前言

在过去不久四五月面试黄金期，有些小伙伴已经找到了理想的工作，当然也有很多小伙伴因为准备不充分，面试挂了，临近毕业季，最近有很多网友都在求大厂面试题；正好我在4月份和5月份整理和收集了 Android
基础高级面试真题解析，于是就发上来分享给大家

> 这些题目是网友去字节跳动、小米、oppo、美团、阿里巴巴、腾讯、360、华为、京东
> 等一线互联网公司面试被问到的题目；熟悉本文中列出的知识点会大大增加通过前两轮技术面试的几率

下面给大家举例一下在面试中经常被问到的问题

view的绘制流程

简单介绍下measureSpace

> Measure
> Space（32位的int值）分为两部分组成；高2位即30、31是mode测量模式；底30位是size测量大小

测量模式mode共分为三种

> 1、exactly：对应的是layoutparams中的match_parent和具体数值这两种模式，这时候view的最终大小就是specsize所指的值
>
> 2、at_most 对应layoutparams中的wrap_content
>
> 3、unspecified 不对view进行任何限制，要多大给多大，一般用于系统内部，如listview，scrollview等

getwidth和getMeasureWidth的区别

getMeasurewidth必须在onmeasure之后才有效，getMeasureWidth的取值最终来源于setMeasureDimension方法调用时传递的参数，getWidth必须在layout执行之后才有效

invalidate和postinvalidate的区别

两者都会刷新view，并且当view为visible的时候。view的onDraw方法才会调用，invalidate是用于UI线程调用的，post是用于非UI线程调用的

requestlayout的作用

requestlayout也可以达到重绘view的目的，它会先调用onLayout重新排版，再调用onDraw
方法，当view确定自身不再适合现有的区域时，该view会要求父控件重新调用onMeasure、onlayout来重新设置自己的位置

ondraw和dispatchdraw的区别

ondraw是用于绘制view本身的内容；而绘制子view是通过dispatchDraw实现的

自定义view为什么不管设置什么类型的宽高，显示的都是占满屏幕

自定义view如何设置margin和padding

事件分发机制

AppcompatActivity的xml解析流程

首先调用Appcompatactivity
的setContentView方法，然后调用AppcompatDelegateImpl的setContentView方法，在AppcompatDelegateImpl中首先会调用ensureSubDecor方法，初始化mSubDecor变量，并最终通过mSubDecor.findViewById(android.R.id.content);获取到contentParent，再通过LayoutInflater.from(mContext).inflate(resId, contentParent)来加载解析布局

ams pms

ams主要是用来统一调度各应用程序的Activity，内存管理和进程管理

pms用来管理所有的package信息，包括安装、卸载、更新以及解析AndroidManifest.xml

谈谈你对 binder 的理解

binder 是 Android 中主要的跨进程通信方式，binder通过 mmap 实现一次拷贝，并基于C/S架构，架构清晰，职责明确，比 Socket 、管道传输速度更快，比共享内存更安全可控

intent 传参有大小限制，这跟 binder 有关系吗？

intent可以携带一些基本数据类型
，或者序列化
对象，一般情况下如果数据太大，可能就会出现异常，其根本原因就是受binder限制的，binder在mmap的时候设置了一个内存上限，其大小就是1M-8k

基于 mmap 又是如何实现一次拷贝的？

mmap其本质是一种进程虚拟内存的映射方法，它可以将一个文件、一段物理内存或者其它对象(也包括内核空间)映射到进程的虚拟内存地址空间，实现这样的映射关系后，进程间就可以采用指针的方式来读写操作这一段内存，进而完成对文件(或者其它被映射的对象)的操作

在传输数据时，发送方进程通过系统调用copy_from_user将数据拷贝到内核空间，接收方进程就可以根据这个映射关系得到数据，这样就实现一次拷贝了

怎么理解页框和页？

页框
是指一块实际的物理内存，页是指程序的一款内存数据单元。内存数据一定是存储在实际的物理内存上，即页必然对应于一个页框，页数据实际是存储在页框上的；页框和页一样大，都是内核对内存的分块单位；一个页框可以映射多个

也就是说一块实际的物理存储空间可以映射多个进程的多个虚拟内存空间，这也是mmap机制依赖的基础规则

简单说下 binder 的整体架构吧

ams启动流程

ams的启动是在systemServer进程中启动的，当systemServer进程启动时，会调用它的main函数
，然后调用systemServer的run方法，在run方法中会创建一个SystemServiceManager对象，然后调用startBootstrap
Services方法用这个SystemServiceManager对象去ams，pms等服务

