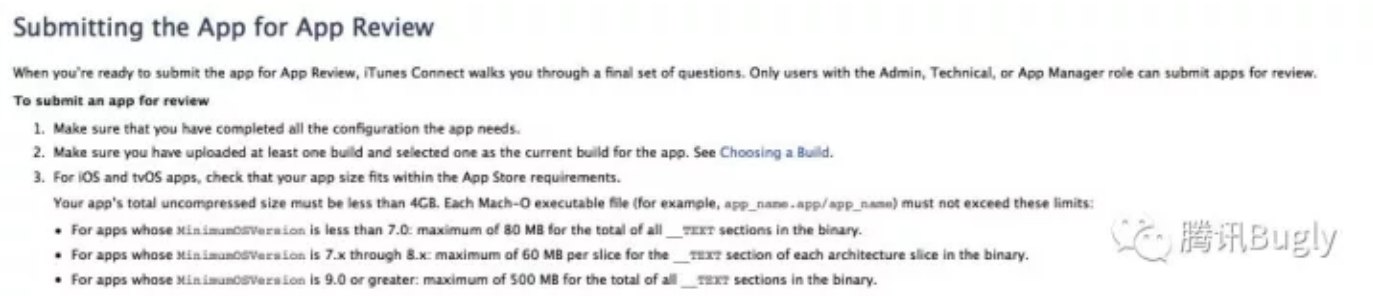
不给“爸爸”添麻烦 - iTOP iOS 动态库改造

原创 2017-12-14 hillsonsong 腾讯Bugly

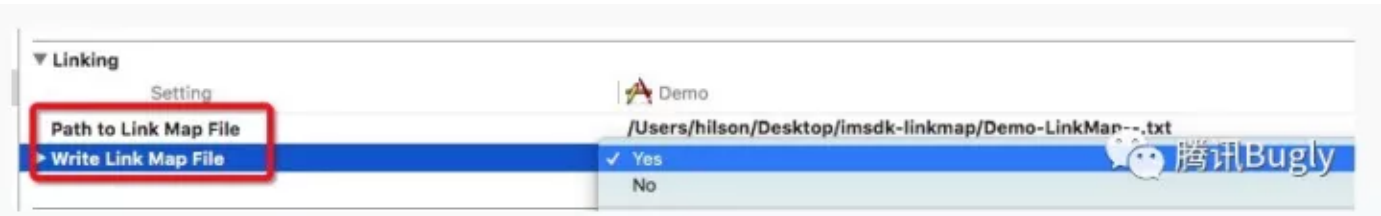
一、背景

苹果官方文档 对提交商店 APP 的二进制文件中__**TEXT**段大小有限制，超过大小限制的应用在提交评审的时候会被拒绝。目前**Ngame**在合入海外潘多拉 SDK 的过程中，发现二进制 __**TEXT** 段大小超过限制，因此需要对应用进行瘦身。



二、Link Map File

- Link Map 文件是 Xcode 产生可执行文件的同时生成的链接信息，用来描述可执行文件的构造成分，包括代码段(__**TEXT**）和数据段（__**DATA**）的分布情况。
- Link Map 里展示了整个可执行文件的全貌，列出了编译后的每一个.o目标文件的信息（包括静态链接库.a里的），以及每一个目标文件的代码段，数据段存储详情。
- 设置 **Project->Build Settings->Write Link Map File** 为**YES**，并设置 Path to Link Map File，build 完后就可以在设置的路径看到 Link Map 文件了。



通过 Link Map File 精确统计出各个组件 __**TEXT** 段大小，因此有必要对 Link Map File 稍做研究。每个 Link Map 文件由3个部分组成：

1、Object files：目标文件列表

```
SDKDemo/IMSDKCoreKit.framework/IMSDKCoreKit(IMSDKUtils.o)
SDKDemo/IMSDKCoreKit.framework/IMSDKCoreKit(IMSDKAFSecurityPolicy.o)
SDKDemo/IMSDKCoreKit.framework/IMSDKCoreKit(IMSDKPayExtendManager.o)
SDKDemo/IMSDKCoreKit.framework/IMSDKCoreKit(IMSDKAccountForOC.o)
SDKDemo/IMSDKCoreKit.framework/IMSDKCoreKit(IMSDKLocationManager.o)
```

