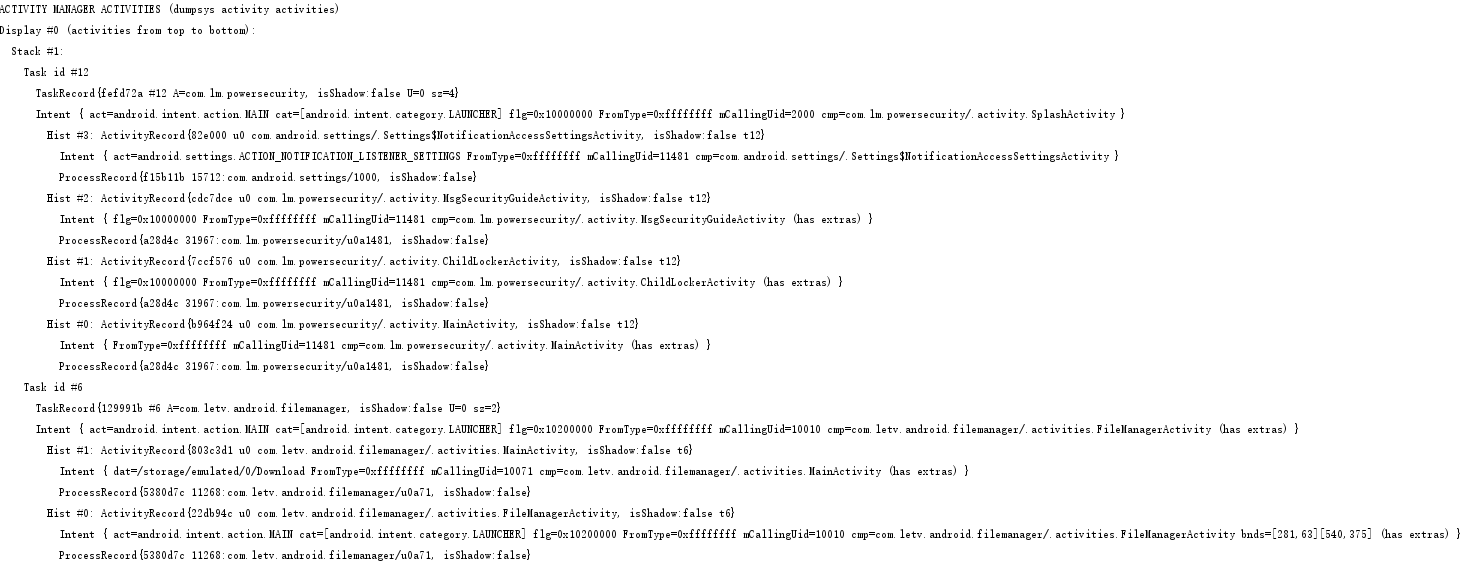
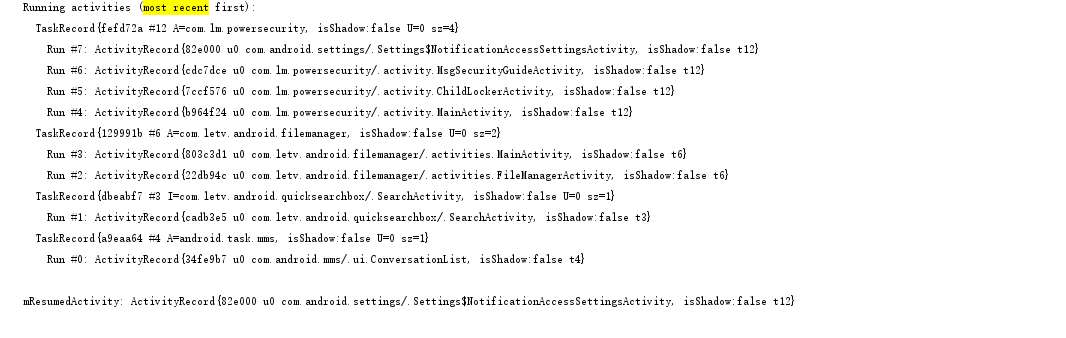
Effective Launch Mode & Task Affinity（21条）

# Launch Mode类型

1. [任务栈](https://developer.android.com/guide/components/tasks-and-back-stack.html" \l "LaunchModes)是一种“后进先出”的栈结构。
2. Android系统通过任务栈管理各个activity间的回退和跳转。
3. 某个activity的任务栈，由参数task affinity（任务相关性）指定，默认情况下，所有activity的任务栈的名字为应用的包名。注意：task affinity属性需要和singletask启动模式或allowTaskReparenting属性配对使用，否则无意义。
4. 运行命令“adb shell dumpsys activity”查看任务栈。搜索关键字“activities from top to bottom”，定位任务栈内容。

