转载请注明链接：https://blog.csdn.net/feather_wch/article/details/79190230

总结所有Android进程间通信相关的面试题。在其他博文已经有答案的会提供猎羽的博文链接进行参考。没有答案的会在文章内给出。

Android面试题-IPC

版本号：2018/09/02-1(14:30)

注意

- [☆] 标记的题目，是额外补充的题目，会直接给出答案。
- 没有标记的题目，详细答案请参考：答案参考-Android进程间通信IPC详解

基本概念(19)

1. 什么是IPC?
2. 进程间通信是什么?
3. 进程是什么?
4. Android中的IPC方法(6种)
5. 线程是什么?
6. ANR导致的原因？如何避免?
7. 什么时候需要用到多进程通信?
8. 开启多进程模式的方法
9. activity的 `android:process` 属性 `=":remote"` 和 `"com.example.remote"` 的区别
10. 多进程会造成的问题？(4)
11. Serializable和Parcelable接口作用?(3)
12. Serializable接口的作用和使用?
13. serialVersionUID的作用?(2)
14. 静态成员变量是否会参加序列化过程?
15. 用 `transient` 关键字标记的成员变量是否会参加序列化过程?
16. java.io.ObjectOutputStream和ObjectInputStream有什么用?
17. 系统中哪些类实现了Parcelable接口?
18. 实现序列化的集合中的元素是否一定要实现序列化?
19. Parcelable和Serializable的区别？(2)

Linux内核基本知识(4)

1. [☆]什么是Linux中的进程隔离?
    1. 用于保护操作系统中的进程互不干扰
    2. 让进程之间无法随意访问和改变数据。
2. [☆]什么是Linux中的虚拟地址空间?
    1. 用来实现进程隔离，因为进程AB所处于的虚拟地址空间不同，所以不能去改变其他进程的数据。
    2. 不同进程间数据不共享，会让进程以为自己独享操作系统。
3. [☆]Linux的系统调用是什么?
    1. 内核会有保护机制，只允许访问许可的数据。
    2. 将Linux的内核层和用户层，抽象分离开。
    3. 用户层可以通过系统调用去访问内核层的资源。
4. [☆]Linux中的binder驱动是什么?
    1. 在android系统的内核空间中，一个负责各个进程通过Binder机制交互的驱动模块就是Binder驱动。
5. [☆]
    1.
    2.

Binder(42)

1. Binder是什么？(从五个角度考虑)
2. [☆] Binder对于Server和Client来说是什么?
    1. 对于服务端：Binder就是指Binder本地对象
    2. 对于客户端：Binder就是指Binder代理对象
    3. 本质通过内核层的ServiceManager进行通信。
3. 设备驱动的作用？好处?
4. Binder机制是做什么的？(2)为什么使用Binder?
    1. 用于进程间通信
    2. Android中各个app、系统服务运行于不同进程中，需要借助Binder进行通信。
    3. 从Linux内核角度看，Linux存在进程隔离和虚拟地址空间，导致进程间无法共享数据。所以需要借助IPC。
5. Android基于的Linux内核，Linux有哪些IPC方法？(6种)优缺点是什么?
6. Binder机制相比于LinuxIPC的优势体现在哪里？(3方面)
7. [☆] Binder的通信模型
    1. Binder驱动：进程隔离的存在，进程间无法直接访问，需要借助内核层提供的系统调用来访问内核层。
    2. 服务端：Server1、2、3等服务，需要借助 `Binder驱动` 去ServiceManager `注册` 。会在存储在表中。
    3. 客户端：Client借助Binder驱动去ServiceManager `查询` 表。
    4. 服务/客户通信：借助Binder驱动进行之间的 `IPC` 。
8. 内部基于Binder的技术？(3)
    1. Message
    2. AIDL和Message(内部AIDL)
9. AIDL文件的本质作用
10. 通过AIDL快速实现Binder的步骤?(4步)
11. 使用aidl时内部类stub是什么?

