http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:hyb3.5:hrte:sdk

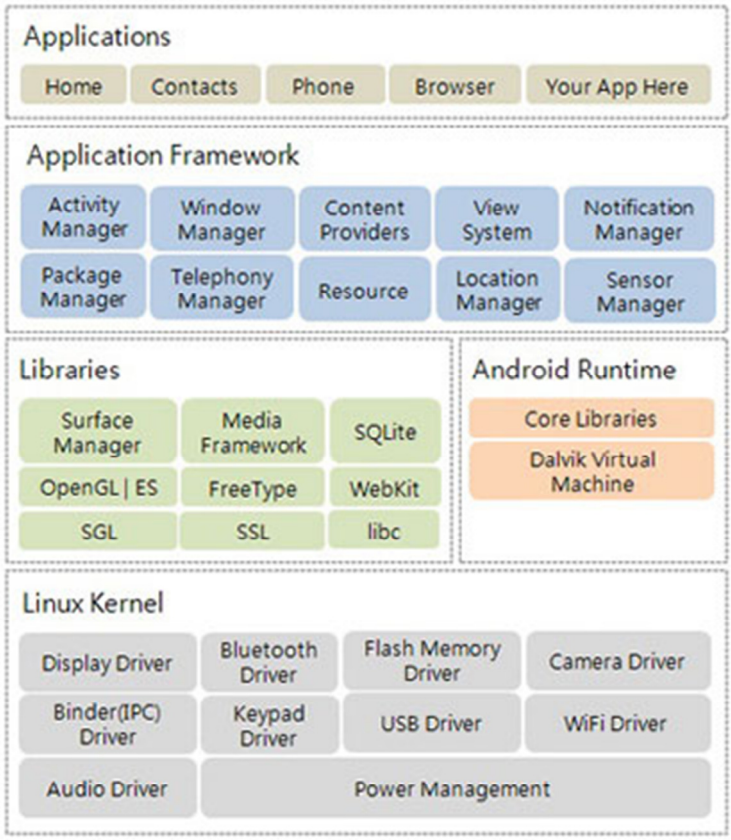
Android

Outline

Android is a software stack for mobile devices and comprises middleware, operating system and core applications.
Android SDK supports tools and APIs that are necessary for development of Android platform application using the programming languages available for Java.

Concept

Refer to the following for the basic Android structure comprising Applications, Application Framework, Libraries, Android Runtime and Linux Kernel.



Composition	Description
Applications	E-mail, SMS, Calendar, Map, Browser, Contact and other core applications.
Application Framework	Application Frameworks that allow the resources used as required by Android.
Libraries	Functions available for other applications

Android Runtime	Core library intended for operating system
Linux Kernel	Kernel information that administers Linux operating system

Description

Applications

Android OS has e-mail client, SMS, Calender, Map, Browser Contact and other core applications itself. These applications are built based on Java Programming Language.

Application Framework

Android developer can access the framework API used, identical to the core application. These architectures have been designed to allow re-use of component much easier. Meanwhile, the applications may allow the external applications to use its own functions, and vise versa (subject to security restrictions administered by framework). Such a mechanism makes feasible substitution of components by users. At the bottom of the application you have the following sets of service and system:

- A multitude of sets of expandable views for development of applications including, without limitation, list, grid, text box, button and embeddable web browser.
- Content Provider that allows an application to access the data of other application (like contacts information) or share its own data.
- Resource Manager that provides access to non-code resources such as localized strings, graphics and layout file.
- Notification Manager that allows applications to activate customized notification messages.
- Activity Manager that administers the lifecycle of application and navigation history of the general application (back stack only).

Libraries

Android contains C/C++ Libraries used for a variety of Android-based system components. These libraries are provided to the developers via Android Application Framework. See the following for a handful of core libraries available:

- System C library : Standard System C Library derived out of BSD libc to fit the embedded Linux-based devices.
- Media Libraries : Structured based on PacketVideo's OpenCORE. Compatible with still images such as MPEG4, H.264, MP3, AAC, AMR, JPG and PNG for audial and visual recording and play.
- Surface Manager: Provides access managements for display sub-systems and 2D and 3D graphic layers used by a variety of applications.
- LibWebCore : A state-of-the-art web browser engine compatible with both Android browser and embeddable web-view.
- SGL : Very basis of 2D graphics engine.
- 3D Libraries : Implemented based on OpenGL ES 1.0 API. Users Hardware's 3D Accelerator or 3D Software Rasterizer.
- FreeType - Rendering Engines for Bit Map and Vector Font Rendering.
- SQLite - A powerful and light relational database engine available for all sorts of applications.

