1.ADJ 如何查看

利用adb shell

```
1.ps | grep 包名  //查看当前app的进程号

2.cat /proc/进程号/oom_adj  //查看当前进程的adj值(早期android和linux使用，现已废弃，但仍然有效)

3.cat /proc/进程号/oom_score_adj     //这个是新版本的查看adj的命令，adj有效值为-1000~1000
```

2.ADJ的值的各种含义

ADJ级别	取值	含义
NATIVE_ADJ	-1000	native进程
SYSTEM_ADJ	-900	仅指system_server进程
PERSISTENT_PROC_ADJ	-800	系统persistent进程
PERSISTENT_SERVICE_ADJ	-700	关联着系统或persistent进程
FOREGROUND_APP_ADJ	0	前台进程
VISIBLE_APP_ADJ	100	可见进程
PERCEPTIBLE_APP_ADJ	200	可感知进程，比如后台音乐播放
BACKUP_APP_ADJ	300	备份进程
HEAVY_WEIGHT_APP_ADJ	400	重量级进程
SERVICE_ADJ	500	服务进程(A list中的service)
HOME_APP_ADJ	600	Home进程
PREVIOUS_APP_ADJ	700	上一个进程
SERVICE_B_ADJ	800	B List中的Service
CACHED_APP_MIN_ADJ	900	不可见进程的adj最小值
CACHED_APP_MAX_ADJ	906	不可见进程的adj最大值

3.ADJ触发顺序

ADJ是一种算法，用于系统判断进程优先级以触发Linux的LMK（LowMemoryKill）机制。一般触发时机（Linux下）是在系统低内存时，为了维护正在运行的进程，杀掉优先级比较低（adj值比较高）的其他进程。

在Android中这一机制有所改动。在ActivityManagerService里有具体的计算Adj值的源码。

进程刚启动时ADJ等于INVALID_ADJ，当执行完attachApplication()，该该进程的curAdj和setAdj不相等，则会触发执行setOomAdj()将该进程的节点/proc/pid/oom_score_adj写入oomadj值。下图参数为Android原生阈值，当系统剩余空闲内存低于某阈值(比如147MB)，则从ADJ大于或等于相应阈值(比如900)的进程中，选择ADJ值最大的进程，如果存在多个ADJ相同的进程，则选择内存最大的进程。 如下是64位机器，LMK默认阈值图：

```
----------ADJ----------------Memory Left------------
FOREGROUND_APP_ADJ(0)          73MB
VISIBLE_APP_ADJ(100)           92MB
PERCEPTIBLE_APP_ADJ(200)       110MB
BACKUP_APP_ADJ(300)            129MB
CACHED_APP_MIN_ADJ(900)        221MB
CACHED_APP_MAX_ADJ(906)        332MB
```

4.高级进程 ADJ<0的进程

1.NATIVE_ADJ(-1000)：是由init进程fork出来的Native进程，并不受system管控；

2.SYSTEM_ADJ(-900)：是指system_server进程；

3.PERSISTENT_PROC_ADJ(-800): 是指在AndroidManifest.xml中申明android:persistent=”true”的系统(即带有FLAG_SYSTEM标记)进程，persistent进程一般情况并不会被杀，即便被杀或者发生Crash系统会立即重新拉起该进程。

4.PERSISTENT_SERVICE_ADJ(-700)：是由startIsolatedProcess()方式启动的进程，或者是由system_server或者persistent进程所绑定(并且带有BIND_ABOVE_CLIENT或者BIND_IMPORTANT)的服务进程

5.总结

Android进程优先级ADJ的每一个ADJ级别往往都有多种场景，使用adjType完美地区分相同ADJ下的不同场景； 不同ADJ进程所对应的schedGroup不同，从而分配的CPU资源也不同，schedGroup大体分为TOP(T)、前台(F)、后台(B)； ADJ跟AMS中的procState有着紧密的联系。

1.adj：通过调整oom_score_adj来影响进程寿命(Lowmemorykiller杀进程策略)；

2.schedGroup：影响进程的CPU资源调度与分配；

3.procState：从进程所包含的四大组件运行状态来评估进程状态，影响framework的内存控制策略。比如控制缓存进程和空进程个数上限依赖于procState，再比如控制APP执行handleLowMemory()的触发时机等。 为了说明整体关系，以ADJ为中心来讲解跟adjType,schedGroup,procState的对应关系，下面以一幅图来诠释整个ADJ算法的精髓，几乎涵盖了ADJ算法调整的绝大多数场景。