> Binder类

12. Stub的内部代理类Proxy如何处理客户端和服务端是否在同一个进程的情况?
    > 不同进程，方法调用后会走跨进程的transact过程。同一进程就不会。

13. Binder的工作流程？(4步)
14. 如何通过AIDL自动生成Binder的java文件?
    > 编写好aidl后，通过“make project”生成。

15. android studio的build中make project生成的aidl文件在哪?
    > 位于目录 `app\build\generated\source\aidl\debug\`包下

16. Binder类中Stub类的DESCRIPTOR是什么?
    > Binder的唯一标识,一般用类名表示

17. Binder类中Stub类的asInterface(android.os.IBinder obj)的作用?
    > 1. 将服务端Binder对象转换为客户端所需要的ADIL类型接口对象。
    > 2. 服务端和客户端在同一进程，直接返回服务端的Stub对象
    > 3. 服务端和客户端在不同一进程，返回服务端的Stub对象的代理对象Stub.proxy

18. Binder类中Stub类的asBinder()的作用
    > 返回当前Binder对象

19. Binder类中Stub类的 onTransact(code, data, reply, flags)的作用?
    > 1. 该方法运行在服务端的Binder线程池中。
    > 2. 通过code确定Client请求的目标方法
    > 3. 从data中取得方法所需参数，执行目标方法。
    > 4. 执行完毕后就向reply中写入返回值。
    > 5. 如果此方法返回false就表示客户端请求失败，我们可以用此来进行权限验证。

20. Binder类中Stub.Proxy的自定义方法的内部流程？注意点?
    > 1. 这些用户自定义方法，运行在客户端。
    > 2. 将该方法的参数信息写入到data中，调用transact方法发起RPC(远程调用请求)，此时当前线程挂起。
    > 3. 服务端的onTransact会被调用，执行完毕后，将结果写入到reply中。
    > 4. 客户端的RPC调用返回后，当前线程继续执行，并从reply中取出返回结果，然后返回reply中的数据。

21. AIDL如何进行权限验证?
    > 1. 利用Binder类也就是Stub的onTransact方法
    > 2. 该方法如果返回false就代表请求失败，可以用于权限验证
    > 3. 在Service的onBind中进行权限验证，比如使用permission验证等等。如果验证不通过则绑定服务就失败。

22. 通过Binder机制，因为具有线程池，所以客户端可以在UI线程直接发起远程调用请求?
    > 错误！！！
    >
    > 1. 客户端的远程调用会挂起当前线程，因此UI线程发起RPC(远程调用)会导致阻塞。
    > 2. Binder机制中的线程池，是指服务端的onTransact()运行在Binder线程池中。
    > 3. 服务端的方法都运行在线程池中，因此要注意同步问题。

23. Binder如何检测服务端异常终止？linkToDeath和unlinkToDeath方法有什么用?
    > 1. 异常终止问题：如果服务端异常终止，而会导致客户端调用失败，甚至可能客户端都不知道binder已经死亡，就会产生问题。
    > 2. 利用linkToDeath和unlinkToDeath可以解决该问题。
    > 3. linkToDeath作用给Binder设置一个死亡代理，Binder会收到通知，还可以重新发起连接请求从而恢复连接。

> 4. binder的isBinderAlive也可以判断Binder是否死亡。

24. 有进程A和进程B，如何利用Binder机制进行两者的通信？核心步骤(6)?
    > 1. 进程A去绑定到进程B注册的Service
    > 2. 进程B创建Binder对象---Servcice的onBind()
    > 3. 进程A接收进程B的Binder对象
    > 4. 进程A利用进程B传过来的对象发起请求
    > 5. 进程B收到并处理进程A的请求
    > 6. 进程A获取进程B返回的处理结果

25. Binder对象是如何具有能被跨进程传输的能力？(2)
26. Binder对象如何具有完成特定任务的能力?
27. 继承IBinder接口实现的attachInterface()、queryLocalInterface()两个方法的作用?
    > 1. binder.attachInterface(): 将key=DESCRIPTOR和value=IInterface对象的键值对，存储在Binder内部。
    > 2. binder.queryLocalInterface(): 通过 DESCRIPTOR查询到IInterface对象。

28. 实现IInterface接口的类中是干什么的?
29. attachInterface()建立了与IInterface对象的关系，为什么通过IInterface对象就能获取到执行特定任务的能力?
30. 服务端的onBind()返回Binder对象后，系统做了什么工作?
    > 1. 系统会收到这个binder对象，然后会生成一个BinderProxy。并且返回给客户端。