Android Runtime

Android's core library contains the greatest part what is available in Java Programming Language. Android applications are designed to have its own instances against Dalvik virtual machine and executed within its own process. Meanwhile, Dalvik is designed to execute a variety of virtual machines within a single device. Dalvik virtual machine executes the unique Dalvik Executable format (.dex) designed to use the memory to the minimal extent. The virtual machine is registered to

be compiledby Java Language Compiler and executes the class converted into .dex format thanks to “dx” contained in SDK. Dalvik virtual machine is thus dependent on the Linux kernel for the basic functions such as threading and low-level memory management.

Linux Kernel

Android is dependent on Linux 2.6 (or, for Android 4.0 or better, Linux 3.x) for core system services such as security, memory administration, process administration, network stack and driver model. Meanwhile, Linux Kernel also serves as the abstract layer between the hardware and Android Platform Stack.

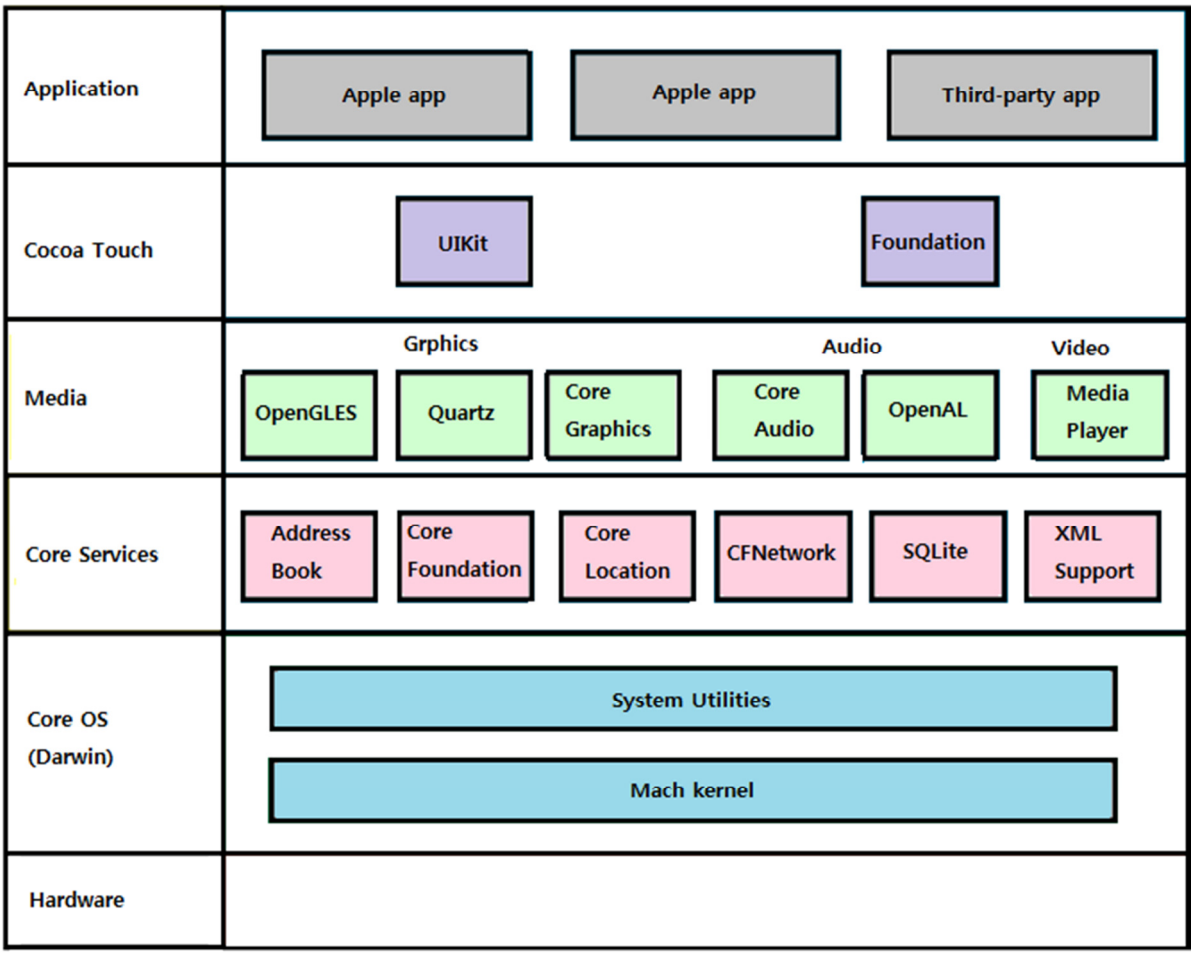
References

Android platform : http://developer.android.com

iOS

Outline

Concept



Layer	Description

iOS Kernel	iOS Kernel : Mach-based. Same with Mac OS X.
Core OS and Core Services Layer	Basic iOS Interfaces. Data types, Bonjour and Network Socket.
Media Layer	2D/3D, audio, video and other base functions. OpenGL ES, Quartz Core Audio and Core Animation.
Cocoa Touch Layer	Objective-C-based. Offers the fundamental infrastructure to generate application program using various framework.

Description

Applications

iOS has e-mail client, SMS, Calender, Map, Browser Contact and other core applications itself. These applications are built based on Objective-C 2.0 Programming Language.

Application Framework

iOS architecture resembles Mac OS X, in that it mediates between underlying hardware and on-screen application. An iOS application does not communicate with hardware directly but by way of Well-defined System Interface that protects application from hardware changes, referred to as "abstraction" that allows the application compatible with multiple devices.
It is thus advised that the developer uses the frameworks of superior hierarchies when coding for application. Keep in mind that the frameworks of superior hierarchies are capable of object-oriented abstraction in their inferior counterparts. Abstraction relieves the developer from the complicated coding requirements from sockets and threads recorded or encapsulated. Note that the technologies unique to the frameworks of superior hierarchies are still available regardless of abstraction. It is the developer's option to use the frameworks of inferior hierarchies that are not available in the superior hierarchies. Ensuring good understanding about the frameworks and class information would thus help the developer make the best of iOS applications.
Refer to the following descriptions for the information of layers available, most notably being **Cocoa Touch Layer**, **Media Layer**, **Core Services Layer** and **Core OS Layer**.

Cocoa Touch Layer

Applications mainly using graphics and events and requiring access to Contacts uses Cocoa Touch Layer.

