简述 IOS

Karry

1 Itroduction:

iOS 是 Darwin 的 ARM 变体，源自 BSD，类 UNIX 内核，再加上 Apple 自己的 Mach 内核扩展系统。

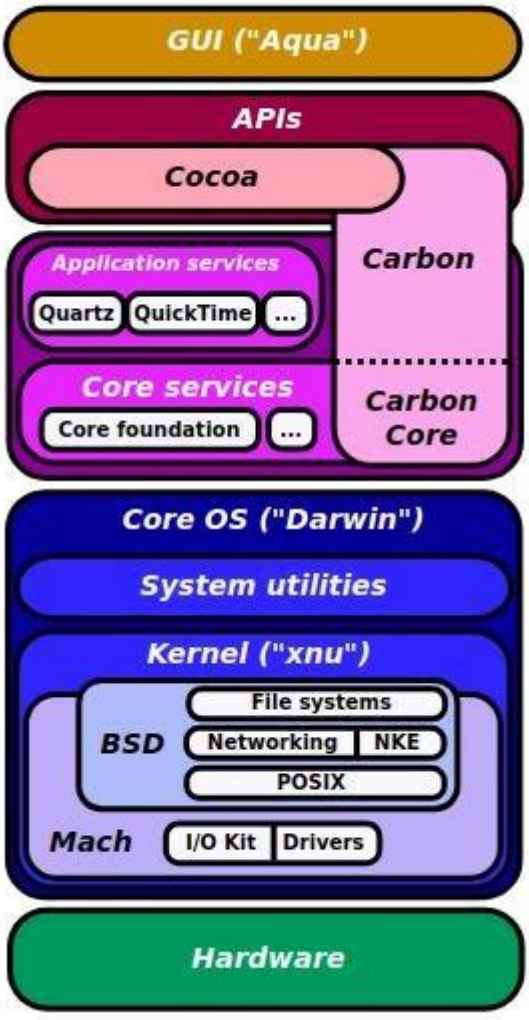

Fig 1. Framework of iOS

2 History:

2005年，当史蒂夫·乔布斯（Steve Jobs）开始规划iPhone时，他可以选择"缩小Mac，这将是一项史诗般的工程壮举，要么扩大iPod"。乔布斯赞成前一种方法，但分别由斯科特·福斯特尔（Scott Forstall）和托尼·法德尔（Tony Fadell）领导的Macintosh和iPod团队在内部竞争中相互竞争，福斯特尔通过创建iPhone操作系统获胜。这一决定使iPhone作为第三方开发人员的平台取得了成功：使用众所周知的桌面操作系统作为其基础，许多第三方Mac开发人员能够以最少的再培训为iPhone编写软件。

该操作系统于2007年1月9日在Macworld Conference&Expo上与iPhone一起亮相，并于当年6月发布。史蒂夫·乔布斯在1月份发布时声称："iPhone运行OS X"并运行"桌面级应用程序"，但在iPhone发布时，该操作系统被重新命名为"iPhone OS"。最初，第三方原生应用程序不受支持。乔布斯的理由是，开发人员可以通过Safari网络浏览器构建Web应用程序，这些应用程序"将像iPhone上的原生应用程序一样"。2007年10月，苹果宣布正在开发一个原生软件开发工具包（SDK），并计划"在二月份将其交到开发者手中"。2008年3月6日，苹果公司举行了新闻发布会，宣布了iPhone SDK。2010年6月，**苹果将iPhone OS更名为"iOS"**。思科十多年来一直使用"IOS"商标作为其路由器上使用的操作系统IOS。为了避免任何潜在的诉讼，苹果从思科获得了"IOS"商标的许可。

3 Fearure:

3.1 接口

iOS用户界面基于直接操作，使用多点触控手势，如轻扫、点击、捏合和反向捏合。界面控制元素包括滑块、开关和按钮。一些应用程序使用内部加速度计来响应设备的晃动（一个常见的结果是撤消命令）或三维旋转设备（一个常见的结果是在纵向和横向模式之间切换）。