31. 客户端在onServiceConnected()方法中就是获取到服务端的Binder对象?
    > 错误!
    >
    > 1. 客户端、服务端在同一个进程，就是服务端Binder对象。
    > 2. 不在同一个进程，就是服务端Binder对象的代理对象Proxy

32. 客户端通过BinderProxy执行操作的原理?
    > 内部借助transact()和onTransact()完成功能的实现。

33. 系统保存了Binder对象和BinderProxy对象的对应关系
34. 对BinderProxy的优化
    > 1. 借助Stub.asInterface(binderproxy)
    > 2. 将BinderProxy生成一个代理对象，内部封装了transact()和onTransact()的流程
    > 3. 通过add()等自定义方法直接进行执行，给人一种获取到服务端IInterface对象的假象。

35. Binder机制中transact()方法和服务端的线程池都是由系统实现的?
36. Binder的服务端为什么不是线程安全的?
    > Server端使用了线程池决定了Server的代码 不是线程安全的

37. Binder服务端的线程池是如何实现的?
    > 1. 系统实现：线程池实现在Binder内部的native方法中

38. ContentProvider中的CRUD(增删改查)是否是线程安全的?
39. AIDL中在Service中实现的接口是否是线程安全的？(mBinder = xxx.Stub())中的接口
40. IBinder接口作用?
    > 1. IBinder是用于远程对象的基本接口，是轻量级远程调用机制的核心部分，用于高性能地进行进程内部和进程间调用。
    > 2. IBinder描述了远程对象间交互的抽象协议。

41. 平时直接实现IBinder接口就可以了?
    > 不能，系统要求去直接继承自Binder类。

42. Binder源码：Binder构造方法中的init是干什么的?
    > init()是native方法，是对底层 `Binder Driver` 的封装。

43. 客户端发起请求时的底层流程是什么？客户端transact是如何回调到服务端的onTransact方法的?
    > 1. 调用IBinder的 `transact()` 接口
    > 2. 调用驱动层的 `mRemote.transact()`
    > 3. Binder Driver会调用 `execTransact()`，内部调用服务端Binder的 `onTransact()` 方法。

44. Binder的Driver层作用
    > 对Binder类进行了完美的封装，开发者只需要继承 `Binder` 和实现 `onTransact()` 即可。

6种IPC(35)

1. Bundle的作用
2. A进程在完成计算后需要启动B进程的一个组件并且将结果传递给B进程，但是这个结果不支持放入Bundle，因此无法通过Intent传输。这个该如何解决?
3. 文件共享的作用?
4. 文件共享可以传递对象吗?
5. 文件共享的特点:
6. Messenger是什么?
7. Messenger中服务端是否需要考虑线程同步问题?
    > 1. 不需要，因为一次处理一个请求

8. 使用Messenger时，客户端的步骤
    > 1. bindService()去绑定服务
    > 2. 在ServiceConnection中用返回的Binder对象创建Messenger，利用该Messenger能读取或者发送信息。
    > 3. 发送信息：1- `msg.setData(bundle)` 2- `msg.replyTo = mGetReplyMessenger` 3- `mMessenger.send(msg)` 4- mGetReplyMessenger继承自Messenger用于接收服务端回复的消息。

9. 使用Messenger时，服务端的步骤?
    > 1. `Messenger clientMessenger = msg.replyTo;` 获取到客户端传来的Messenger
    > 2. 利用该Messenger发送信息

10. Messenger缺点?
11. AIDL进程间通信流程
12. [☆] AIDL是什么?
    > 1. Android Interface Definition Language, 即Android接口定义语言。
    > 2. 系统提供的一种快速实现Binder的工具。

13. AIDL支持的数据类型
    > 1. 基本数据类型(int、long、char、boolean、double等)
    > 2. String和CharSequence
    > 3. List：只支持ArrayList，且里面所有元素必须是AIDL支持的数据。
    > 4. Map：只支持HashMap，且里面所有元素必须是AIDL支持的数据，包括key和value
    > 5. Parcelable：所有实现Parcelable接口的对象
    > 6. AIDL：所有AIDL接口都可以在AIDL中使用

14. AIDL中List只能用ArrayList，远程服务端能使用CopyOnWriteArrayList(并非继承自ArrayList):
    > 1. Binder会根据List规范去访问数据，并且生成一个新的ArrayList传给客户端，因此没有违反数据类型的规定。
    > 2. ConcurrentHashMap也是类似功能

