**内存泄露监控工具**

启动工具： tool.bat

目前监控的数据为：

1：dumpsys meminfo pkg

2：cat /proc/meminfo

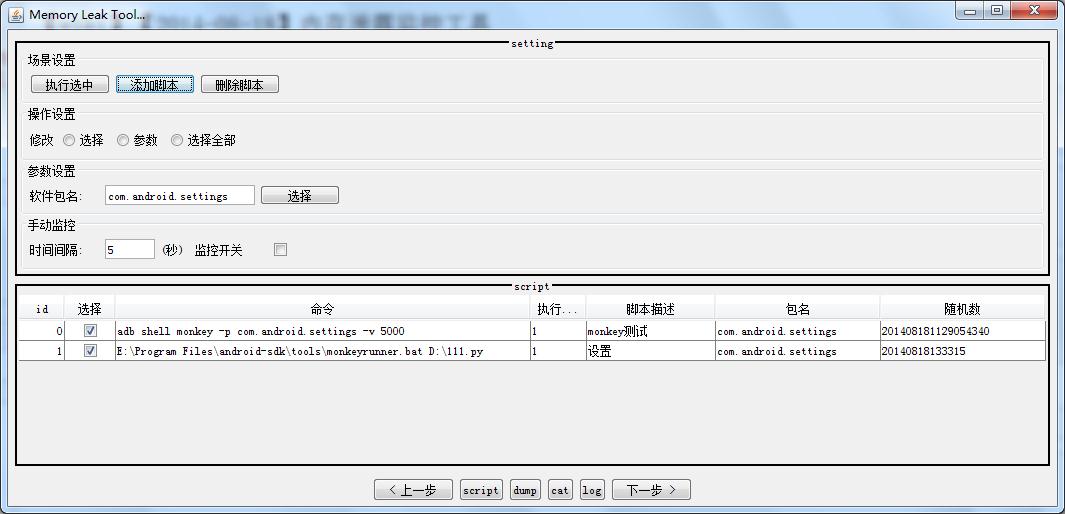
使用说明：

1：设置监控时间间隔，软件包名，打开监控开关，直接监控手机某一个app的内存情况.关闭监控开关停止执行

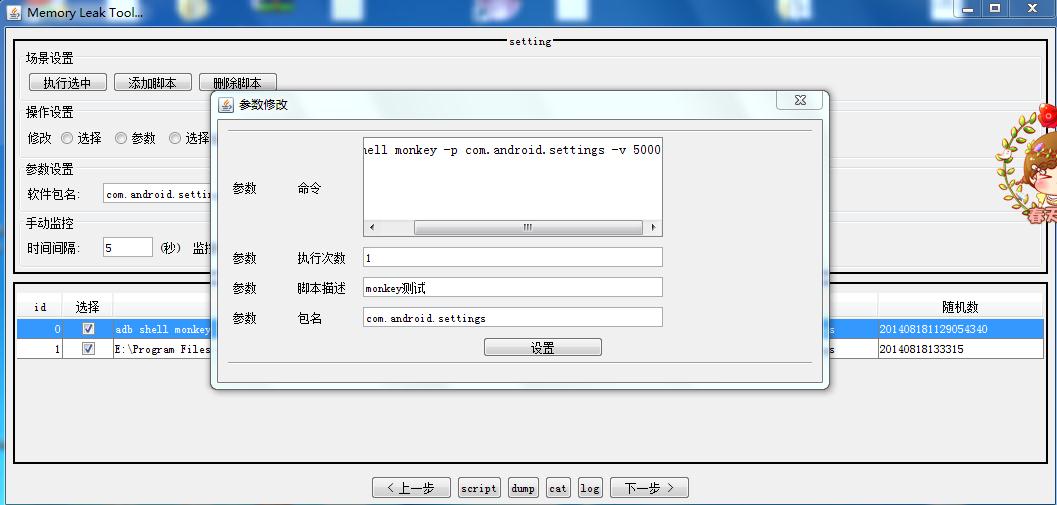
2：当执行某一命令的时候，监控某一app的内存变化情况

命令可以是monkey命令，或者某一个脚本。

比如要在执行monkey测试的过程中监控内存变化情况

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Leak001.jpg)

选中某一行。双击。进入设置对话框

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Leak002.jpg)

选中某一行， 按下 ctrl+l 可以进入测试数据以及log 文件夹

[](http://10.120.10.100:9002/%E6%96%87%E4%BB%B6:Leak003.jpg)

执行过程中不要去打开data.xlsx,这样会导致文件被加密,写入数据会异常...

