#### **IOS VS Android**

本次分享无任何恶意,请android同学....

#### 问题

- 为什么IOS比android使用更少的内存?
- 为什么IOS比android更流畅?
- 为什么IOS比android更安全?

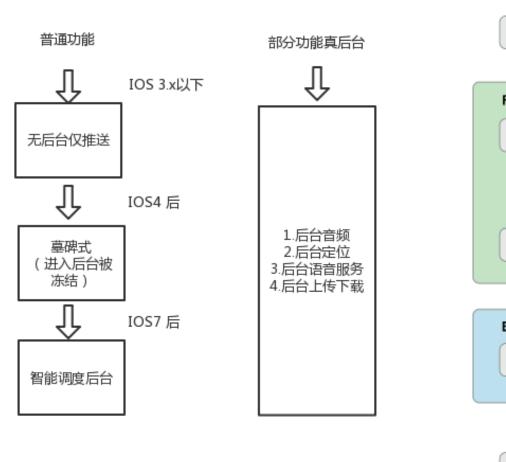
#### 目录

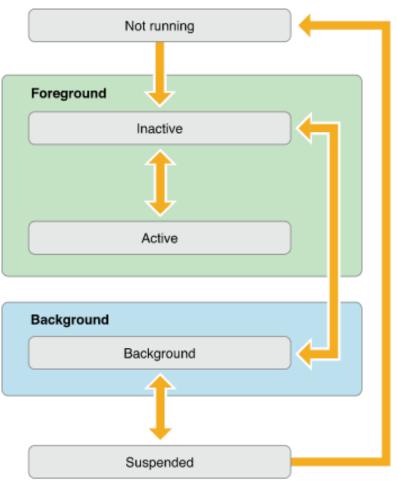
- · IOS的智能后台模式
- IOS的内存管理模式
- IOS APNs
- IOS的沙盒原理
- IOS的签名原理

# IOS的后台模式

- Android真后台,允许app留存在内存,依靠 service服务组件继续运行
- IOS伪后台(智能后台)

#### IOS后台趋势历程



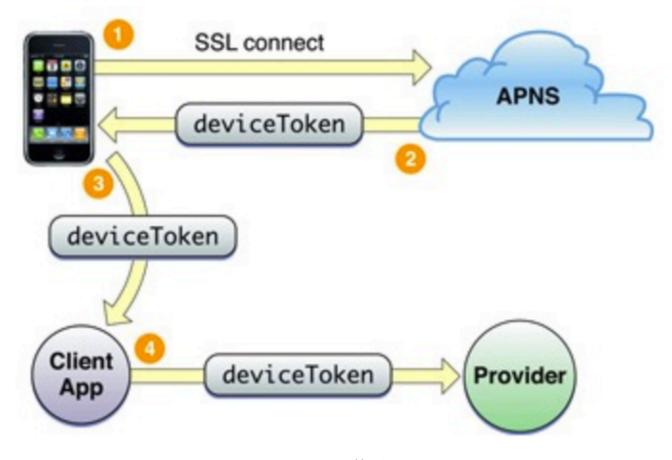


#### 思考

· los为什么把后台模式搞得如此麻烦?

### 伪后台如何获取实时消息

APNs(App Push Notification Service)



Token获取

dag Attributes

friendlyName: Apple Development IOS Push Services: <a href="com.wuba.banghuangye">com.wuba.banghuangye</a> tocalKeyID: 72 B7 5B E9 B1 EE 4C 48 DB 91 F2 EE 47 5A 75 EB E6 30 43 29 subject=/UID=com.wuba.banghuangye/(N=Apple Development IOS Push Services: <a href="com.wuba.banghuangye/Ou=EFH63YDHFL/C=US">com.wuba.banghuangye/Ou=EFH63YDHFL/C=US</a>

issuer=/C=U5/O=Apple Inc./OU=Apple Worldwide Developer Relations/CN=Apple Worldwide Developer Relations Certification Authority
-----BEGIN CERTIFICATE-----

