Launch-on-Demand

Adapting to a dynamic world

Damien Sorresso
Daemon Spawn

Launch-on-Demand

Introduction

- Mac OS X's process lifecycle model
- Managed by launchd
- Ad-hoc bootstrapping model

Launch-on-Demand

What you'll learn

- Why Launch-on-Demand is better
- How to write a Launch-on-Demand job
- Leveraging GCD for asynchronous design
- How to write a privileged helper

In the Before Time…

- Startup order was well-defined
- Process ordained by Holy Priesthood of rc
- Rigid and inflexible
- Unsuited to modern, hot-plugging world

What to Unlearn

- Race-prone PID files
- “Make me a daemon“ code
- Makefile approach to bootstrapping
- Services correlating to processes

Learn to Let Go!

Launch-on-Demand

Once you've let go

Plug-in model

Pay-as-you-go computing

Boilerplate code

Holy Priesthood

Launch-on-Demand

Basic concepts

- Two job types
  - Daemons (system-wide)
  - Agents (per-user)
- Daemons run in privileged environment
- Agents run in user environment
- Can both advertise services

Launch-on-Demand

Basic concepts

- Services as virtual endpoints
- Not necessarily backed by a process
- No explicit dependency graph
- Dependencies expressed through IPC

Launch-on-Demand

Advantages

Efficiency

Reliability

Flexibility

Grand Central Dispatch

Crash course

- New synchronization primitive: queues
- Work submitted to queues
- Runtime guarantees synchronous execution of work
- Incredibly easy
- Incredibly fast

Grand Central Dispatch

Crash course

- Queues act as run loop primitive
- Sources schedule event handlers on queue
- Sources monitor for events
  - File descriptor readability/writability
  - Process events (exit, fork(2), exec(3), etc.)
  - Signals (SIGTERM, SIGHUP, etc.)
  - Timers

Blocks

Crash course

- Inline/anonymous functions
- Implicitly capture any stack state referenced
- No more marshaling state into a struct
- Acts as an archive for function call

Blocks

Crash course

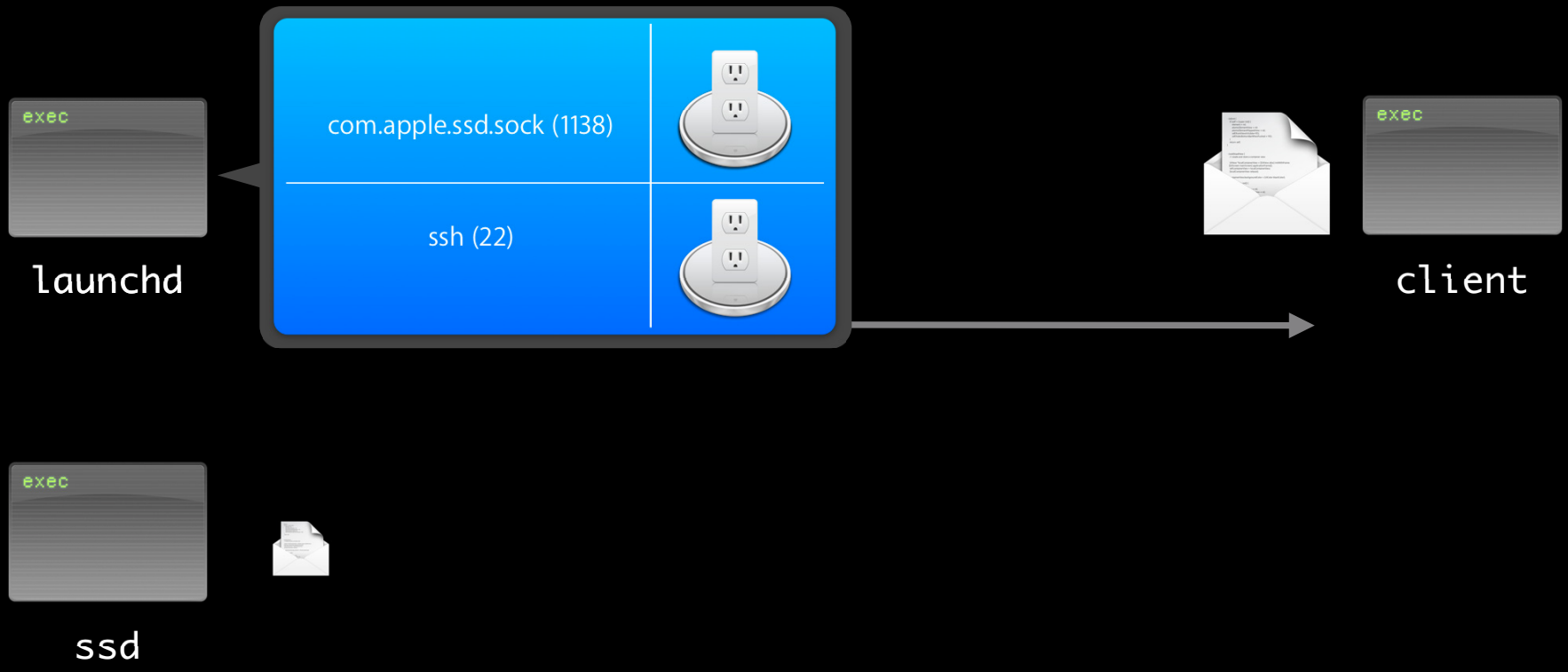
```
#include <dispatch/dispatch.h>

int
foo(dispatch_queue_t q)
{
    int a = 0; /* Captured as const. */
    int b = 1; /* Captured as const. */
    __block int c = 2; /* Mutable. */

    /* Copies block to heap. */
    dispatch_async(q, ^{
        c = 1; /* Groovy. */
        a = 1; /* Harshes the compiler's buzz. */
    });
}
```