Android 查看内存使用情况

查看应用程序的命令：adb shell procrank

adb shell cat /proc/meminfo

显示如下：

PID Vss Rss Pss Uss cmdline

190 79560K 74736K 49624K 43604K system\_server

14335 55776K 55740K 31308K 26676K com.android.launcher2

13074 47408K 47380K 24947K 22428K com.android.settings

7626 42060K 42028K 21312K 18432K com.android.systemui

13948 32992K 32944K 11687K 9800K android.process.acore

283 25516K 25476K 8136K 7148K com.android.phone

422 24560K 24524K 7338K 6292K com.android.inputmethod.pinyin

12871 25804K 25740K 6288K 4880K com.snda.tt

454 23672K 23616K 5735K 4544K com.google.process.gapps

14723 25744K 25684K 5706K 4100K com.rili.android.client

14844 23816K 23752K 4706K 3276K com.dopid.android

14601 23980K 23920K 4686K 3432K android.process.media

13060 22892K 22828K 4570K 3428K com.tencent.WBlog

13303 22876K 22820K 4013K 2852K com.android.vending

13396 21672K 21608K 3978K 2960K com.limitfan.gojuuon

93 4132K 4112K 3322K 3164K /system/bin/mediaserver

603 18732K 18668K 3274K 2476K com.mediatek.mdlogger

420 18676K 18616K 3033K 2220K com.mediatek.bluetooth

14734 21744K 21680K 3010K 1868K com.yingyonghui.market

14829 20792K 20728K 3002K 1924K com.uucun105286.android.cms

13323 20312K 20248K 2851K 1848K com.cootek.smartinputv5

92 20008K 19932K 2680K 1556K zygote

111 2008K 2004K 1438K 1396K /system/bin/drmserver

105 1852K 1848K 1355K 1324K /system/bin/em\_svr

108 1512K 1504K 1100K 1080K /system/bin/mdpd

7696 816K 812K 501K 468K /system/bin/wpa\_supplicant

167 780K 776K 429K 416K /system/bin/rild

96 560K 556K 237K 228K /system/bin/mtkbt

84 484K 480K 234K 224K /system/bin/vold

85 576K 572K 229K 216K /system/bin/netd

1 284K 284K 210K 172K /init

103 424K 420K 200K 196K /system/bin/mtk\_agpsd

150 420K 416K 192K 188K /system/bin/gsm0710muxd

151 1036K 508K 181K 172K /system/bin/mdlogger

91 516K 512K 173K 164K /system/bin/dm\_agent\_binder

110 184K 184K 164K 164K /sbin/adbd

80 388K 384K 162K 156K /system/bin/aee\_aed

106 496K 492K 161K 152K /system/bin/nvram\_agent\_binder

14682 416K 412K 141K 132K /system/bin/dhcpcd

14873 356K 348K 139K 132K /system/bin/sh

88 372K 368K 136K 128K /system/bin/hald

100 368K 364K 129K 124K /system/bin/installd

99 340K 336K 120K 116K /system/bin/atci

87 348K 344K 116K 112K /system/bin/mobile\_log\_d

64 152K 152K 114K 76K /sbin/ueventd

7689 340K 336K 110K 104K /system/bin/debuggerd

86 324K 320K 109K 104K /system/bin/netdiag

