the most important change is that instead of one book - there are now three. This is Volume I, dealing strictly with user mode perspective of the system. Volume II - dedicated to the kernel perspective and many topics such as the boot process, hardware and networking - is planned next. Somewhat surprisingly, Volume III - dealing solely with security and the insecurity of the OSes - has been left out for a while. It had originally started out as this book's sixteenth chapter - before I had realized (100 pages in) that it merits its own book.

o Hangs and Crashes

- `spindump(8)`
- Core dumps
- Crash reporting

16. Het-Work: Darwin Networkingbut chose instead to start "tabula rasa" and rewrite 477

- o Darwin Extensions of the BSD Socket APIs
- o Networking Configuration

o Networking Statistics

I tried to cover as many bases as I could - which often felt like a Sisyphean task as, by the time I thought I understood an API, I found some other dependency or hidden feature.

Documenting the entire API from scratch is akin to exploring an uncharted planet, with new terrain and features at every turn.

o Firewalling (ALF and PF)

o Packet Filtering (BPF)

o Network Extension Control Policies (NECP)

o SkyWalk