IIIFjTCCBHWgAwIBAgIILXFQaYFqx6YwDQYJKoZIhvcNAQEFBQAwgZYxCzAJBgNV 3AYTAlvTMRMwEQYDVQQKDApBcHBsZSBJbmMuMSwwKgYDVQQLDCNBcHBsZSBXb3Js ZHdpZGUgRGV2ZWxvcGVyIFJlbGF0aW9uczFEMEIGA1UEAww7QXBwbGUgV29ybGR3 aWRlIERldmVsb3BlciBSZWxhdGlvbnMgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkw HhcNMTcxMTE2MDY1MDQ4WhcNMTgxMTE2MDY1MDQ4WjCBjDEkMCIGCgmSJomT8ixk AQEMFGNvbS53dWJhLmJhbmdodWFuZ3llMUIwQAYDVQQDDDlBcHBsZSBEZXZlbG9w DWYUdCBJT1MgUHVzaCBTZXJ2aWN\czogY29tLnd1YmEuYmFuZ2h1YW5neWUxEzAR 3gNVBASMCkVGSDYzWURIRkwxCzAJBgNVBAYTA\VTMIIBIjANBgkghkiG9w0BAQEF AAOCAQ8AMIIBCgKCAQEAvysNdXHaE1l9iUe1M0Yl3KNT3XN2gjgF47BXM+UGKaGd 359Rhd0FXT0CqneatmWUyrLkJf9KTqA0000SFD8Ie0sr8yFHEnW1T+7taxU8tqNJ m8vQer2hA2Y7Ez/Qllfsz0uaHoyzdHq3bREF/getzEpSv3J2zNceRlWwdyjK00xV nBAZ/Dh9/IJWPcZHpGpPkRlWGNgInmamqukLq09a77Vf3ks0UFkLq8eRR4Qofe1V bCSbW7B+C2ZzYROa7wXM0X/N+F3gXUjnoOnU0BTgr07wvYPiojnMyJuv8JGu/E8e CvREaiCAsRlmXY4rzd9KBu6jswDpoBxkcGvKRn/AVwIDAQABo4IB5TCCAeEwHQYD √R00BBYEFHK3W+mB7kxI25Hy7kdadevmMEMpMAkGA1UdEwQCMAAwHwYDVR0jBBgw FoAUiCcXCam2GGCL70u69kdZxVJUo7cwggEPBgNVHSAEggEGMIIBAjCB/wYJKoZI nvdiZAUBMIHxMIHDBggrBgEFBQcCAjCBTayBsIJlbGlhbmNliG9uIHRoaXMgY2Vy IGlmaWhHGUgYTMSYHBHGMTSIFERSIFERSITZXMgYWBIZXBGYBIZSBGBGU IGhlbiBhcHBsaWNNYmXlHNcWSYKXXLKHRLcmIZIGFuZCBibZKbgXARbZSZIG9m IHVzZSwgY2VydGlmaWNhdGUgcG9saWN5IGFuZCBjZXJ0aWZpY2F0aW9uIHByYWN0 aWNlIHN0YXRlbWVudHMuMCkGCCsGAQUFBwIBFh1odHRw0i8vd3d3LmFwcGxlLmNv pS9hcHBsZWNhLzBNBgNVHR8ERjBEMEKgQKA+hjxodHRw0i8vZGV2ZWxvcGVyLmFw cGxlLmNvbS9jZXJ0aWZpY2F0aW9uYXV0aG9yaXR5L3d3ZHJjYS5jcmwwCwYDVR0F BAQDAgeAMBMGA1UdJQQMMAoGCCsGAQUFBwMCMBAGCiqGSIb3Y2QGAwEEAgUAMA0G SqGSIb3DQEBBQUAA4IBAQBNK3EODcKsFm23CNJCXhsZRf82MmYuANtywlftRNWj ld1oVDK/SQXf5gTdcTZ55X5DW/XuuSzPw6ZPZghDcVwcLytSc2mIMVt3tXw/3P6e \$5gx6z7mn@ADHrA5FpojUXOLQOmfTPimNbpZkIjqT9d8sLNn37Q1DX19h8u8ati6 \$s0gFYlp9ZgfsFwLc/zxP@TL70fWuBTEYUhujWkOLgCp7ogSmYJVIVhigFnT@wgo P4UlQqcl0x/sCEa763kUNpwBfGXoMHaVqXVXu9YlU7MjRT/2ll63t1m4Fmk+5g9f SIdsel4E1zq/8GLV/ufcan5mvcs0LU8/XtHLasljFecz

