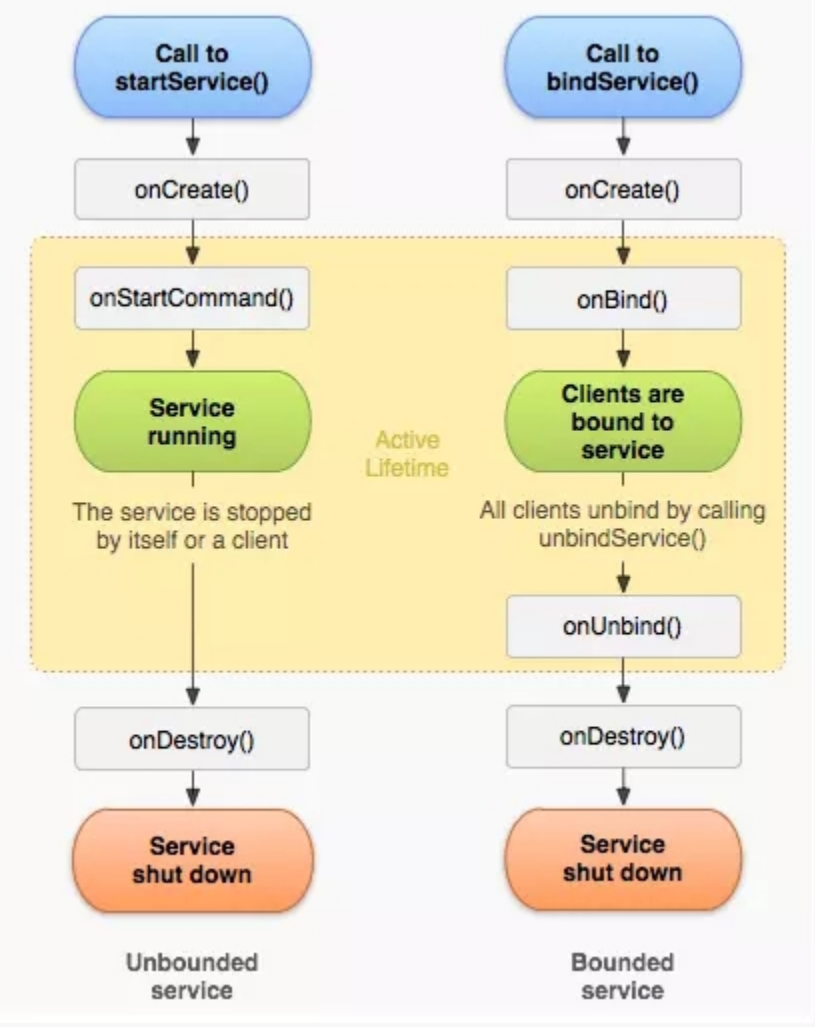
Service

1、谈一谈 Service 的生命周期？

- 参考回答：Service 的生命周期涉及到六大方法
  - **onCreate()**：如果 service 没被创建过，调用 startService()后会执行 onCreate()回调；如果 service 已处于运行中，调用 startService()不会执行 onCreate()方法。也就是说，onCreate()只会在第一次创建 service 时候调用，多次执行 startService()不会重复调用 onCreate()，此方法适合完成一些初始化工作；
  - **onStartComand()**：服务启动时调用，此方法适合完成一些数据加载工作，比如会在此处创建一个线程用于下载数据或播放音乐；
  - **onBind()**：服务被绑定时调用；
  - **onUnBind()**：服务被解绑时调用；
  - **onDestroy()**：服务停止时调用；

2、Service 的两种启动方式？区别在哪？

- 参考回答：Service 的两种启动模式
  - **startService()**：通过这种方式调用 startService，onCreate()只会被调用一次，多次调用 startSercie 会多次执行 onStartCommand()和 onStart()方法。如果外部没有调用 stopService()或 stopSelf()方法，service 会一直运行。
  - **bindService()**：如果该服务之前**还没创建**，系统回调顺序为 onCreate()→onBind()。如果调用 bindService()方法前服务**已经被绑定**，多次调用 bindService()方法不会多次创建服务及绑定。如果调用者希望与正在绑定的服务**解除绑定**，可以调用 unbindService()方法，回调顺序为 onUnbind()→onDestroy()；



