# Android内存泄漏

## 一．根本原因

内存泄漏的根本原因是：

生命周期长的对象，引用了生命周期短的对象。

在短命对象，使用完毕，正准备回收的时候，发现还有人再引用它，变不回收它了。

这里从两个方面讨论：

1. 长生命周期对象
2. 短生命周期对象

## 二.什么对象是长生命周期对象，还会经常持有别人的引用？

### 2.1单例模式

单例模式的对象，是静态的，生命周期基本同进程相同。

用的最多的是各种工具类，这类东西经常在初始化的时候，传入一个cotext，如果不巧这个context是一个Activity的话，那么那么在Activity执行onDestory之后，是不会被回收的。

这个就是第一个Activity不被释放，不会反复增加内存。

### 2.2系统服务

广播监听器，Activity在onDestory的时候，不把释放掉注册的广播监听器……

这个主要是由于，register的时候要传入一个context，这个context一般都是当前Activity，那这个context被系统服务引用，内存当然释放不掉。

如果只在onCreate里面调用了register，在onDestory里没有unRegister，

那么在反复进入Activity的时候，会造成所有的Activity都不会释放，导致oom。

### 2.3消息循环Loop

这个主要针对，被post到loop的runnable。

## 三 什么对象是短生命周期对象，还会经常被别人的引用？

### 3.1 Activity

Activity又占资源，又有明显的生命周期，对它的管理，要格外注意。

自己在Activity里面写一个静态变量，然后引用自己的情况，好像不多。

经常被当成context参数传来传去，不巧被哪个静态对象引用了，就释放不掉了。

### 3.2 Activity的内部类

内部类持有外部类的引用，如果一个Activity的内部类，被静态对象持有引用，那也会导致Activity不释放

### 3.3 Activity的匿名内部类

匿名的listener，runnable，这个太常见了。

### 3.4 持有Activity引用的对象

比如一个本地对象button，它在初始化的时候，是需要传入一个context的，

比如在Activity1里面调用findViewById，这个context就是Activity1

如果这个button被静态对象持有，也会导致Activity1不被释放.

### 3.5 Activity的成员变量

这个好像不用多说，持有Activity1成员变量的前提就是，持有Activity1的引用。