----END CERTIFICATE-------BEGIN RSA PRIVATE KEY----

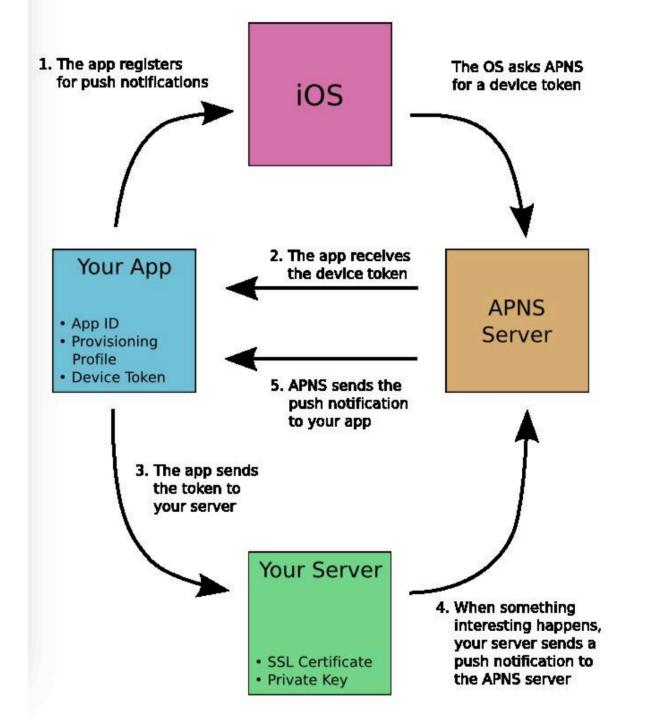
!!!EpAIBAAKCAQEAvvsNdXHaE119iUe1M0Yl3KNT3XN2gjgF47BXM+UGKaGd859R nd0FXT0CqneatmWUyrLkJf9KTqA0000SFD8Ie0sr8yFHEnW1T+7taxU8tqNJm8yQ er2hA2Y7Ez/Qllfsz0uaHoyzdHg3bREF/getzEpSv3J2zNceRlWwdyjK00xVhBAZ /Dh9/IJWPcZHpGpPkRlWGNgInmamqukLq09a77Vf3ks0UFkLq8eRR4Qofe1VbCSb w/7B+C2ZzYROa7wXM0X/N+F3gXUjnoOnUOBTgr07wvYPiojnMyJuv8JGu/E8eCvRE aiCAsRlmXY4rzd9KBu6jswDpoBxkcGvKRn/AVwIDAQABAoIBAFnRbIT9pJhREC90 IDL9KnVYAws1UFjxzCNHPdvULE/MCKMZHJgeWHV/Sg+t7H83ywCBnbcZF3MTp5yy hix/YJfLxM61tt1BqqILNnGULVef9/BkBtcS8/TLUaE98/WhKGhdoAAaIBP4reiH hw/ywJdBpSUQyjqQ0Vb02g1Q17q863NeK8w8zrWW1ysQWWCu1+P/cLrcQa4F6eoC 3W5vckHytCViULjug4N5TtmgvPpMRrDe/OpYPAXhjdXKtAIk64WShE7007IE6sYv 31PzL06Ia7hXAsyvZKxi0VJ8/PhnHlHvbWuaeiFUG8HSl4r05Hfyl2dcmKRt6b20 eFipcLECgYEA5HGG1vdheeQBnsV2PN+LvDRsTLeu0ogWx8U+Pp4b6xKgHT4sYUwc mJ7iUb3acifwJIYm0I+QvimheufmXhoco/pF2gcQE5pR36kY50AT3GwRKoAzKaGT 77A22GYQdQE5PWFnur1y4DS5iJgOiLjBbEZWCCxNKcxapvxpilJDTUCgYEA1gSm 79d4BgT0bX/xYtWtruWcNACOjbonmttlEgkkEEC1/hGsGvlx6w1SEUiVG+PbhyLF I0fNxMgGbByL5ETIFrrnR0U0QzdXEoHXwc3OeiPvGaO8X1YMDAe3VXx3beBxxNxf r1J7kyc4WwFeVq+uz+5zcTJuAZv9DD0cfCV0JNsCqYEA1iYGfWYSHq27Rxdcq9vz