ADJ	adjType	schedGroup	procState	解读
maxAdj<=0	pers-top-activity	T	PER (0)	当Activity处于resumed状态
	pers-top-ui	T	PER (0)	当非activity的UI位于屏幕最顶层
	fixed	T / F	PER (0) / PERU (1)	固定maxAdj<=0，则至少为F
fore(0)	top-activity	T	TOP (2)	当Activity处于resumed状态
	instrumentation	F	FGS (3)	通过startInstrumentation()启动的进程
	broadcast	F / B	RCVR (10)	正执行onReceive()，F/B 取决于前/后台广播队列
	exec-service	F / B	SVC (9)	正执行Service回调方法，F/B 取决于前/后台进程发起的服务
	service	T / F	SVC (9)	可见activity进程采用BIND_ADJUST_WITH_ACTIVITY方式绑定服务
	ext-provider	F	IMPF(5)	调用getContentProviderExternal()方法
	top-sleeping	B	TPSL (11)	设备处于sleeping状态
vis(100)	vis-activity	F		当Activity处于visible状态
prcp(200)	pause-activity	F		当Activity处于PAUSING、PAUSED状态
	fg-service		FGS (3)	执行startForegroundService()方法的前台服务进程
	has-overlay-ui		IMPF (5)	非activity的UI位于屏幕最顶层
	force-imp		TRNB(7)	执行setProcessImportant()方法，例如弹出Toast
	stop-activity	B	LAST (14)	当Activity处于STOPPING状态
bkup(300)	backup	B	BKUP (8)	执行bindBackupAgent()方法的备份进程
hvy(400)	heavy		HVY (12)	privateFlags带PRIVATE_FLAG_CANT_SAVE_STATE标识的应用
svc(500)	started-services		SVC (9)	当没有启动过Activity，且30分钟内活跃过的服务进程
home(600)	home		HOME (13)	类型为ACTIVITY_TYPE_HOME的桌面应用
prev(700)	previous		LAST (14)	上一个被使用的activity进程
	recent-provider		LAST (14)	20s内刚被使用的provider进程
svcb(800)	started-services		SVC (9)	由A Service转换而来
cch(900)	cch-started-services	B		超过30min没有活跃过的服务进程
	cch-started-ui-services			启动过Activity的服务进程
	cch-act		CAC (15)	带有activity的cache进程
	cch-as-act		CAC (15)	`treatLikeActivity`
	cch-client-act		CACC (16)	`hasClientActivities`
	cch-rec		CRE (17)	`recentTasks`
	cch-empty		CEM (18)	空进程
clientAdj	cch-bound-ui-services	B		启动过activity的服务进程，且clientAdj重要性低于perceptible
	service	F / B		根据clientAdj和flags来决定该进程的adj
	cch-ui-provider	B		启动过activity的provider进程，且clientAdj重要性低于perceptible
	provider	F / B		根据clientAdj来决定该进程的adj

image

6.最后，开发时应注意：

1.UI进程与Service进程一定要分离，因为对于包含activity的service进程，一旦进入后台就成为"cch-started-ui-services"类型的cache进程(ADJ>=900)，随时可能会被系统回收；而分离后的Service进程服务属于SERVICE_ADJ(500)，被杀的可能性相对较小。尤其是系统允许自启动的服务进程必须做UI分离，避免消耗系统较大内存。 只有真正需要用户可感知的应用，才调用startForegroundService()方法来启动前台服务，此时ADJ=PERCEPTIBLE_APP_ADJ(200)，常驻内存，并且会在通知栏常驻通知提醒用户，比如音乐播放，地图导航。切勿为了常驻而滥用前台服务，这会严重影响用户体验。

2.进程中的Service工作完成后，务必主动调用stopService或stopSelf来停止服务，避免占据内存，浪费系统资源；

3.不要长时间绑定其他进程的service或者provider，每次使用完成后应立刻释放，避免其他进程常驻于内存；

4.APP应该实现接口onTrimMemory()和onLowMemory()，根据TrimLevel适当地将非必须内存在回调方法中加以释放。当系统内存紧张时会回调该接口，减少系统卡顿与杀进程频次。

5.减少在保活上花心思，更应该在优化内存上下功夫，因为在相同ADJ级别的情况下，系统会选择优先杀内存占用的进程。