* Hist，代表此task中的activity记录。
* Stack #0 ，保存Launcher相关的activity的Task。
* Stack #1，保存除Launcher外的其他应用的activity组成的task。Task的划分，从用户触摸桌面图标开始。
* TaskRecord里面：sz，表示此任务下Hist的总数。
* mResumedActivity，当前可见的activity。





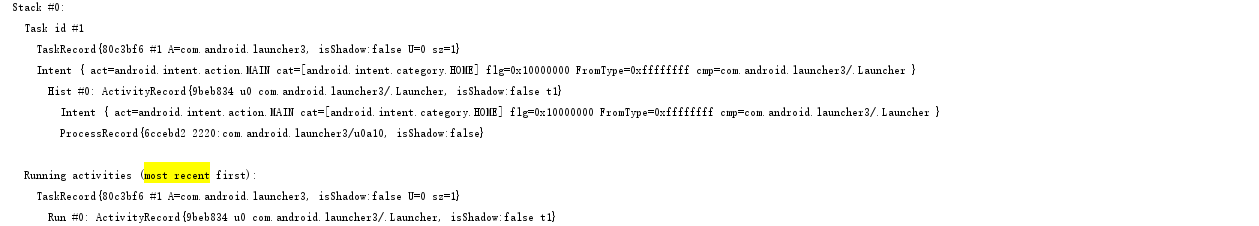


图1

## Single Task，栈内复用模式

1. 此启动模式默认具有FLAG\_ACTIVITY\_CLEAR\_TOP特性，即如果activity a是single task模式，被启动后，栈内顶部的activity被清出栈。
2. 某activity设置了此启动模式，多次启动不会重建实例，会调用onNewIntent。例如：2中，此时activity Y以singletask模式请求启动，其所需任务栈为B（如果未被task affinity指定，那么默认为包名），如果activity Y和任务栈B均不存在，系统会创建任务栈B，然后创建activity Y的实例，并压入栈B；如果任务栈B存在，且情况是XYZ，那么栈顶activity Z会出栈，启动后栈B情况是XY。
3. 在启动的时候设置FLAG\_ACTIVITY\_NEW\_TASK和在XML中指定该启动模式效果相同。
4. 任务栈分为前台任务栈和后台任务栈，如果 Activity 已经是某个拥有自己的返回栈的后台任务的一部分，则整个返回栈也会上移到当前任务的顶部。后台任务栈中的activity处于停止状态。切换过程如图1所示。

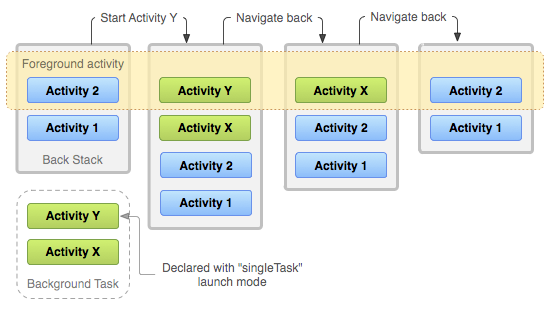


图2

## Standard，标准模式

1. 默认启动模式，每次启动都会创建一个新的activity实例，各个实例可以在一个任务栈内，也可属于不同任务栈。
2. 如果Activity X启动了activity Y，如果Y是标准模式，那么Y在X所属的任务栈内。附注：当以ApplicationContext启动activity Y（标准模式），会报错：“Calling startActivity from outside of an activity context requires the FLAG\_ACTIVITY\_NEW\_TASK flag. Is the really what you want?”，需要设置FLAG\_ACTIVITY\_NEW\_TASK为Y创建一个新的任务栈。

## Single Top，栈顶复用模式

1. 如果被启动的Activity已经位于当前任务栈的栈顶，则不会重新创建，同时onNewIntent会被调用。如果不在栈顶，则创建新的实例。举例：当前栈情况为WXYZ，其中Z位于栈顶，此时再次启动Z，如果Z的启动模式是singleTop，那么启动后栈内情况为WXYZ，情况不变；如果为standard，Z会被重新创建，栈情况为WXYZ。
2. 在启动的时候设置FLAG\_ACTIVITY\_SINGLE\_TOP和在XML中指定该启动模式效果相同。

## SingleInstance，单实例模式

1. 是一种加强的singleTask模式，即具有singleTask所有的特性。
2. 此模式会使得启动的activity单独位于一个任务栈，只要此任务栈存在，就永远不会创建新的实例。如图3所示，即便affinity配置为相同的值，不同activity仍在不同的task中。