#### 推送流程

Bag Attributes

friendlyName: Apple Development IOS Push Services: com.wuba.banghuangye localKeyID: 72 B7 5B E9 81 EE 4C 48 DB 91 F2 EE 47 5A 75 EB E6 30 43 29 subject=/UID=com.wuba.banghuangye/CN=Apple Development IOS Push Services: com.wuba.banghuangye/OU=EFH63YDHFL/C=US issuer=/C=US/O=Apple Inc./OU=Apple Worldwide Developer Relations/CN=Apple Worldwide Developer Relations Certification Authority -----BEGIN CERTIFICATE-----

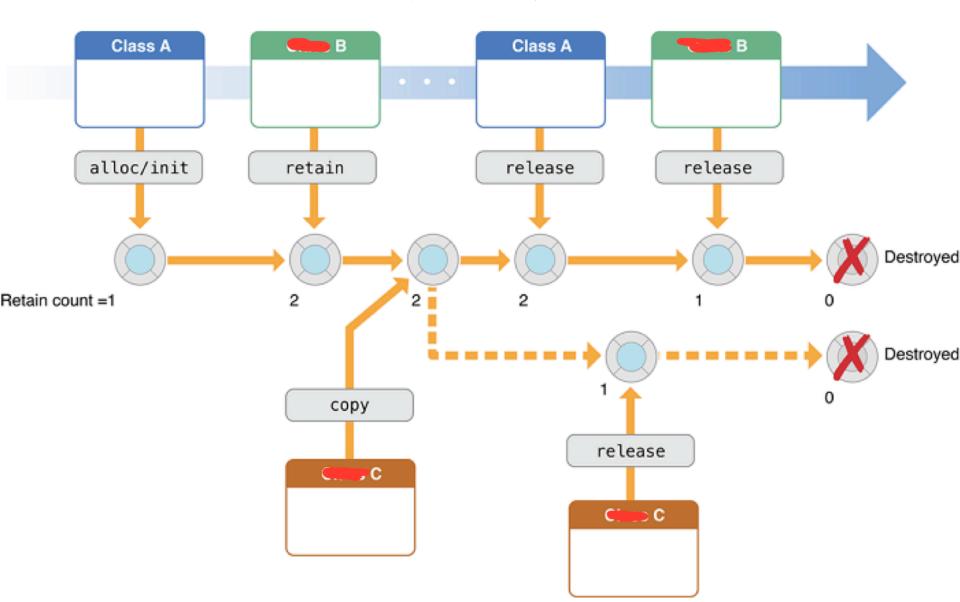
Provider notification notification notification notification notification notification



