一、onStartCommand方法，返回START_STICKY

在运行onStartCommand后service进程被kill后，那将保留在开始状态，但是不保留那些传入的intent。不久后service就会再次尝试重新创建，因为保留在开始状态，在创建 service后将保证调用onstartCommand。如果没有传递任何开始命令给service，那将获取到null的intent。

【结论】 手动返回START_STICKY，亲测当service因内存不足被kill，当内存又有的时候，service又被重新创建，比较不错，但是不能保证任何情况下都被重建，比如进程被干掉了....

二、提升service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。

三、提升service进程优先级

Android中的进程是托管的，当系统进程空间紧张的时候，会依照优先级自动进行进程的回收。Android将进程分为6个等级,它们按优先级顺序由高到低依次是:

1. 前台进程(FOREGROUND_APP)
2. 可视进程(VISIBLE_APP)
3. 次要服务进程(SECONDARY_SERVER)
4. 后台进程 (HIDDEN_APP)
5. 内容供应节点(CONTENT_PROVIDER)
6. 空进程(EMPTY_APP)

当service运行在低内存的环境时，将会kill掉一些存在的进程。因此进程的优先级将会很重要，可以使用startForeground 将service放到前台状态。这样在低内存时被kill的几率会低一些。

在onStartCommand方法内添加如下代码:

```
Notification notification = new Notification(R.drawable.ic_launcher,
         getString(R.string.app_name), System.currentTimeMillis());

         PendingIntent pendingintent = PendingIntent.getActivity(this, 0,
         new Intent(this, AppMain.class), 0);
         notification.setLatestEventInfo(this, "uploadservice", "请保持程序在后台运
行",
         pendingintent);
         startForeground(0x111, notification);
```

注意在onDestroy里还需要stopForeground(true)，运行时在下拉列表会看到自己的APP在:

【结论】如果在极度极度低内存的压力下，该service还是会被kill掉，并且不一定会restart

四、onDestroy方法里重启service

直接在onDestroy（）里startService或service +broadcast 方式，就是当service走ondestory的时候，发送一个自定义的广播，当收到广播的时候，重新启动service;

【结论】当使用类似口口管家等第三方应用或是在setting里-应用-强制停止时，APP进程可能就直接被干掉了，onDestroy方法都进不来，所以还是无法保证~.~

五、监听系统广播判断Service状态

通过系统的一些广播，比如：手机重启、界面唤醒、应用状态改变等等监听并捕获到，然后判断我们的Service是否还存活，别忘记加权限啊。

六、将APK安装到/system/app，变身系统级应用