3、如何保证 Service 不被杀死 ？

- 参考回答：
  - onStartCommand 方式中，返回 START_STICKY 或则 START_REDELIVER_INTENT
    - START_STICKY：如果返回 START_STICKY，表示 Service 运行的进程被 Android 系统强制杀掉之后，Android 系统会将该 Service 依然设置为 started 状态（即运行状态），但是不再保存 onStartCommand 方法传入的 intent 对象
    - START_NOT_STICKY：如果返回 START_NOT_STICKY，表示当 Service 运行的进程被 Android 系统强制杀掉之后，不会重新创建该 Service
    - START_REDELIVER_INTENT：如果返回 START_REDELIVER_INTENT，其返回情况与 START_STICKY 类似，但不同的是系统会保留最后一次传入 onStartCommand 方法中的 Intent 再次保留

下来并再次传入到重新创建后的 Service 的 onStartCommand 方法中

- **提高 Service 的优先级** 在 AndroidManifest.xml 文件中对于 intent-filter 可以通过 android:priority = "1000"这个属性设置最高优先级，1000 是最高值，如果数字越小则优先级越低，同时适用于广播；
- **在 onDestroy 方法里重启 Service** 当 service 走到 onDestroy()时，发送一个自定义广播，当收到广播时，重新启动 service；
- **提升 Service 进程的优先级** 进程优先级由高到低：前台进程 一 可视进程 一 服务进程 一 后台进程 一 空进程 可以使用 startForeground 将 service 放到前台状态，这样低内存时，被杀死的概率会低一些；
- **系统广播监听 Service 状态**
- **将 APK 安装到/system/app，变身为系统级应用**

- **注意**：以上机制都不能百分百保证 Service 不被杀死，除非做到系统白名单，与系统同生共死

4、能否在 Service 开启耗时操作 ？ 怎么做 ？

- 参考回答：
  - Service 默认并不会运行在子线程中，也不运行在一个独立的进程中，它同样执行在主线程中（UI 线程）。换句话说，不要在 Service 里执行耗时操作，除非手动打开一个子线程，否则有可能出现主线程被阻塞（ANR）的情况；

5、用过哪些系统 Service ？

- 参考回答：

常用系统服务

传入的Name	返回的对象	说明
WINDOW_SERVICE	WindowManager	管理打开的窗口程序
LAYOUT_INFLATER_SERVICE	LayoutInflater	取得XML里定义的View
ACTIVITY_SERVICE	ActivityManager	管理应用程序的系统状态
POWER_SERVICE	PowerManger	电源服务
ALARM_SERVICE	AlarmManager	闹钟服务
NOTIFICATION_SERVICE	NotificationManager	状态栏服务
KEYGUARD_SERVICE	KeyguardManager	键盘锁服务

6、了解 ActivityManagerService 吗？发挥什么作用

- 参考回答： ActivityManagerService 是 Android 中最核心的服务， 主要负责系统中四大组件的启动、切换、调度及应用进程的管理和调度等工作，其职责与操作系统中的进程管理和调度模块类似；

Broadcast Receiver

1、广播有几种形式 ？都有什么特点 ？

- 参考回答：
  - 普通广播：开发者自身定义 intent 的广播（最常用），所有的广播接收器几乎会在同一时刻接受到此广播信息，**接受的先后顺序随机**；
  - 有序广播：发送出去的广播被广播接收者**按照先后顺序接收**，同一时刻只会有一个广播接收器能够收到这条广播消息，当这个广播接收器中的逻辑执行完毕后，广播才会继续传递，且优先级（priority）高的广播接收器会先收到广播消息。有序广播可以被接收器截断使得后面的接收器无法收到它；
  - 本地广播：仅在自己的应用内发送接收广播，也就是只有自己的应用能收到，数据更加安全，效率更高，但只能采用**动态注册**的方式；
  - 粘性广播：这种广播会**一直滞留**，当有匹配该广播的接收器被注册后，该接收器就会收到此条广播；