# IOS内存管理模式

- MRC
- ARC及实现原理

# IOS内存管理

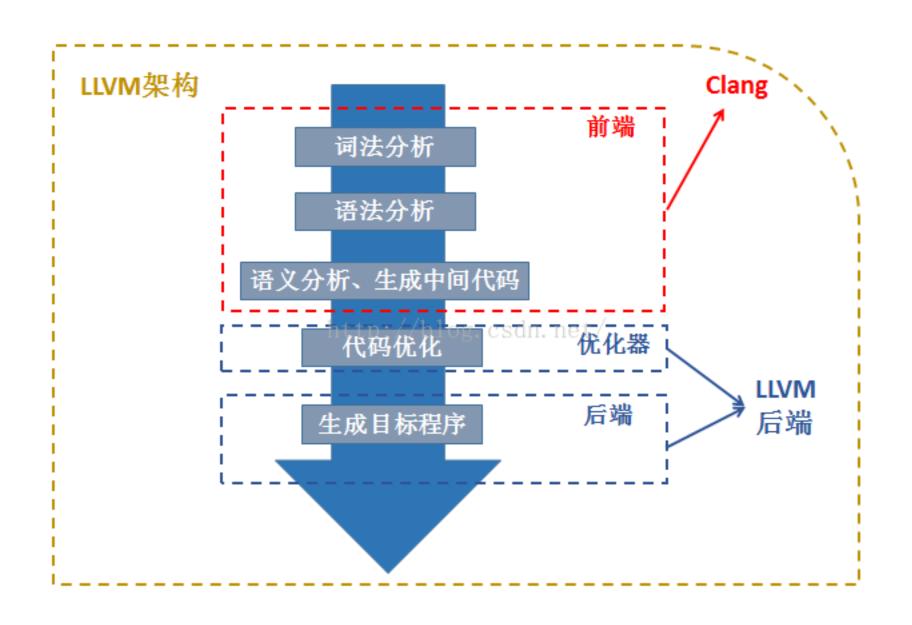


#### MRC(Manual Reference Counting)

```
int main(int argc, const char * argv[]) {
   @autoreleasepool {
       // 只要创建一个对象默认引用计数器的值就是1
       Person *p = [[Person alloc] init];
       NSLog(@"retainCount = %lu", [p retainCount]); // 1
       // 只要给对象发送一个retain消息,对象的引用计数器就会+1
       [p retain];
      NSLog(@"retainCount = %lu", [p retainCount]); // 2
       // 通过指针变量p,给p指向的对象发送一条release消息
       // 只要对象接收到release消息,引用计数器就会-1
       // 只要一个对象的引用计数器为0,系统就会释放对象
       [p release];
       // 需要注意的是: release并不代表销毁\回收对象, 仅仅是计数器-1
       NSLog(@"retainCount = %lu", [p retainCount]); // 1
       [p release]; // 0
      NSLog(@"----");
     [p setAge: 20]; // 此时对象已经被释放
   return 0;
```

#### **ARC**

```
int main(int argc, const char * argv[]) {
    // 不用写release, main函数执行完毕后p会被自动释放
    Person *p = [[Person alloc] init];
    return 0;
}
```



