Glossary

application A specific style of program that displays a graphical interface to the user.

asynchronous design approach The principle of organizing an application around blocks of code that can be run concurrently with an application's main thread or other threads of execution. Asynchronous tasks are started by one thread but actually run on a different thread, taking advantage of additional processor resources to finish their work more quickly.

block object A C construct for encapsulating inline code and data so that it can be performed later. You use blocks to encapsulate tasks you want to perform, either inline in the current thread or on a separate thread using a dispatch queue. For more information, see *Blocks Programming Topics*.

concurrent operation An operation object that does not perform its task in the thread from which its `start` method was called. A concurrent operation typically sets up its own thread or calls an interface that sets up a separate thread on which to perform the work.

condition A construct used to synchronize access to a resource. A thread waiting on a condition is not allowed to proceed until another thread explicitly signals the condition.

critical section A portion of code that must be executed by only one thread at a time.

custom source A dispatch source used to process application-defined events. A custom source calls your custom event handler in response to events that your application generates.

descriptor An abstract identifier used to access a file, socket, or other system resource.

dispatch queue A Grand Central Dispatch (GCD) structure that you use to execute your application's tasks. GCD defines dispatch queues for executing tasks either serially or concurrently.

dispatch source A Grand Central Dispatch (GCD) data structure that you create to process system-related events.

descriptor dispatch source A dispatch source used to process file-related events. A file descriptor source calls your custom event handler either when file data is available for reading or writing or in response to file system changes.

dynamic shared library A binary executable that is loaded dynamically into an application's process space rather than linked statically as part of the application binary.

framework A type of bundle that packages a dynamic shared library with the resources and header files that support that library. For more information, see *Framework Programming Guide*.

global dispatch queue A dispatch queue provided to your application automatically by Grand Central Dispatch (GCD). You do not have to create global queues yourself or retain or release them. Instead, you retrieve them using the system-provided functions.

Grand Central Dispatch (GCD) A technology for executing asynchronous tasks concurrently. GCD is available in OS X v10.6 and later and iOS 4.0 and later.

input source A source of asynchronous events for a thread. Input sources can be port based or manually triggered and must be attached to the thread's run loop.

joinable thread A thread whose resources are not reclaimed immediately upon termination. Joinable threads must be explicitly detached or be joined by another thread before the resources can be reclaimed. Joinable threads provide a return value to the thread that joins with them.

library A UNIX feature for monitoring low-level system events. For more information see the `kqueue` man page.

Mach port dispatch source A dispatch source used to process events arriving on a Mach port.

main thread A special type of thread created when its owning process is created. When the main thread of a program exits, the process ends.

mutex A lock that provides mutually exclusive access to a shared resource. A mutex lock can be held by only one thread at a time. Attempting to acquire a mutex held by a different thread puts the current thread to sleep until the lock is finally acquired.

Open Computing Language (OpenCL) A standards-based technology for performing general-purpose computations on a computer's graphics processor. For more information, see *OpenCL Programming Guide for Mac*.

operation object An instance of the `NSOperation` class. Operation objects wrap the code and data associated with a task into an executable unit.

operation queue An instance of the `NSOperationQueue` class. Operation queues manage the execution of operation objects.

private dispatch queue A dispatch queue that you create, retain, and release explicitly.

process The runtime instance of an application or program. A process has its own virtual memory space and system resources (including port rights) that are independent of those assigned to other programs. A process always contains at least one thread (the main thread) and may contain any number of additional threads.

process dispatch source A dispatch source used to handle process-related events. A process source calls your custom event handler in response to changes to the process you specify.

program A combination of code and resources that can be run to perform some task. Programs need not have a graphical user interface, although graphical applications are also considered programs.

reentrant Code that can be started on a new thread safely while it is already running on another thread.

run loop An event-processing loop, during which events are received and dispatched to appropriate handlers.

run loop mode A collection of input sources, timer sources, and run loop observers associated with a particular name. When run in a specific "mode," a run loop monitors only the sources and observers associated with that mode.

run loop object An instance of the `NSRunLoop` class or `CFRunLoopRef` opaque type. These objects provide the interface for implementing an event-processing loop in a thread.

run loop observer A recipient of notifications during different phases of a run loop's execution.

semaphore A protected variable that restricts access to a shared resource. Mutexes and conditions are both different types of semaphore.

signal A UNIX mechanism for manipulating a process from outside its domain. The system uses signals to deliver important messages to an application, such as whether the application executed an illegal instruction. For more information see the `signal` man page.

signal dispatch source A dispatch source used to process UNIX signals. A signal source calls your custom event handler whenever the process receives a UNIX signal.

task A quantity of work to be performed. Although some technologies (most notably Carbon Multiprocessing Services) use this term differently, the preferred usage is as an abstract concept indicating some quantity of work to be performed.

thread A flow of execution in a process. Each thread has its own stack space but otherwise shares memory with other threads in the same process.

timer dispatch source A dispatch source used to process periodic events. A timer source calls your custom event handler at regular, time-based intervals.