Framework	Description
AddressBookUI.framework	UI framework for editing address books
EventKitUI.framework	Framework for verification and editing of calender data
GameKit.framework	Framework for Game Center
iAd.framework	Framework for iAd
MapKit.framework	Framework for MapKit
MessageUI.framework	UI framework for messaging

Twitter.framework	Framework for Twitter
UIKit.framework	UI Kit Framework

Media Layer

Media Layer provides a variety of frameworks for multimedia available for framework and mobile devices.

Framework	Description
AssetsLibrary.framework	Framework for photo albums
AudioToolbox.framework	Framework for audio recording and play and format conversion
AudioUnit.framework	Framework for audio unit and devices
AVFoundation.framework	Framework for audio / video recording, editing and play and audio sessions
CoreAudio.framework	Framework for data type, audio stream, complicated buffer and time values
CoreGraphics.framework	Framework for 2D rendering, gradient, image, color control and PDF documents
CoreImage.framework	Framework for image processing and video image configuration
CoreMIDI.framework	Framework for hardware keyboard, synthesizer and MIDI devices
CoreText.framework	Framework for font styles
CoreVideo.framework	Framework for low-level, pipe-line based API for movie and video works
GLKit.framework	Saves significant amount of time for development of OpenGL ES Framework for mathematics library, rendering loop, etc.
ImageIO.framework	Framework for reading, writing, color control and image metadata for image file types
MediaPlayer.framework	Framework for music and video plays
OpenAL.framework	Framework for quality performance and advanced audio quality
OpenGLES.framework	Framework for 2D and 3D processing
QuartzCore.framew	Framework for advanced animation

ork

Core Services Layer

Application uses Core Services Layer to access the fundamental iOS services, files, low-level data, Bonjour Service or network sockets. Core Services Layer is also preferred over Cocoa Touch Layer and Media Layer for access to data.

Framework	Description
Accounts.framework	Framework for account accessibility information
AddressBook.framework	Framework for contacts information
CFNetwork.framework	Framework for network service access and network composition changes
CoreData.framework	Framework for core data
CoreFoundation.framework	Framework for basic system services
CoreLocation.framework	Framework for compass and location information
CoreMedia.framework	Framework for visual media
CoreMotion.framework	Framework for gyroscope
CoreTelephony.framework	Framework for core telephony
EventKit.framework	Framework for calendar data
Foundation.framework	Foundation framework for NS Object.
MobileCoreServices.framework	Framework for access to standard types and constants
NewsstandKit.framework	Framework for News Stand
QuickLook.framework	Framework for Quick Look
StoreKit.framework	Framework for In-app Purchase
SystemConfiguration.framework	Framework for network availability and status
UIAutomation.framework	

Core OS Layer

Core OS Layer administers virtual memory system, thread, file system and network communication. Note that Core OS Layer contains kernel information, driver and iOS basic interfaces.