前面中括号表示该文件的编号，**IMSDKCoreKit** 静态库中所有目标文件都会列出来（包括私有文件）。

2、Sections：段表，描述各个段在最后编译成的可执行文件中的偏移位置和大小，包括了代码段（**__TEXT，保存程序代码段编译后的机器码）和数据段（__DATA**，保存变量值）

```
# Sections:
# Address    Size        Segment    Section
0x1000046B4    0x0007B238    __TEXT    __text
0x10007F8EC    0x00000BDC    __TEXT    __stubs
0x1000804C8    0x00000BDC    __TEXT    __stub_helper
0x1000810A4    0x0000C2CA    __TEXT    __objc_methname
0x10008D36E    0x00000ACE    __TEXT    __objc_classname
0x10008DE3C    0x0000AFAF    __TEXT    __objc_methtype
0x100098DEC    0x0000EE48    __TEXT    __gcc_except_tab
0x1000A7C34    0x00009E1C    __TEXT    __cfstring
0x1000B1A50    0x00000180    __TEXT    __const
0x1000B1BD0    0x00000242    __TEXT    __ustring
0x1000B1E14    0x000021B0    __TEXT    __unwind_info
0x1000B3FC8    0x00000034    __TEXT    __eh_frame
0x1000B4000    0x00000258    __DATA    __got
0x1000B4258    0x000007E8    __DATA    __la_symbol_ptr
0x1000B4A40    0x00003520    __DATA    __const
0x1000B7F60    0x000080C0    __DATA    __cfstring
0x1000C0020    0x000002A0    __DATA    __objc_classlist
0x1000C02C0    0x00000008    __DATA    __objc_nlclslist
0x1000C02C8    0x00000060    __DATA    __objc_catlist
0x1000C0328    0x00000010    __DATA    __objc_nlcatlist
0x1000C0338    0x00000108    __DATA    __objc_protolist
0x1000C0440    0x00000008    __DATA    __objc_imageinfo
0x1000C0448    0x00015CA0    __DATA    __objc_const
0x1000D60E8    0x000031F0    __DATA    __objc_selrefs
0x1000D92D8    0x00000050    __DATA    __objc_protorefs
0x1000D9328    0x00000498    __DATA    __objc_classrefs
0x1000D97C0    0x000001F8    __DATA    __objc_superrefs
0x1000D99B8    0x000006AC    __DATA    __objc_ivar
0x1000DA068    0x00001A40    __DATA    __objc_data
0x1000DBAA8    0x00000CB0    __DATA    __data
0x1000DC758    0x00000278    __DATA    __bss
0x1000DC9D0    0x00000038    __DATA    __common
```

首列是数据在文件的偏移位置，第二列是这一段占用大小，第三列是段类型，代码段和数据段，第四列是段名称。

__text表示编译后的程序执行语句，__data表示已初始化的全局变量和局部静态变量，__bss表示未初始化的全局变量和局部静态变量，__cstring表示代码里的字符串常量。

3、Symbols：详细描述按每个文件列出每个对应字段的位置和占用空间

```
# Symbols:
# Address    Size        File  Name
0x1000046B4    0x00000050    [  1] -[ViewController viewDidLoad]
0x100004704    0x00000050    [  1] -[ViewController didReceiveMemory
0x100004754    0x000000A4    [  2] _main
0x1000047F8    0x0000013C    [  3] -[AppDelegate application:didFini
0x100004934    0x0000004C    [  3] -[AppDelegate applicationWillResi
0x100004980    0x0000004C    [  3] -[AppDelegate applicationDidEnter
0x1000049CC    0x0000004C    [  3] -[AppDelegate applicationWillEnte
0x100004A18    0x0000004C    [  3] -[AppDelegate applicationDidBecom
0x100004A64    0x0000004C    [  3] -[AppDelegate applicationWillTerm
0x100004AB0    0x0000002C    [  3] -[AppDelegate window]
0x100004ADC    0x0000004C    [  3] -[AppDelegate setWindow:]
0x100004B28    0x00000044    [  3] -[AppDelegate .cxx_destruct]
0x100004B6C    0x00000244    [  4] +[IMSDKUtils getAppPlistInfo]
```