点击应用图标启动APP流程

首先桌面也是一个应用程序，最终也是通过ams来启动APP，ams在启动app时，会检查app进程是否存在，不存在就会请求Zygote进程将需要的应用进程启动，并最终调用activityThread的main函数，在main函数会初始化ActivityThread，并调用attach方法

在这里会通过activityManager.getService得到Ams代理对象，并通过attachApplication函数传入applicationThread对象，让ams持有applicationThread

这样方便AMS与activityThread互相调用，在ams中会初始化一些关于APP的application和activity的数据，并通过thread.bindApplication方法回调给applicationThread，然后会通过sendMessage发送一条bind_application消息，通知Handler

将传递过来的数据创建一个application对象，通过mInstrumentation对象来调用application的onCreate方法，创建完application后会继续执行am

s后面的逻辑

首先会通过调用Activity堆栈对象的方法，创建一个activity的事务管理器，并
最终
回调给act
ivityThread的sche
duleTransaction函数，然后再通过sendMessage
发送一条消息去执行事务，然后通过lunchActivityItem发送启动窗体的请求，再回调给ActivityThread的handleLaunchActivity函数中

通过performLaunchActivity创建一个activity并通过mInstrumentation.c
allActivityonCreate
函数，调用activity的preformCreate，最终执行activity的onCreate函数

activity启动流程

从Activity中调用了startActivity()之后，经过一系列跳转，会执行到instru
mentation的exec
StartActivity方法中,然后通过ActivityTaskManager.getService()获得Acti
vityTaskManagerService代理对象并调用ActivityTaskManagerService的
startActivity方法

最终会执行ActivityTaskManagerService的startActivityAsUser方法，根据启动intent去找到或者创建合适的Task来启动activity

这个Instrumentation是什么呢？

可以说它是应用进程的管家，监控着应用进程与系统的所有交互，所有的创建、暂停、停止activity，都是通过它去发起的，它可以统计所有的开销

说了这么多，其实最重要的
就是一句话，问问你自己：
你现在所拥有的技术层次真的有信心在这家公司入职吗？

近段时间我这里整理了一份完整的《2022年 Android

中高级面试题汇总》希望这份系统化的技术体系对大家有一个方向参考

有需要的同学，可以顺手给我点赞评论支持一下

内容如果对大家有用的话，可以转发分享一下

获取方式：

私信发送“面试”或“进阶”即可免费获取

《2022年 Android 中高级面试题汇总》

由于篇幅有限，仅展示部分内容

第一章 Java 基础

第一节 静态内部类和非静态内部类的比较

1.1 静态内部类和非静态内部类的区别

1.2 扩展:内部类都有哪些?

1.3 同部内部类

1.4 匿名内部类:是一种没有炎名的内部类

第二节 多态的理解与应用

2.1 多态概述

2.2 多态中成员的持点

2.3 instanceof关桩字

2.4 多态的转型

2.5 多态案例

第三节 java 方法的多态性理解

3.1 什么是java的多态

3.2 远行时多态3.3代码理解

3.4 深一点

3.5 再深一点

3.6 最后一个炼习

第四节 java中接口和选承的区别

第五节 线程池的好处，详解，单例(绝对好记)

5.1 线程池的好处

5.2 线程池的详解

5.3 线程池的单例

第二章 Android 基础

第一节 Activity 知识点（必问）

1.1 Activity 启动过程全解析

1.2 启动模式以及使用场景

1.3 onSaveInstanceStatelJBonRestoreInstar

1.4onConfigurationChanged使用以及问题解决

第二节 Fragment 知识点

2.1 Fragment的通信问题，新建Fragment为何不

2.2 为什么官方推荐Fragment.setArguments(B

2.3 Androidx下Fragment懒加载的新实现

2.4 Fragment全解析系列(一）:那些年深过的

2.5 Google-Fragment 概览

2.6 Google -与其他 Fragment 通信

第三节 Service 知识点

3.1 Handle 知识点（必问）

3.2 Android 主线程阻塞处理及优化

3.3深入聊聊Android消息机制中的消息队列的

3.4深入理解MessageQueue

3.5 你真的懂Handler.postDelayed(的原理吗?

3.6 Handler.postDelayed0是如何精确延迟指成

3.7 Handler 延迟消息执行机制，会阻塞吗?

第四节Intent知识点

4.1 Android 跨进程传递大内存数据

4.2 数据存健

获取方式：

私信发送“面试”或“进阶”即可免费获取

技术是无止境的，你需要对自己提交的每一行代码、使用的每一个工具负责，不断挖掘其底层原理，才能使自己的技术升华到更高的层面

Android 架构师之路还很漫长，与君共勉

PS:有问题欢迎指正,可以在评论区留下你的建议和感受；

欢迎大家点赞评论，觉得内容可以的话，可以转发分享一下