转载请注明链接：https://blog.csdn.net/feather_wch/article/details/81135996

Android面试题的基础题部分，包括系统架构，APK打包/安装、序列化等各方面的基本题目，因为范围比较杂，难以归类，所以都放在该文章中。

本文是我一点点归纳总结的干货，但是难免有疏忽和遗漏，希望不吝赐教。

Android面试题-基础题(18题)

版本：2018-7-25(17:17)

1、Android的系统架构分为几层，大概有哪些内容?

> 1. 应用层 ： Acyivity、Application、Service.
> 2. `Binder`通信层 ： Binder Proxy、Binder Driver
> 3. 系统服务层 ： WindowManagerService、ActivityManagerService
> 4. 运行时层 ： Davlik Runtime/ART Runtime
> 5. 硬件抽象层 ： Camera HAL, Audio Hal
> 6. `Linux`内核层 ： Camera Driver，Audio Driver

Intent

2、Android中Intent传递数据的大小限制？如何解决该问题?

> 1. 大小限制大约是`1MB` ： 超过该限制可能导致 OOM。
> 2. 解决办法
>   1. 进程内： `EventBus` 、文件缓存
>   2. 进程间：通过 `ContentProvider` 进行进程间数据共享和传递。
>
> 总结： 不要通过 Intent 在 Android 基础组件之间传递大数据（binder transaction 缓存为1MB）

APK

3、APK的打包流程分为几部分?

> 1. 资源打包
> 2. 代码打包

4、APK打包流程(6步)

> 1. APPT工具 对 资源文件进行打包(AndroidManifest.xml、布局等) 生成 R.java文件；通过 AIDL工具 处理AIDL文件，生成对应的 Java文件 。
> 2. Javac工具 对 R.java、项目源码、aidl对应的Java文件 这三部分进行编译，生成 class文件
> 3. Dex工具 将所有 class文件 转换为 DEX文件 ： 该过程进行将 Java字节码 转换为 Dalvik字节码 、 压缩常量池 、 清除冗余信息 等工作。
> 4. ApkBuilder工具 将 资源文件、Dex文件 打包成 APK文件
> 5. KeyStore 对 APK文件 进行签名。
> 6. 正式版APK，需要用 ZipAlign工具 进行对齐处理：过程中是将APK中所有资源文件的起始地址都便宜 4字节的整数倍 ， 通过内存映射访问APK文件的速度会更快

5、APK的安装流程(6步)

> 1. 复制APK 到 /data/app目录 下，解压并扫描安装包。
> 2. 资源管理器 解析 APK 里的资源文件。
> 3. 解析 AndroidManifest 文件，并在 /data/data/ 目录下 创建对应的应用数据目录
> 4. 对 dex文件进行优化 ， 并且保存在 dalvik-cache 目录下。
> 5. 将 AndroidManifest文件 解析出的 四大组件信息 注册到 PackageManagerService 中。
> 6. 安装完成后，发送广播

6、点击应用图标是如何启动APP的?

> 1. 点击应用图标后会去启动应用的 LauncherActivity
> 2. 如果 LauncherActivity 所在进程没有创建，就会创建新进程（以Socket形式通知 Zygote进程 去fork新进程）。
> 3. 整体的流程就是 Activity 的启动流程。

7、PathClassLoader和DexClassLoader的区别

> 1. PathClassLoader：只能加载安装到Android系统的APK，即 /data/data 目录，是Android默认的类加载器。
> 2. DexClassLoader：可以加载任意目录下的 dex、jar、apk、zip 文件。

Context

7、Android有哪些Context的相关类?

1. Context抽象类
2. ContextImpl ： Context的实现类
3. ContextWrapper ： Context的包装类（内部是 ContextImpl ）， Applicaiton、Activity、Service 都间接或者直接继承自 ContextWrapper

进程

8、Android有哪些进程

1. 前台进程: 用户当前操作所必须的进程
2. 可见进程：没有任何前台组件，但是任辉影响屏幕上可见内容的进程。
3. 服务进程：正在运行服务（该服务通过startService（）启动），且不属于上面两者的进程。
4. 后台进程：包含目前对用户不可兼得Activity的进程。
5. 空进程：不包含任何活动组件的进程。

9、前台进程需要满足的条件（下列条件的任意一个）

1. 具有 可交互的Activity ---该Activity已经调用 onResume()
2. 通过 bindService 启动的Service，并且绑定的Activity处于可交互状态。此时 Service 的进程就是前台进程。
3. 明确调用了前台Service(Service 执行了 startForeground() 方法)---高版本的Service都必须要调用 startForeground ， 不然很容易被杀死。
4. 正在执行某一生命周期的Service(onCreate、onStart、onDestory)
5. 正在执行 onReceive() 的BroadcastReceiver处于前台进程。(onReceive() 是在UI线程执行的，即使用户在Home首页，此时广播接受器接收到广播就会处于前台进程。)

10、可见进程需要满足的条件(任意一条即可)

1. 具有不在前台，但是任然可见的 Activity
2. 绑定到 可见或者前台Activity 的 Service

11、服务进程的作用

1. 不影响用户交互的内容。
2. 但是执行比较重要的任务，比如 下载

12、空进程的作用

1. 用于 缓存 ， 能缩短下一次打开组件的时间。

13、如何进行进程保活

1. 提升进程优先级，降低进程被杀死的概率。
2. 拉活已经被杀死的进程。

14、如何提升优先级?

> 1. 监控手机锁屏事件
> 2. 锁屏 是启动一个像素的Activity，在用户解锁的时候，将Activity销毁掉，前台Activity会将进程变成前台进程，且优先级最高

15、如何拉活已经杀死的进程

> 1. 利用广播拉活Activity
> 2. 手机去监听 系统广播 ： 如开机广播，锁屏解锁广播等。

序列化

16、什么是序列化

> 序列化 就是将 对象 转化为 二进制流 ， 便于 存储和传输

17、Serializable和Parcelable的区别

> 1. Serilizable 是java实现的一套序列化方法，会触发频繁的IO操作，效率较低，适合将对象存储到磁盘上的情况。
> 2. Parcelable 是 Android 提供的序列化方法。 Parcelable 将序列化后的字节流写入到一个共享内存中，其他对象可以从该共享内存中读出字节流，并反序列化成对象，效率较高，适合对象间和进程间传递信息。

其他

18、64k问题的产生原因和如何解决?

> 1. 产生原因： Dex文件 中class、method的索引使用 short类型 ， 因此如果方法、类的总数超过了 2字节的short-65535 就会出问题。
> 2. 解决办法：使用 Google的Multidex

参考资料

1. 进程保活