## ARC实现原理

• Ilvm(Low Level Virtual Machine)底层虚拟机

它是一个编译器的基础建设,是为了任意一种编程语言写成的程序,利用虚拟技术,创造出编译时期,链结时期,运行时期以及"闲置时期"的优化

Xcode3之前,用的是GCC Xcode3,GCC仍然保留,但是也推出了LLVM,苹果推荐

LLVM-GCC混合编译器,但还不是默认编译器

Xcode4,LLVM-GCC成为默认编译器,但GCC仍保留

Xcode4.2,LLVM3.0成为默认编译器,纯用GCC不复可能

Xcode4.6,LLVM升级到4.2版本

Xcode5,LLVM-GCC被遗弃,新的编译器是LLVM5.0,从GCC 过渡到LLVM的时代正式完成

#### 编译器代码优化帮我们加入了内存释放

```
int main()
                                                clang -S -fobjc-arc -emit-llvm main.m -o main.ll
                             void objc_storeStrong(id *object, id value) {
        id a;
                                 id oldValue = *object;
                                 value = [value retain];
; ModuleID = 'main.m'
source_filename = "main.m"
                                 *object = value;
target datalayout = "e-m:o-i64:64-
target triple = "x86 64-apple-maco
                                 [oldValue release];
; Function Attrs: nounwind ssp uwtl
define i32 @main() #0 {
 %1 = alloca i8*, align 8
 store i8* null, i8** %1, align 8
 call void @objc_storeStrong(i8** %1, i8* null) #1
 ret 132 v
declare void @objc_storeStrong(i8**, i8*)
attributes #0 = { nounwind ssp uwtable "correctly-rounded-divide-sqrt-fp-math"="
false" "disable-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-point
er-elim"="true" "no-frame-pointer-elim-non-leaf" "no-infs-fp-math"="false" "no-j
ump-tables"="false" "no-nans-fp-math"="false" "no-signed-zeros-fp-math"="false"
"stack-protector-buffer-size"="8" "target-cpu"="penryn" "target-features"="+cx16"
,+fxsr,+mmx,+sse,+sse2,+sse3,+sse4.1,+ssse3,+x87" "unsafe-fp-math"="false" "use-
soft-float"="false" }
attributes #1 = { nounwind }
```

"main.ll" 28L, 1392C

### IOS硬件

- 《一只IPhone的全球之旅》中介绍,iphone用了专用的芯片处理触摸屏的信息处理和手势识别,硬件的差异估计才是比较根本的。用cpu处理ui显然会卡。"著名电子设备调查机构 iSuppli就曾经指出,Apple 打破业内常规的把最大部分成本花费在了改善用户体验上,因此,我们看 iPhone 的硬件架构就可以发现,为了处理一个小小的触控屏,apple动用了三块芯片,一个Broadcom 的模拟信号处理器,用来处理触控屏传感器传来的模拟信号,转换为一组代表 x、y位置信息的数据流,一个飞利浦(NXP)ARM7CPU,用来作为手势算法处理器,把触控指令解析出来(后期 iPhone型号把这两个芯片整合在一起成为一个双核结构的处理器,称为改进型 Broadcom芯片),而主处理器则有一个高级别的优先独立线程专门处理触控操作类指令。
- 相对于其他电容触控手机大多把除了模数转换之外的大部分触控控制任务都交给主 CPU 的做法, iPhone 能够有"一触即发"的操作快感就不难理解了。

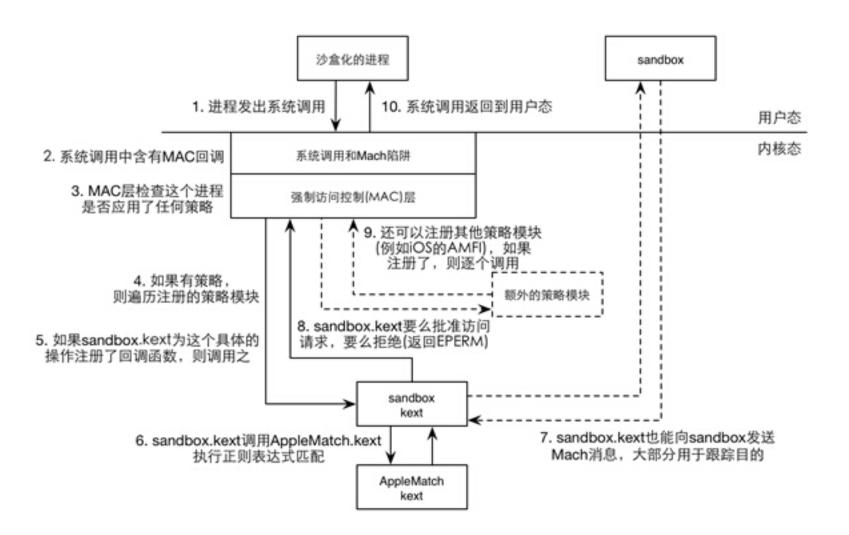
### 对比优缺点

- 优点:除了语言上的问题,伪后台,不同与android的垃圾回收机制,对用户体验的高度投入,使得ios需要较少的内存,较小的电池,后台占用cpu少,使得app更流畅
- 缺点: 伪后台