15. 如何在AIDL中使用观察者模式?
16. AIDL观察中解除注册会导致服务端找不到该观察者，这是为什么?
17. 如何用RemoteCallbackList解决这个问题?
    > 系统专门提供用于删除跨进程Listener的接口。该类本身是一个泛型，支持任何AIDL接口。

18. RemoteCallbackList实现原理
    > 1. 内部有一个Map结构，key是Ibinder，value是Callback类型。
    > 2. 将Listener的信息存入CallBack, Callback能保存所有AIDL回调
    > 3. 每次跨进程Client的同一个对象，会在服务端生成多个对象。但是这些对象本身的Binder都是同一个。
    > 4. 以Binder作为Key，能找到唯一的Listener。

19. 使用RemoteCallbackList的流程和特点
    > 1. 客户端解注册，服务端会遍历所有listener，找出那个和解注册的listener具有相同Binder的listener，并删除。
    > 2. 客户端终止后，会自动移除客户端注册的所有listener
    > 3. 自动实现线程同步，无需额外操作。

20. 客户端的onServiceConnected和onServiceDisconnected可以进行耗时操作吗?
    > 1. 不可以，都运行在UI线程池中。

21. 服务端的方法需要开线程进行异步操作吗?
    > 不需要。本身已经运行在服务端的Binder线程池中。

22. 服务端中的方法能直接去操作控件吗?
    > 不可以，需要通过Handler切换到UI线程。

23. Binder意外死亡的两种解决方法?
24. Binder意外死亡的两种解决方法:
    > 1. 客户端的onServiceConnected中调用 `Binder.linkToDeath(mDeathRecipient)` 设置DeathRecipient监听，Binder死亡时回调DeathRecipient.binderDied()
    > 2. 或者在onServiceDisconnected中重连远程服务。

25. Binder意外死亡的解决方法的区别
    > 1. onServiceDisconnected在客户端UI线程中被回调
    > 2. binderDied在客户端的Binder线程池中被回调(无法操作UI)

26. onServiceDisconnected是如何解决Binder死亡问题的？何时被调用?
27. 大量业务模块都需要使用AIDL进行进程间通信，如何在不创建大量Service的情况下解决该问题?(=Binder连接池是什么？)
28. ContentProvider是什么
29. ContentProvider的使用
30. ContentResolver的作用
31. 自定义ContentProvider的步骤
    > 1. 继承ContentProvider
    > 2. 实现：onCreate-创建的初始化工作
    > 3. getType-返回url请求对应的MIME类型(媒体类型)，比如图片、视频等。不关注该类型，可以返回null或者 “*/*”
    > 4. query-查数据
    > 5. insert-插入数据
    > 6. update-更新数据
    > 7. delete-删除数据

32. ContentProvider的onCreate()运行在ContentProvider进程中的主线程中?
33. getType、增删改查这五个方法由外界回调，运行在外界的线程中?
    > 错误！都运行在ContentProvider所在进程的Binder线程池中
34. ContentProvider存储方式
35. Sokcet如何可以进行进程间通信

IPC方式总结(1)

1. IPC各种方法的优缺点和适用场景

名称	优点	缺点	适用场景
Bundle	简单易用	只能传输Bundle支持的数据类型	四大组件间的进程间通信
文件共享	简单易用	不适合高并发场景， 无法做到进程间的即时通信	无并发访问情形， 交换简单数据实时性不高的场景
AIDL	功能强大， 支持一对多并发通信， 支持实时通信	适用稍复杂， 需要处理好线程同步	一对多通信且有RPC需求
Messenger	功能一般， 支持一对多串行通信， 支持实时通信	不能很好处理高并发情形， 不支持RPC， 数据通过Message传输因此只能传输Bundle支持的数据类型	低并发的一对多即时通信， 无RPC需求， 或者无须要返回结果的RPC需求
ContentProvider	在数据源访问方面功能强大， 支持一对多并发数据共享， 可通过Call方法扩展其他操作	可以理解为受约束的AIDL， 主要提供数据源的CRUD操作	一对多的进程间的数据共享
Socket	功能强大， 可以通过网络传输字节流， 支持一对多并发实时通信	实现细节稍微繁琐， 不支持直接的RPC	网络数据交换

参考资料

1. 面试题-听说你精通Binder?