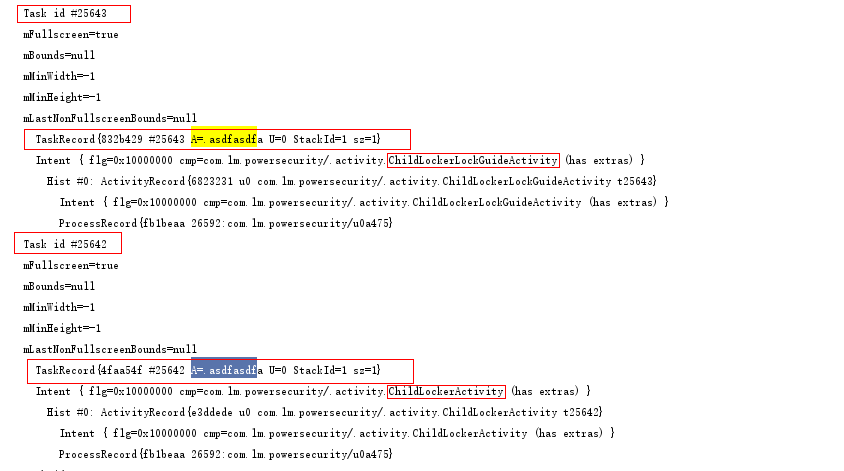


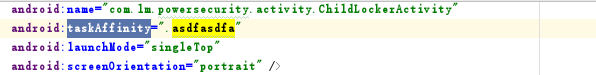
图3

## 总结

1. 只有SingleTask和SingleInstance受taskAffinity影响，即指定taskAffinity的activity必定运行在指定的任务栈中，而standard和singleTop启动模式的activity运行在启动此activity所在的任务栈中。
2. onNewIntent值在以standard模式启动的activity中不会调用，因为每次都是启动一个新的实例。
3. 当设置FLAG\_ACTIVITY\_CLEAR\_TOP后，同一个任务站内的且位于被启动activity上面的activity都要出栈。

# Task Affinity

1. 必须singleTask启动模式或allowTaskReparenting配合使用，否则此属性无意义。如图4所示，taskAffinity并没有起作用。



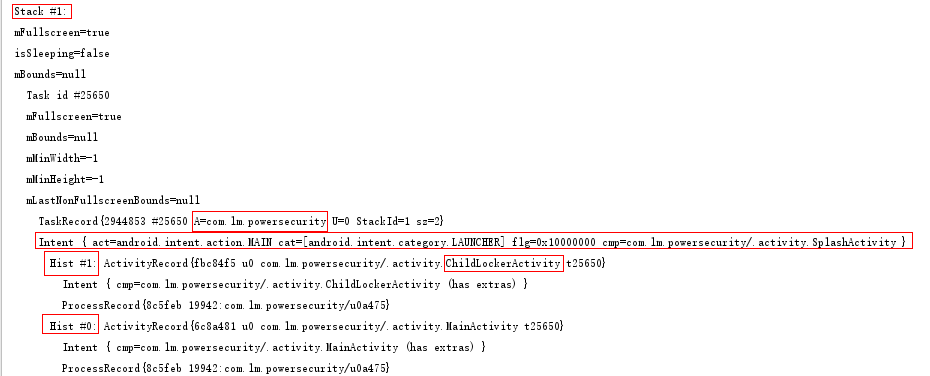


图4

1. 和allowTaskReparenting结合使用，会产生转移所在task的效果。举例：比如有一个天气预报程序，它有一个用于显示天气信息的Activity，allowTaskReparenting属性设置成true，这个Activity和天气预报程序的所有其它Activity具体相同的affinity值。这个时候，你自己的应用程序通过Intent去启动了这个用于显示天气信息的Activity，那么此时这个Activity应该是和你的应用程序是在同一个任务当中的。但是当把天气预报程序切换到前台的时候，这个Activity会被转移到天气预报程序的任务当中，并显示出来。如果将你自己的应用切换到前台，发现你自己应用Task里的那个Activity消失了。
2. 和singleTask使用，待启动的activity运行在task affinity指定的栈内。

# 清理返回栈

1. 如果用户长时间离开任务栈，系统会清除所有的Activity任务，根activity（即最后一次点击桌面，弹出的那个activity）除外，当用户再次返回任务栈，仅恢复根activity；如果在根activity设置“[alwaysRetainTaskState](https://developer.android.com/guide/topics/manifest/activity-element.html" \l "always)”属性为true，在很长一段时间，任务将所有activity保留栈中。如果设置“[clearTaskOnLaunch](https://developer.android.com/guide/topics/manifest/activity-element.html" \l "clear)”为true，则每次离开任务，再次回到该任务，只能回到根activity；如果设置“[finishOnTaskLaunch](https://developer.android.com/guide/topics/manifest/activity-element.html" \l "finish)”，离开任务则清理activity。