第18节课内容总结

App启动分为俩个阶段

- pre-main，main()函数之前，系统加载可执行文件，加载动态链接库dyld，dyld递归加载所有依赖的动态库，然后dyld调起main()函数。

  pre-main分为四个阶段：

  i. dylib loading：加载动态库。可以通过减少动态库的数量来优化这一部分所消耗的时间。苹果的建议是一个项目里面自己制作的动态库的数量不超过6个。
  ii. ObjC setup：注册Objc类，进行selector唯一性检测等。可以通过减少Objc类的数量，减少selector的数量来进行优化。
  iii. rebase/binding：rebase指针修复，binding符号绑定。这一步的优化手段和第2步一样。
  iv. initializer：各种初始化的操作，比如执行objc的+load函数，C++的构造函数等。不要在+load函数里面做一些耗时的操作，或者把一些操作延时的放在+initialize里面去执行。

- main，从main()函数开始到执行完appDelegate的didFinishLaunchingWithOptions方法展示首页数据。
  优化手段：

  i. 少使用xib和storyboard。
  ii. 删除NSLog打印。
  iii. 整理didFinishLaunchingWithOptions方法里面的业务逻辑，可以异步请求的异步请求，可以延时加载的延时加载。

ASLR

ASLR(address space layout randomization)，地址空间布局随机化。是一种针对缓冲区溢出的安全保护技术，通过对堆、栈、共享库映射等线性区布局的随机化，通过增加攻击者预测目的地址的难度，防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的的一种技术。使得可执行文件和动态库在虚拟内存中的地址在每次启动都不固定。

Page Fault

由于虚拟内存的出现，进程不直接访问物理内存，这样安全性更高。为了提高效率和方便管理，对虚拟内存和物理内存进行分页(Page)管理。当进程访问一个虚拟内存页，而对应的物理内存却不存

在的时候，会触发一个缺页中断，这个就叫Page Fault。

二进制重排

`概念`：找到程序在启动时候需要调用的符号，然后修改编译参数完成二进制文件的重新排布。

`LinkMap`：是iOS编译过程的中间产物，记录了二进制文件的布局。可以在Xcode的Build Settings里开启Write Link Map File。

`order文件`：编译器会按照order文件的内容，对二进制文件进行排列。可以在Xcode的Build Settings里的Order File处设置。

`二进制重排能够优化启动时间的原理`：App在执行的过程中会存在大量的Page Fault，一个Page Fault的耗时很少，但是当大量的Page Fault存在时，就会影响到代码的执行速度。同理，在App启动的时候，就可能会出现大量的Page Fault。二进制重排就是把在启动过程中需要使用到的符号，重新排列在一个或者几个Page里面，减少Page Fault的次数，从而达到减少启动时间的目的。