A Simple Socket Server—ssd

Design goals

- As close to stateless as possible
- If needed, archive state
- Crashes will happen
- Asynchronous, HTTP-style request handling

A Simple Socket Server



A Simple Socket Server

launchd.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/
DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.apple.ssd</string>
    <key>Program</key>
    <string>/usr/libexec/ssd</string>
    <key>Sockets</key>
    <dict>
        <key>com.apple.ssd.sock</key>
        <dict>
            <key>SockServiceName</key>
            <string>1138</string>
        </dict>
    </dict>
</dict>
</plist>
```

A Simple Socket Server

Checking in

```
#include <launch.h>

launch_data_t req = launch_data_new_string(LAUNCH_KEY_CHECKIN);
launch_data_t resp = launch_msg(req);

launch_data_t sockets = launch_data_dict_lookup(resp, LAUNCH_JOBKEY_SOCKETS);
launch_data_t sarr = launch_data_dict_lookup(sockets, "com.apple.ssd.sock");

launch_data_t sockv6 = launch_data_array_get_index(sarr, 0); /* IPv6 */
launch_data_t sockv4 = launch_data_array_get_index(sarr, 1); /* IPv4 */
int sockfd = launch_data_get_fd(sockv4);
```

Don't Use These APIs for Anything Other Than This

Always Check Return Codes and Types

A Simple Socket Server

Message structure

```
struct ss_msg_s {
    uint32_t _len;
    unsigned char _bytes[0];
};
```

0x0 – 0xffffffff

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD
PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
 <key>SSDoSomethingInteresting</key>
      <dict>
           <key>SSOperation>=</key>
           <integer>1</integer>
           <key>SSFile</key>
           <string>/sbin/launchd</string>
```

A Simple Socket Server

GCD for accept(3)

```
(void)fcntl(sockfd, F_SETFL, O_NONBLOCK); /* REQUIRED! */

dispatch_source_t s = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, sockfd, 0, q);
dispatch_source_set_event_handler(s, ^{
    struct sockaddr saddr;
    socklen_t slen = 0;

    int afd = accept(sockfd, (struct sockaddr *)&saddr, &slen));
    (void)fcntl(fd, F_SETFL, O_NONBLOCK);
    server_accept(afd, dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0));
});

/* REQUIRED! */
dispatch_source_set_cancel_handler(s, ^{
    dispatch_release(s);
    (void)close(sockfd);
});
dispatch_resume(s);
```

A Simple Socket Server

GCD to read(2)

```
void
server_accept(int fd, dispatch_queue_t q)
{
    __block size_t total = 0;
    size_t buff_sz = 10 * 1024 * 1024; /* Big enough. */
    unsigned char *buff = malloc(buff_sz);

    dispatch_source_t s = dispatch_source_create(DISPATCH_SOURCE_TYPE_READ, fd, 0, q);
    dispatch_source_set_event_handler(s, ^{
        if (server_read(fd, buff, buff_sz, &total)) {
            struct ss_msg_s *msg = (struct ss_msg_s *)buff;
            server_handle_request(fd, msg->_bytes, msg->_len);
            dispatch_source_cancel(s);
        }
    });
    dispatch_source_set_cancel_handler(s, ^{
        dispatch_release(s);
        free(buff);
    });
    dispatch_resume(s);
}
```