Framework	Description
Accelerate.framework	Framework for complicated mathematics and image calculations

CoreBluetooth.framework	Framework for Bluetooth
ExternalAccessory.framework	Framework for external accessories to iOS devices
Security.framework	Framework for data security
System.framework	System framework

Visit https://developer.apple.com/library/ios/navigation/#section=Frameworks for more details.

Libraries

Libraries is a series of complied files containing codes (functions or classes). Frequently used functions can be referred in the form of libraries for easier maintenance and debugging, as well as faster compilation. Without libraries, the developer must be bothered by modifying main codes every time modification becomes necessary. With libraries, the time-consuming chores are no longer necessary as all you need to do to modify main codes is to compile the concerned library(s) and link to the concerned main function, saving a heap of time. Plus, iOS's dynamic library system boasts its unique functions that are much advanced from the conventional static libraries.
However, the developer has a total of three options to determine which type of library is to be used, depending on Loading Time Allowance: (Typical) Static Library, Shared Library and Dynamic Library

Category	**Description**
Static library	A simple set of libraries concluded by an object file (.o). Features specific extension (.a). Boasts simplicity. Less flexible loading time is involved in the course of compilation. In recent days, less popular than other library options. Involves larger binary due to loading time.
Shared Library	Involves loading time when the program is executed. Once created, library is shared over the course of execution of program. Makes feasible flexible development. Does not involve loading time, contrary to static library. Unfortunately, 1-5% latency compared to static library is unavoidable despite smaller volume of program. Latency typically left inappreciable, though.
Dynamic Library	Loading takes place while the program is being executed. Best-fit for plug-in module, etc. Suitable for development of the flexible application where the developer can choose the desired libraries to be executed.

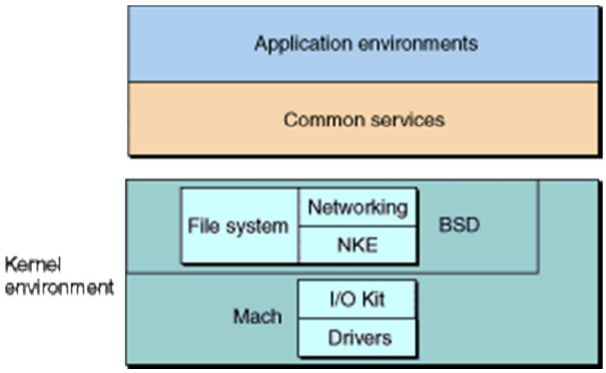
iOS Runtime

Speed and security are keys to iOS runtime. Refer to the following paragraphs for how iOS runtime environment can be described and is used most optimally:

- The User Interface
- Fast Launch, Short Use
- Specialized System Behaviors
- Security
- Use Memory Efficiently
- The File System
- Backup and Restore
- The Simulator

Visit About iOS App Programming for more information.

Unix Kernel



Being the very basis of iOS components, iOS Kernel very much resembles Unix Kernel. Designed for multi-user security, UNIX Kernel was first developed when the developer entitled “Next” deployed Unix Kernel Mach(Multiple Asynchronously Communication Hosts) OS with GUI, which is what OS X is based upon. This OS X is also adopted by iOS.

With OS X Kernel commonly using Mach Kernel, (Apple TV, iPhone, Mac OS X, etc.) which has Kernel Extension as its part. (USB Input and VPN are the Kexts that iPhone uses.) At the very bottom of Mac OS X there is a Kernel, followed by BSD Unix of Mac OS X. In Unix, we use the term ‘userland’ to refer to the processes run outside the kernel. BSD Unix Userland in mackintosh environment is not verifiable by the desktop users. (available when the console is open.)

Userland serves as a mediator between Application Program and Hardware. In Userland, a multitude of application programs (processes and threads) engages in memory management and resource distribution.

Visit Kernel Architecture Overview for more information.

References

Title	Link
Apple Developer Library	https://developer.apple.com/library/ios/navigation/
Apple Developer Library - Framework	https://developer.apple.com/library/ios/navigation/#section=Frameworks
iOS Programming Guide	About iOS App Programming