### (监狱)沙盒模式

- UID/GID permission
- iOS Mandatory Access Control

#### MAC

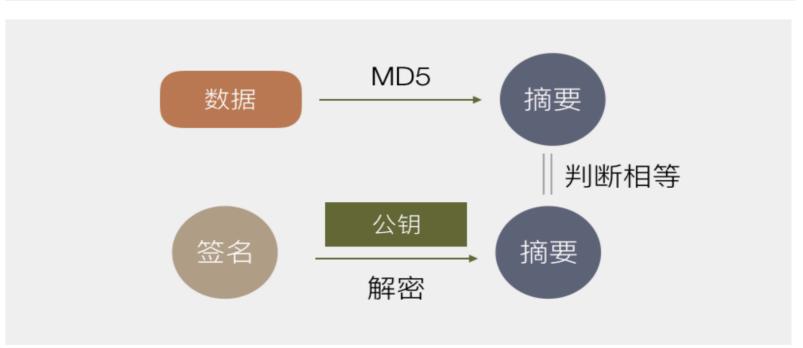


#### Kext(kernel extension)

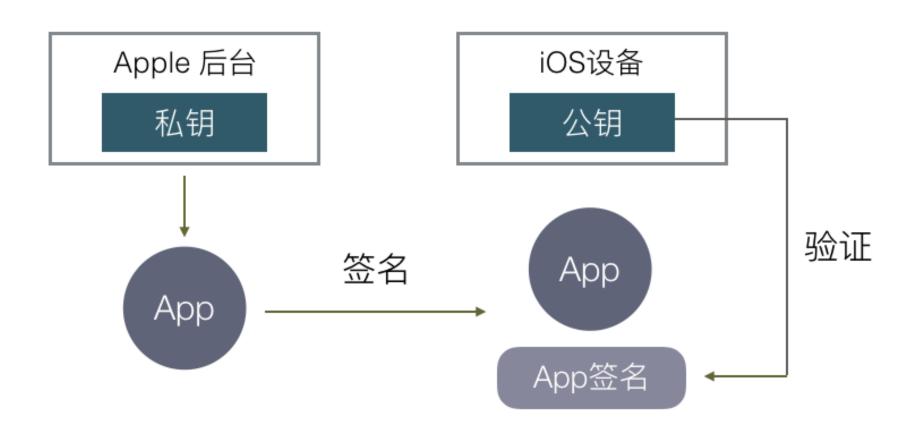
文件/子目录	内 容
CodeDirectory	kext 的代码目录文件
CodeRequirements	kext 的代码需求设置
CodeResources	代码资源 XML 文件,包含 kext 中文件的散列和规则
CodeSignature	kext 的代码签名——通常包含苹果的数字证书
Info.plist	bundle 清单属性列表
MacOS	这个目录下包含了实际的 kext 二进制代码,二进制代码文件的类型为 BUNDLE(Mach-O 类
	型 8)或 KEXTBUNDLE(用于 64 位, Mach-O 类型 11)
_CodeSignature	包含 Code*文件的目录,这些文件实际上是上述 Code*文件的符号连接
version.plist	表示 kext 版本信息的属性列表