同样首列是数据在文件的偏移地址，第二列是占用大小，第三列是所属文件序号，对应上述Object files列表，最后是名字。

三、代码段大小统计

1.如上文提到的，每个文件都有一个固定的标号，如 IMSDKUtils 这个目标文件的标号是**4** `[4]/Users/hilson/Desktop/IMSDKDemo/IMSDKDemo/IMSDKCoreKit.framework/IMSDKCoreKit(IMSDKUtils.o)`；

2.遍历 Link Map 文件第三部分 **Symbols** 里的每一行，将文件编号（第三列）为[4] 的数据都取出，将每一行的 size（第二列）相加，就是这个目标文件的大小；

3.静态库中全部目标文件的大小相加，就是这个第三方库的占用空间大小；
目前业界有数款统计小工具，如 nodejs(https://gist.github.com/bang590/8f3e9704f1c2661836cd) 脚本 ，Link Map解析工具-MAC。

四、优化思路

虽然 IMSDK 代码段已经足够小，对项目影响较小，本着负责到底实事求是的态度，还是有必要对静态库大小瘦身做进一步研究。

1、 代码级别优化

- 查找无用selector，以往C++在链接时，没有被用到的类和方法是不会编进可执行文件里。但 Objctive-C 不同，由于它的动态性，它可以通过类名和方法名获取这个类和方法进行调用，所以编译器会把项目里所有OC源文件编进可执行文件里，哪怕该类和方法没有被使用到。
- 查找无用oc类，与查找无用 selector 情况类似
- 扫码重复代码，可以利用第三方工具simian扫描

整体而言，代码层面优化见效甚微。与此同时，删除重复代码会导致代码重构， 极有可能影响代码的稳定性，而且由于 Objctive-C 的 runtime 机制，我们无法从 Link Map 文件中确认某个类和 selector 究竟有没在某个特殊情况下通过反射机制调用到。基于稳定性考虑，IMSDK 暂时不采用**代码级别优化**。

2、编译器优化级别

- Build Settings->Optimization Level有几个编译优化选项，release版应该选择 **Fastest, Smalllest**，这个选项会开启那些不增加代码大小的全部优化，并让可执行文件尽可能小。

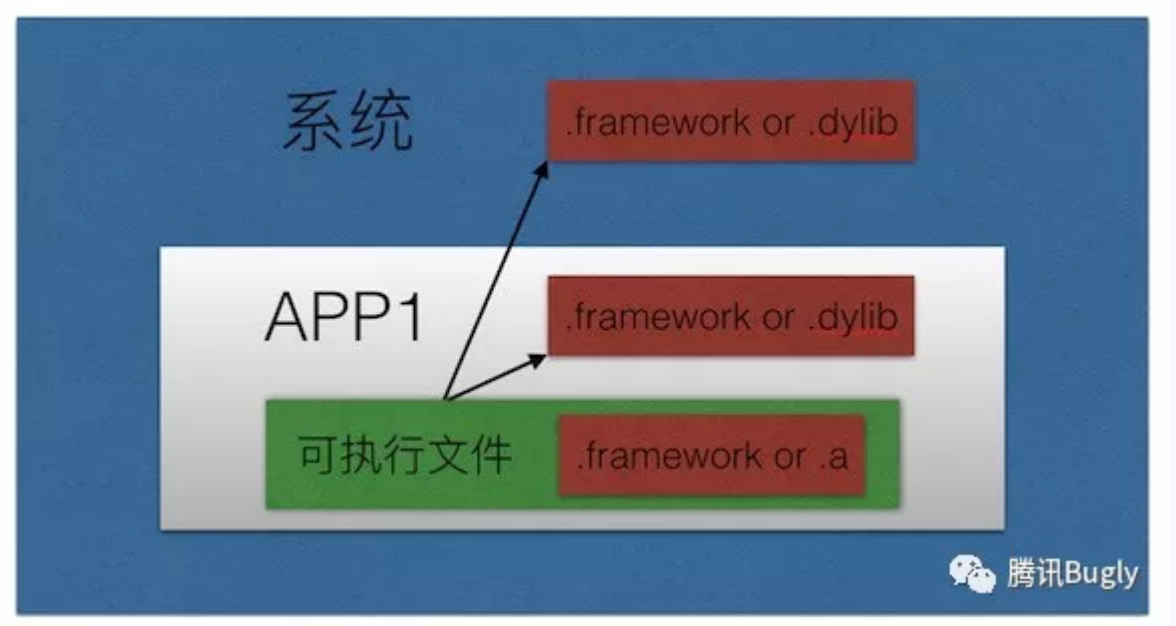
- 实测，开启 Fastest, Smalllest 选项后， **__TEXT** 段大小能裁剪 **25%**。比如，IMSDKWebView **__TEXT** 段大小从 249.93K 缩减到179.93K，IMSDKHelp __TEXT 段大小从 458.77K 缩减到 336.66K；