A Simple Socket Server

Marshaling the request

```
bool
server_read(int fd, unsigned char *buff, size_t buff_sz, size_t *total)
{
    bool result = false;
    struct ss_msg_s *msg = (struct ss_msg_s *)buff;

    unsigned char *track_buff = buff + *total;
    size_t track_sz = buff_sz - *total;
    ssize_t nbytes = read(fd, track_buff, track_sz);
    if (nbytes + *total >= sizeof(struct ss_msg_s)) {
        if (msg->_len == (*total - sizeof(struct ss_msg_s))) {
            result = true;
        }
    }
    *total += nbytes;
    return result;
}
```

Beware of Buffer Overflows!

A Simple Socket Server

Demuxing the request

```
void
server_handle_request(int fd, const void *buff, size_t total)
{
    CFDataRef data = CFDataCreate(NULL, buff, total);
    CFPropertyListRef plist = CFPropertyListCreateWithData(NULL, data,
                                    kCFPropertyListImmutable, NULL, NULL);

    /* Handle request, create reply (of a property list type). */

    CFDataRef replyData = CFPropertyListCreateData(NULL, (CFPropertyListRef)reply,
                                kCFPropertyListBinaryFormat_v1_0, 0, NULL);
    CFRelease(data);
    CFRelease(plist);
    CFRelease(reply);

    server_send_reply(fd, replyData);
}
```

A Simple Socket Server

GCD to send the reply

```
void
server_send_reply(int fd, dispatch_queue_t q, CFDataRef data)
{
    size_t total = sizeof(struct ss_msg_s) + CFDataGetLength(data);
    unsigned char *buff = (unsigned char *)malloc(total);

    struct ss_msg_s *msg = (struct ss_msg_s *)buff;
    msg->_len = total - sizeof(struct ss_msg_s);
    (void)memcpy(msg->_bytes, CFDataGetBytePtr(data), CFDataGetLength(data));

    __block unsigned char *track_buff = buff;
    __block size_t track_sz = total;
    dispatch_source_t s = dispatch_source_create(DISPATCH_SOURCE_TYPE_WRITE, fd, 0, q);

    /* Continues... */
}
```

A Simple Socket Server

GCD to send the reply

```
void __attribute__((continued))
server_send_reply(int fd, dispatch_queue_t q, CFDataRef data)
{
    dispatch_source_set_event_handler(s, ^{
        ssize_t nbytes = write(fd, track_buff, track_sz);
        if (nbytes != -1) {
            track_buff += nbytes;
            track_sz -= nbytes;
            if (track_sz == 0) {
                dispatch_source_cancel(s);
            }
        }
    });
    dispatch_source_set_cancel_handler(s, ^{
        (void)close(fd);
        free(buff);
        dispatch_release(s);
    });
    dispatch_resume(s);
}
```

A Simple Socket Server

Idle exiting

- Globals

```
dispatch_source_t g_timer_source = NULL;
uint32_t g_transaction_count = 0;
```

- Initialization

```
g_timer_source = dispatch_source_create(DISPATCH_SOURCE_TYPE_TIMER, 0, 0, q);
dispatch_time_t t0 = dispatch_time(DISPATCH_TIME_NOW, 20llu * NSEC_PER_SEC);
dispatch_source_set_timer(g_timer_source, t0, 0llu, 0llu);
```

A Simple Socket Server

Idle exiting

- Add to server_read()…

```
if (OSAtomicIncrement32(&g_transaction_count) - 1 == 0) {
    dispatch_source_set_timer(g_timer_source, DISPATCH_TIME_FOREVER, 0llu, 0llu);
}
```

- Add to server_send_reply()…

```
if (OSAtomicDecrement32(&g_transaction_count) == 0) {
      dispatch_time_t t0 = dispatch_time(DISPATCH_TIME_NOW, 20llu * NSEC_PER_SEC);
      dispatch_source_set_timer(g_timer_source, t0, 0llu, 0llu);
}
```

A Simple Socket Server

Instant Off

- Add to launchd.plist

```
<key>EnableTransactions</key>
<true/>
```

- Open new transaction upon receiving request
- Close transaction after sending reply