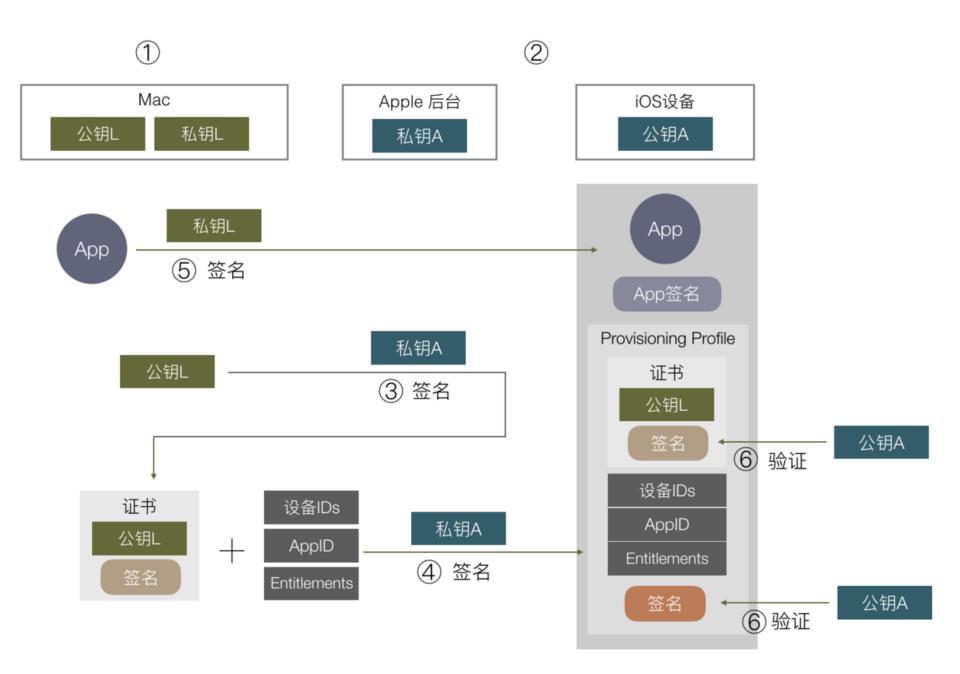
## 签名原理

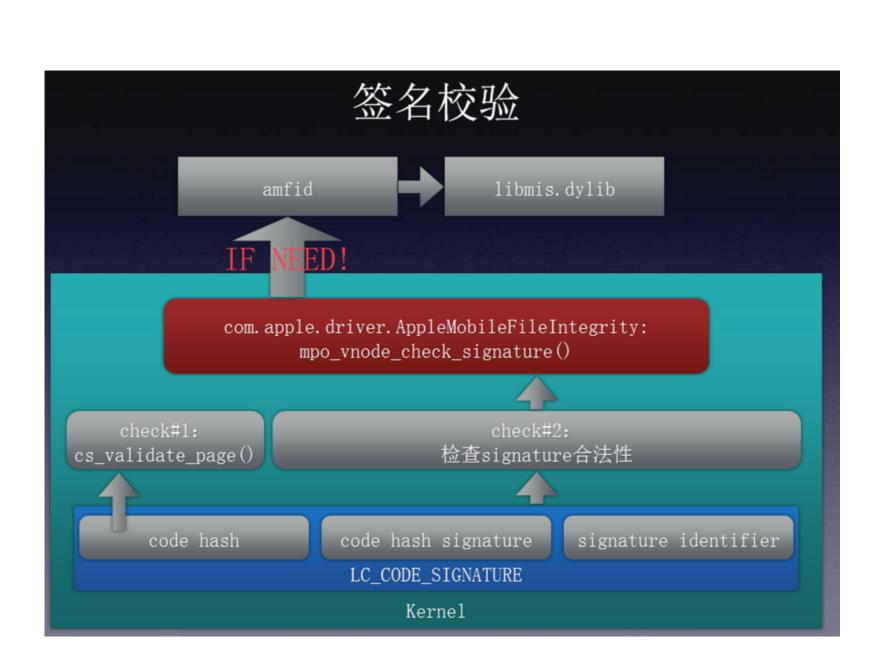




## 苹果商店签名







## 总结

• 为什么快,内存使用少,耗电少?

伪后台(智能后台),APNs(推送),代码中释放内存,没有系统垃圾回收模块。使得后台程序消耗很少的cpu和内存,硬件上对用户触摸做了较大的投入。

为什么安全?

沙盒使得app只能在自己的空间中处理,并限制了api方法调用,签名模式,使得只有从appstroe下载的app(企业发布,需要手动认证证书),才能在硬件上运行。