iOS设备启动到**主屏幕**，这是iOS设备上的主要导航和信息"中心"，类似于个人计算机上的桌面。iOS主屏幕通常由应用程序图标和小部件组成;应用图标会启动关联的应用，而小部件会直接在主屏幕上显示实时的自动更新内容。**屏幕顶部**是一个状态栏，显示有关设备及其连接的信息。状态栏本身包含两个元素：控制中心和通知中心。在新款iPhone上，控制中心可以从缺口的*右上角"拉下"*，从而可以访问各种开关以更快地管理设备，而无需打开"设置"。可以管理亮度，音量，无线连接，音乐播放器等。相反，从*左上角*滚动到底部将打开通知中心，在最新版本的iOS中，它与锁屏非常相似。它按时间顺序显示通知，并按应用程序对通知进行分组。从某些应用程序的通知中，可以直接进行交互，例如直接从它回复消息。**通知以两种模式发送**，一种是显示在锁定屏幕上并由独特声音发出信号的重要通知，并伴有警告横幅和应用锁屏提醒图标，另一种是辅助模式，其中通知显示

在"通知中心"中，但不会显示在锁定屏幕上，也不会由警告横幅指示。 徽章图标或声音。可以选择来自应用程序的通知是否可以显示在锁定屏幕，通知中心，横幅或所有三个屏幕上; 横幅应该是临时的还是永久性的; 激活或停用声音;选择是否按应用分组，以及锁定时是否显示预览。可以关闭不需要的应用程序通知。较旧的通知会在几天后自动删除。

主屏幕可能由**多个页面组成**，用户可以在页面之间来回滑动，其中一种方法是按住每个页面上显示的"点"并向左或向右滑动。在最后一页的右侧，应用程序库列出并分类设备上安装的应用程序。每个类别中的应用都根据其使用频率进行排列。除了建议应用的类别外，"最近"类别还会将最近安装的应用与最近访问的应用剪辑一起列出。用户可以搜索所需的应用，也可以按字母顺序浏览应用。

在 iOS 上，主页按钮通常位于右上角。要在应用程序中返回，几乎总是有一个"后退"按钮，一般情况下用户可以通过4种不同的方式返回：

- 按显示屏左上角的"后退"按钮
- 从屏幕左边缘向右轻扫（手势）
- 按屏幕右上角的"完成"操作
- 在屏幕内容上向下滚动

3.2 应用：

iOS应用程序大多使用UIKit（一种编程框架）的组件构建。它允许应用程序与操作系统具有一致的外观和感觉，但提供自定义。元素会随着 iOS 的更新而自动更新，包括新的界面规则。UIKit元素的适应性很强，这允许开发人员设计一个在任何iOS设备上看起来相同的应用程序。除了定义 iOS 界面之外，UIKit 还定义了应用程序的功能。

3.3 系统字体：

iOS最初使用**Helvetica**作为系统字体。苹果在iPhone 4及其Retina Display上专门改用 Helvetica Neue，并保留了Helvetica作为 iOS 4 上旧款iPhone设备的系统字体。在iOS 7中，苹果宣布将系统字体更改为Helvetica Neue Light，这一决定引发了对低分辨率移动屏幕不当使用轻薄字体的批判。苹果最终选择了Helvetica Neue。iOS 7 的发布还引入了通过"设置"缩放文本或应用其他形式的文本辅助功能更改的功能。在iOS 9中，苹果将字体改为**旧金山(San Francisco)**，这是苹果设计的字体，旨在最大限度地提高其产品阵容的易读性和字体一致性。

3.4 硬件

iOS的主要硬件平台是ARM架构（ARMv7，ARMv8-A，ARMv8.2-A，ARMv8.3-A）。 iOS 7 之前的 iOS版本只能在配备32 位ARM 处理器（ARMv6和 ARMv7-A架构）的 iOS 设备上运行。2013年，iOS 7]发布，完全支持64位（包括原生64位内核、库、驱动程序以及所有内置应用程序），此前苹果宣布，随着苹果A7芯片的推出，他们将切换到64位ARMv8-A处理器。App Store中的所有应用程序也强制实施了64位支持。所有提交到 App Store 的新 app 的截止日期为 2015 年 2 月，以及所有 app 更新的提交到 App Store 的截止日期为 2015 年 6 月 1 日。iOS 11不再支持所有搭载32位ARM处理器和32位应用程序的iOS设备，而iOS仅支持64位