A Simple Socket Server

Instant Off

- Add to server_read()…

```
vproc_transaction_t t = vproc_transaction_begin(NULL);
```

- Add to server_send_reply()…

```
vproc_transaction_end(NULL, t);
```

A Simple Socket Server

Instant Off

- Add SIGTERM handler…

```
(void)signal(SIGTERM, SIG_IGN); /* REQUIRED! */

dispatch_source_t s = dispatch_source_create(DISPATCH_SOURCE_TYPE_SIGNAL, SIGTERM, 0, q);
dispatch_source_set_event_handler(s, ^{
    g_accepting_transactions = false;
});
dispatch_resume(s)
```

- Check g_accepting_transactions in server_handle_request()
- Close accept(2)'s socket if not

Launch-on-Demand

Privileged helper tools

- GUI apps need to elevate privilege
- Preserve drag-n-drop install
- Avoid setuid tricks
- On-demand daemon

Launch-on-Demand

Secure handshake

- ServiceManagement framework provides API
- Securely tie application and helper together
- Application specifies tool's identity (and vice-versa)
- Identity verifiable through code signing

Launch-on-Demand

Application side

- Add to application's Info.plist

```
<key>SMPrivilegedExecutables</key>
<dict>
    <key>com.apple.wwdc.HelperTool</key>
    <string>identifier com.apple.wwdc.HelperTool and certificate leaf[subject.CN]
            = "WWDC Developer"</string>
</dict>
```

Launch-on-Demand

Tool side

- Embed in tool's Info.plist

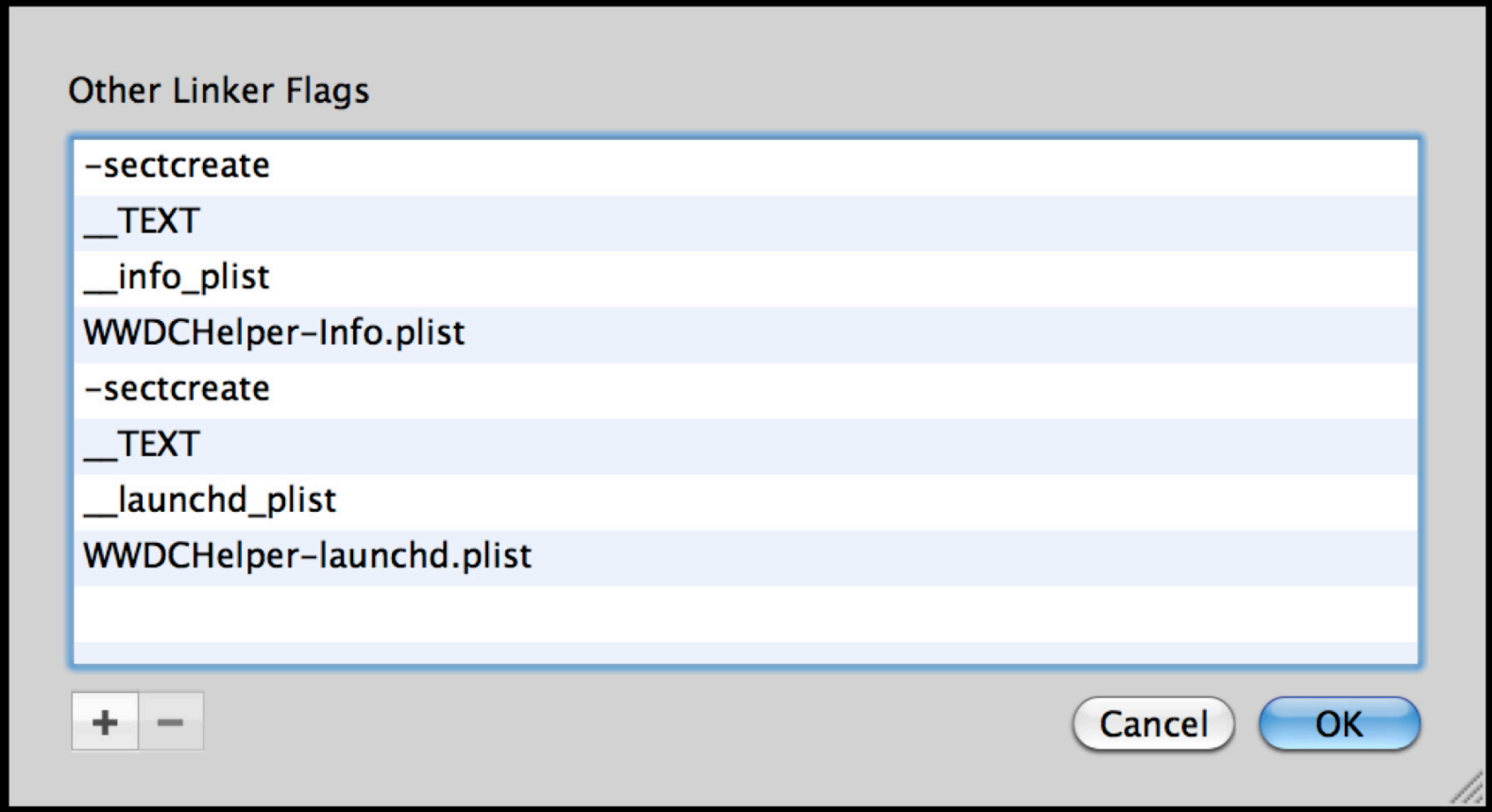
```
<key>SMAuthorizedClients</key>
<array>
    <string>identifier com.apple.wwdc.SMApplication and certificate
            leaf[subject.CN] = "WWDC Developer"</string>
</array>
```