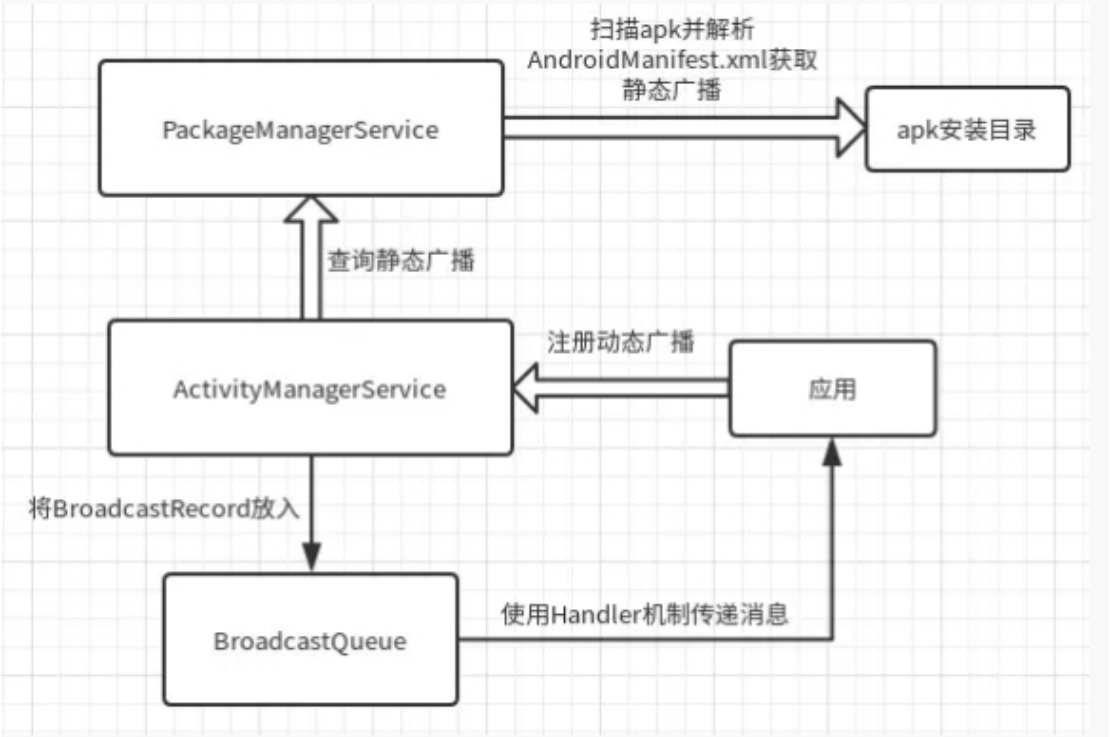
2、广播的两种注册方式 ？

- 参考回答：

注册方式	区别		
	使用方式	特点	应用场景
静态注册 （常驻广播）	在AndroidManifest.xml里通过<receiver>标签声明	• 常驻，不受任何组件的生命周期影响 （应用程序关闭后，如果有消息广播来，程序也们自会被系统调用） • 缺点：耗电、占内存	需要时刻监听广播
动态注册 （非常驻广播）	在代码中调用Context.registerReceiver（）方法	非常驻，灵活，跟随组件的生命周期变化 （组件结束=广播结束，在组件结束前，必须移除广播接收器）	需要特定时刻监听广播

3、广播发送和接收的原理了解吗 ？（Binder 机制、AMS）

- 参考回答：


扫描apk并解析
AndroidManifest.xml获取
静态广播
PackageManagerService
apk安装目录
查询静态广播
注册动态广播
ActivityManagerService
应用
将BroadcastRecord放入
BroadcastQueue
使用Handler机制传递消息