- iMSDK 组件发布版本默认已经选择 **Fastest, Smalllest** 模式，因此此方案对 iMSDK 组件暂无提升，**但可以提供思路给其他 SDK 做瘦身。**

3、静态库改用动态库

- 从 iOS 8 开始，由于 Extendsion 的出现，苹果开始允许自建动态库并在 iOS APP 中引用，这样宿主 APP 和插件之间共享动态库；
- 从目前来看，iOS 仍然不允许进程间共享动态库，即 iOS 上的动态库只能是私有的，因为我们仍然不能将动态库文件放置在除了自身沙盒以外的其它任何地方；
- 另外，苹果沙盒会验证动态库的签名，所以如果是动态从服务器更新的动态库，是签名不了的，因此应用插件化、软件版本实时模块升级等功能在 iOS 上无法实现；

- 由于动态库在应用编译打包的时候，仅把链接信息编译到应用二进制可执行文件中，将 framework 的加载推迟到运行时，因此，应用在提交评审时的代码段大小计算，是不会将动态库的代码段计算计算在内，从而能够节省出一大截代码段大小空间。



五、改动汇总

1.IMSDK 统一修改打包脚本，将 framework 的 MACH_O_TYPE 从 staticlib 改成 mh_dylib ，并且 将 GCC_SYMBOLS_PRIVATE_EXTERN 赋值为 NO (在 Build settings 将 “Symbols Hidden By Default“ 为 NO)；

2.IMSDKCoreKit（动态库） 使用了 libmtasdk.a（静态库），并且 libmtasdk.a 自建系统类的类别 category，通过 nm 命令可以看到动态库并不会把静态库中所有的 Objective-C 类和类都加载到最后的可执行文件中，会导致运行 crash。因此，需要在 IMSDKCoreKit target 的 Other linker flags 新增 -ObjC 选项。

3.项目工程 iOS 最低系统版本支持从 iOS 7 提升到 iOS 8（根据腾讯移动分析-数据中心数据(https://mta.qq.com/mta/data/device/os)的统计，iOS 7用户群里已经基本忽略不计）

4.项目工程 将 iMSDK 各个组件从 Garenal->Linked Frameworks and Libraries 添加到 Garenal->Embedded Binaries（When should we use “embedded binaries” rather than “Linked Frameworks” in Xcode?(https://stackoverflow.com/questions/32675272/when-should-we-use-embedded-binaries-rather-than-linked-frameworks-in-xcode)）

六、 效果

根据 IMSDK Demo 测试结果，静态库改用动态库后，结论如下

1、IMSDKCoreKit 动态库能和静态库的插件混用，业务可以根据情况自由选择动态库更新；

2、安装包大小会对应增加，因为动态库 SDK 没有编译到应用可执行二进制文件里，而是类似资源的形式以一个单独 framework 文件存在安装包中，导致安装包大小压缩有限。

3、IPA 可执行二进制文件体积大大减少，动态库的代码段信息只有符号链接信息，大小基本可以忽略不计。

七、常见问题

1.打包的时候出现 Failed to verify bitcode in xxxxx

解决方法：在 build setting 中关闭 Enable Bitcode 配置项

2.Found an unexpected Math-O Header

解决方法：将 Embedded Binaries 中的静态库移到 Linked Frameworks and Libraries 中

八、参考链接

https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/DynamicLibraries/000-Introduction/Introduction.html#//apple_ref/doc/uid/TP40001908-SW1

https://stackoverflow.com/questions/30173529/what-are-embedded-binaries-in-xcode

https://stackoverflow.com/questions/32675272/when-should-we-use-embedded-binaries-rather-than-linked-frameworks-in-xcode

https://mta.qq.com/mta/data/device/os

https://github.com/huanxsd/LinkMap

http://www.cocoachina.com/ios/20151211/14562.html
