Multithreading and multitasking in iOS

Asynchronous and parallel tasks

- Asynchronous tasks have undetermined order of execution start (or end) in respect to each other. It's a simulated multitasking.

- True parallel execution makes each task execute on each computing/processing unit. Thus, computing two or move tasks effectively in parallel.

Means of concurrent/async/ parallel execution in iOS.

- Methods of NSObject
- NSThread class
- NSOperation/NSOperationQueue
- GCD
- POSIX threads

Methods of NSObject

- performSelectorInBackground: withObject:
- performSelectorOnMainThread: withObject: waitUntilDone:
- performSelector: onThread: withObject: waitUntilDone:
- performSelector: withObject: afterDelay:
- target and argument are retained when methods are called.

NSThread class

- detachNewThreadSelector: toTarget: withObject:
- start and main
- cancel
- sleepUntilDate: and sleepForTimeInterval:
- currentThread, mainThread, isMainThread
- target and argument are retained when methods are called
- Make sure your thread has autorelease pool and do not throw uncatched exceptions

NSOperation/ NSOperationQueue

- NSOperation is able to execute synchronously and asynchronously.
- NSOperation has subclasses: NSInvocationOperation, NSBlockOperation, your custom subclasses.
- NSOperationQueue is a queue of NSOperations to be executed either concurrently or serially.

NSOperation

- start and main
- cancel
- addDependency:, removeDependency:
- completionBlock
- you are responsible for making operations async or sync if you plan to run them manually, not placing on a Operations Queue.

NSInvocationOperation

- initWithTarget: selector: option:
- initWithInvocation: - invocation is an instance of NSInvocation
- has property ‘result’

NSBlockOperation

- blockOperationWithBlock:
- addExecutionBlock:

NSOperationQueue

- addOperation:, addOperationWithBlock:, addOperations: waitUntilFinished:
- cancelAllOperations, waitUntilAllOperationsAreFinished
- maxConcurrentOperationCount
- currentQueue, mainQueue

Async and sync operations

- To make async NSOperation override: start, asynchronous, executing, finished.
- To make sync NSOperation override: main
- To make NSOperation available both for sync and async override main in addition to methods of async NSOperation. Call start to run async, call main to run sync or put on NSOperationQueue.

Grand Central Dispatch

- dispatch_queue_t
- Queues: four system concurrent queues of priorities DISPATCH_QUEUE_PRIORITY_LOW, …_DEFAULT, …_HIGH, …BACKGROUND.
- You can create DISPATCH_SERIAL_QUEUE queues.

Create queues

- dispatch_queue_create(const char *, dispatch_queue_attr_t)
- dispatch_get_global_queue(long, long)
- dispatch_get_main_queue

Run/schedule tasks

- dispatch_sync, dispatch_async(dispatch_queue_t, dispatch_block_t)
- dispatch_after(dispatch_time_t, dispatch_queue_t, dispatch_block_t), dispatch_time(dispatch_time_t, int64_t delta)
- dispatch_apply(size_t, dispatch_queue_t, void(^)(size_t))
- dispatch_once(dispatch_once_t, dispatch_block_t)

Dispatch Groups

- Group is a set of tasks that run async (or sync, depending on a target queue) to each other, but can notify a caller on completion.
- Group can be used as a DIY semaphores.

Create and use Dispatch Group

- dispatch_group_create, dispatch_group_release
- dispatch_group_async
- dispatch_group_enter, dispatch_group_leave, dispatch_group_wait,
- dispatch_group_notify

Precautions

- Do not ever schedule a task on the same queue with dispatch_sync. Like, do not call dispatch_sync(dispatch_get_main_queue(), …) on the main thread.
- Do not forget to release dispatch objects (with dispatch_release) for ARC < 6.0 and MRC.

Synchronization

- @property (atomic/nonatomic) …
- @synchronized(…)
- Foundation primitives
- GCD approach
- OSAtomic

Atomic and non-atomic properties

```
- (void)setProperty:(id)value {
    [_lock lock];
    id prevValue = self->_value;
    self->_value = [value retain];
    [prevValue release];
    [_lock unlock];
}
- (id)property {
   [_lock lock];
   id value = [[self->_value retain] autorelease];
   [_lock unlock];
   return value;
}
```

@synchronized(object)

- Recursive lock RL is created on ‘object’
- Then RL is locked
- The statements in the block are executed
- The RL is unlocked and released
- As we have a recursive lock, then it’s pretty safe to lock with @synchronized(…) in recursive methods

Foundation primitives

- NSLocking protocol which has lock and unlock methods
- NSLock
- NSRecursiveLock
- NSConditionLock
- NSCondition

NSLock, NSRecursiveLock

- These are basically mutexes with simple API: lock, tryLock, lockBeforeDate: and unlock.

NSCondition

- Kind of a semaphore: if you need to manage shared resources among readers and writers, for instance.
- lock, unlock, wait/waitUntilDate:, signal, broadcast.
- Calls to wait, signal, broadcast must be protected with locks.

```
// Thread 1

[condition lock];

while (!predicate()) [condition wait];

// … do smth. in case predicate is positive

[condition unlock];



// Thread 2

[condition lock];

// .. Do the stuff.

[condition signal];

[condition unlock];
```

NSConditionLock

- Used when you need to order execution of codeblocks.
- initWithCondition:
- lockWhenCondition:
- unlockWithCondition:

GCD approaches

- It’s recommended to try to redesign your code and idea from locks and primitives to use of different queue objects like queues themselves (concurrent and serial) and groups.

- dispatch_semaphore: dispatch_semaphore_create, dispatch_semaphore_wait (decrements semaphore), dispatch_semaphore_signal (increments semaphore)

- dispatch_barrier_sync, dispatch_barrier_async(dispatch_queue_t, dispatch_block_t)

OSAtomic

- Group of functions for atomic operations on scalar items (test and set, binary boolean, binary arithmetics), and a primitive OSSpinLock and OSAtomicQueue

NSRunLoop

- Runloop is an event-processing machinery behind UI events, timer events and other kind of sources (like ports or NSURLConnection)
- Runloop is like an eternal loop of processing events from different sources
- Each NSThread has a preset but not running Runloop (except main thread)
- Runloop runs is several modes that process events set for particular mode

NSRunLoop

- currentRunLoop, mainRunLoop
- run, runMode:, runUntilDate:
- addTimer:
- addPort: forMode:, removePort: forMode:

NSTimer

- NSTimer allows users to run scheduled tasks after a certain amount of time.
- scheduledTimerWithTimeInterval: target: selector: userInfo: repeats:
- initWithFireDate: interval: target: selector: userInfo: repeats:
- invalidate
- fire
- Runs only on threads with activated Run Loops.
- Retains its target and userInfo
- NSObject.performSelector:withObject:afterDelay: uses timer.
- Unscheduled timer can be scheduled on a run loop later with NSRunLoop.addTimer:

Coding time.