4 The Difference Between Android and iOS

4.1 系统和框架结构

4.1.1 Android系统的底层建立在Linux系统之上，而ios基于UNIX系统。

这一点造成了Android与iOS的生态不同：Android完全开源，任何软件开发商或者个人都能开发安卓的软件；ios完全封源开发。

> Linux的大多数版本都开源，而Unix系统主要分为两个阵营：System V和BSD。商业版本的Unix一般属于System V阵营，通常是不开源的，如IBM的AIX系统，惠普的HP-UX系统等；而BSD阵营的版本通常都是开源的，如FreeBSD、OpenBSD等。

4.1.2 编程语言：Android的编程语言是Java和KotLin；而ios的则为ObjectC和Swift

Android的Java，面向对象，性能比C语言和OC低；ios的OC，基于对象，完全兼容C语言的语法，可以直接操作内存。Android生成 .class 文件，需要虚拟机来进行解释；iOS 直接执行程序的二进制代码。

这也在根本上造成了iOS与Android性能不同：Android和Window一样，目的是打造一款通用性非常好的系统，在任何机器上面都可以运行；ios目的是让软件和硬件完美的结合到一块，该操作系统只能在极少数机器上面才能运行。

4.1.3 运行机制：ios采用的是沙盒运行机制；安卓采用的是虚拟机运行机制。

iOS采用伪后台，当用户HOME键退出应用时，IOS其实关闭了程序，只保留应用的图像入口，只会默认将最后的运行数据记录在RAM中。之所以IOS也能收到推送，是因为应用程序开启推送后，系统会增加一些进程，这些进程会从苹果服务器接收信息，然后再通过服务器发给用户，苹果服务器在这里是起到了中转的作用。**安卓手机的后台是真后台**，将应用保留在RAM中，之所以能够收到推送，也因为它常驻内存。所以Android在软件关闭的情况下，无法接收推送信息；ios在软件关闭的情况下，依然可以接收推送信息，iOS系统在系统内存不足时会自动释放内存。

> iOS：沙盒运行机制
>
> - 每个程序都有自己的虚拟地址空间。所以，程序之间是都不能进行访问
> - 默认只会将应用的最后运行数据，记录在RAM里面
>
> Android：虚拟机运行机制
>
> - 所有的应用程序都是运行在虚拟机中，用户界面其实是由虚拟机传递的，并且通过虚拟机，Android的任何程序都就可以轻松访问其他程序文件
> - 所有的Android的应用程序都是运行在RAM里面的，所以会发现有时候Android用着用着就开始有点卡了

4.2 渲染机制

4.2.1 iOS最先响应屏幕

IOS的UI渲染采用实时优先级，Android的UI渲染遵循传统电脑模式的主线程普通优先级。

> iOS的响应顺序依次为Touch－Media－Service－Core架构
> Android系统的优先级响应层级是Application－Framework－Library－Kernal架构

当我们使用iOS或者是Android手机时，第一步就是滑屏解锁找到相应程序点击进入。而这个时候往往是所有操控开始的第一步骤，iOS系统产品就表现出来了流畅的一面，但Android产品却给人一种卡顿的现象，更别说后续深入玩游戏或者进行其它操控了。

这也就是为什么我们常说iOS比Android流畅了，因为iOS最先响应屏幕，iOS对屏幕反应的优先级是最高的，它的响应顺序依次为Touch–Media–Service–Core架构，换句话说当用户只要触摸接触了屏幕之后，系统就会最优先去处理屏幕显示也就是Touch这个层级，然后才是媒体(Media)，服

务(Service)以及Core架构。

而Android系统的优先级响应层级则是Application–Framework–Library–Kernal架构，和显示相关的图形图像处理这一部分属于Library，你可以看到到第三位才是它，当你触摸屏幕之后Android系统首先会激活应用，框架然后才是屏幕最后是核心架构。

4.2.2 iOS图形特效基于GPU加速渲染

iOS 系统对图形的各种特效处理基本上都是**基于GPU硬件进行加速的**，它可以不用完全借助CPU或者程序本身，而是通过GPU进行渲染以达到更流畅的操控表现。但是Android系统产品则并非如此，因为Android需要适应不同的手机硬件，需要满足各种差异配置，所以很多图形特效大多都要靠程序本身进行加速和渲染(虽然Android 4.0以及4.1等更高版本中进行了改进将硬件加速设为默认开启，但依旧无法做到所有特效全部都靠GPU进行加速。在很多Android手机里面都自带有“是否开启GPU渲染”这个功能选项，不过开启之后的改善也是微乎其微) 并严重依赖CPU运算的操作自然会加大处理器的负荷，从而出现卡顿的问题。