Launch-on-Demand

Requirements

- No Program or ProgramArguments in tool's launchd.plist
- Both launchd.plist and Info.plist part of tool's code signature
- Tools live in Contents/Library/LaunchServices
- Must be named identically to launchd label

Launch-on-Demand



Launch-on-Demand

Magic API call

```
#include <ServiceManagement/ServiceManagement.h>

AuthorizationItem authItem   = { kSMRightBlessPrivilegedHelper, 0, NULL, 0 };
AuthorizationRights authRights= { 1, &authItem };
AuthorizationFlags flags  = kAuthorizationFlagDefaults |
    kAuthorizationFlagInteractionAllowed| kAuthorizationFlagPreAuthorize |
    kAuthorizationFlagExtendRights;

AuthorizationRef authRef = NULL;
OSStatus status = AuthorizationCreate(&authRights, kAuthorizationEmptyEnvironment,
                                      flags, &authRef);
if (status == noErr) {
    CFErrorRef *error = NULL
    Boolean result = SMJobBless(kSMDomainSystemLaunchd,
                          CFSTR("com.apple.wwdc.HelperTool"), authRef, &error);

    /* Check result, issue request. */
}
```

Wrapping Up…

- Basics of Launch-on-Demand
- Why Launch-on-Demand is better
- Writing a GCD-based on-demand server
- Bootstrapping a privileged helper

Related Sessions

Introducing Blocks and Grand Central Dispatch on iPhone	Russian Hill Wednesday 11:30AM
Simplifying iPhone App Development with Grand Central Dispatch	Mission Friday 10:15AM

Designing for launchd (WWDC 2009)	Available on iTunes
Managing User Privileges and Operations with Authorization Services (WWDC 2009)	Available on iTunes
Assigning Your Application an Identity with Code Signing (WWDC 2009)	Available on iTunes

Labs

Mac OS X Performance Lab	Application Frameworks Lab C Friday 11:15AM
Grand Central Dispatch Lab	CoreOS Lab A Friday 11:15AM

More Information

Craig Keithley
Security and I/O Technology Evangelist
keithley@apple.com

Paul Danbold
CoreOS Technology Evangelist
danbold@apple.com

Apple Developer Forums
http://devforums.apple.com

More Information

Website and Sources

http://launchd.macosforge.org
http://libdispatch.macosforge.org

E-mail Discussion List

launchd-dev@lists.macosforge.org
libdispatch-dev@lists.macosforge.org

Documentation

launchd(8)
launchd.plist(5)
launchctl(1)
dispatch(3)
ServiceManagement/ServiceManagement.h

More Information

Sample Code

ssd
http://developer.apple.com/wwdc/mac/library/samplecode/ssd/index.html

ServiceManagement Sample
http://developer.apple.com/wwdc/mac/library/samplecode/SMJobBless/index.html

BetterAuthorizationSample
http://developer.apple.com/mac/library/samplecode/BetterAuthorizationSample/Introduction/Intro.html

Q&A


WWDC10

The last slide after the logo is intentionally left blank for all presentations.