353 316K 312K 96K 92K /system/xbin/mnld

94 284K 276K 92K 88K /system/bin/dbus-daemon

146 272K 268K 92K 88K /system/bin/memsicd

97 368K 276K 87K 84K /system/bin/ccci\_fsd

102 268K 260K 87K 84K /system/bin/keystore

14681 312K 304K 81K 72K /system/bin/logwrapper

83 256K 248K 75K 72K /system/bin/servicemanager

95 252K 244K 75K 72K /system/bin/6620\_launcher

7694 272K 264K 73K 68K /system/bin/logwrapper

109 232K 224K 66K 64K /system/bin/sh

98 236K 228K 59K 56K /system/bin/ccci\_mdinit

MemTotal: 所有可用RAM大小 （即物理内存减去一些预留位和内核的二进制代码大小）

MemFree: LowFree与HighFree的总和，被系统留着未使用的内存

Buffers: 用来给文件做缓冲大小

Cached: 被高速缓冲存储器（cache memory）用的内存的大小（等于 diskcache minus SwapCache ）.

SwapCached:被高速缓冲存储器（cache memory）用的交换空间的大小

已经被交换出来的内存，但仍然被存放在swapfile中。用来在需要的时候很快的被替换而不需要再次打开I/O端口。

Active: 在活跃使用中的缓冲或高速缓冲存储器页面文件的大小，除非非常必要否则不会被移作他用.

Inactive: 在不经常使用中的缓冲或高速缓冲存储器页面文件的大小，可能被用于其他途径.

HighTotal:

HighFree: 该区域不是直接映射到内核空间。内核必须使用不同的手法使用该段内存。

LowTotal:

LowFree: 低位可以达到高位内存一样的作用，而且它还能够被内核用来记录一些自己的数据结构。Among many

other things, it is where everything from the Slab is

allocated. Bad things happen when you're out of lowmem.

SwapTotal: 交换空间的总大小

SwapFree: 未被使用交换空间的大小

Dirty: 等待被写回到磁盘的内存大小。

Writeback: 正在被写回到磁盘的内存大小。

AnonPages：未映射页的内存大小

Mapped: 设备和文件等映射的大小。

Slab: 内核数据结构缓存的大小，可以减少申请和释放内存带来的消耗。

SReclaimable:可收回Slab的大小

SUnreclaim：不可收回Slab的大小（SUnreclaim+SReclaimable＝Slab）

PageTables：管理内存分页页面的索引表的大小。

NFS\_Unstable:不稳定页表的大小

注：

VSS - Virtual Set Size 虚拟耗用内存（包含共享库占用的内存）

RSS - Resident Set Size 实际使用物理内存（包含共享库占用的内存）

PSS - Proportional Set Size 实际使用的物理内存（比例分配共享库占用的内存）

USS - Unique Set Size 进程独自占用的物理内存（不包含共享库占用的内存）

一般来说内存占用大小有如下规律：VSS >= RSS >= PSS >= USS

查看单个应用程序内存占用量的命令：adb shell dumpsys meminfo $包名 或者 $进程号 //当然你也可以adb shell进去后在输入余下部分。

android程序内存被分为2部分：native和dalvik，dalvik就是我们平常说的java堆，我们创建的对象是在这里面分配的，

而bitmap是直接在native上分配的，对于内存的限制是 native+dalvik 不能超过最大限制。android程序内存一般限制在16M，当然也有24M的

显示如下：

Applications Memory Usage (kB):

Uptime: 70102403 Realtime: 104776333

\*\* MEMINFO in pid 14335 [com.android.launcher2] \*\*

native dalvik other total

size: 27576 8455 N/A 36031

allocated: 26850 5311 N/A 32161

free: 385 3144 N/A 3529

(Pss): 13830 1134 16034 30998

(shared dirty): 2980 1932 13776 18688

(priv dirty): 13736 184 12504 26424

Objects

Views: 0 ViewRoots: 0

AppContexts: 0 Activities: 0

Assets: 3 AssetManagers: 3

Local Binders: 162 Proxy Binders: 81

Death Recipients: 1

OpenSSL Sockets: 0

SQL

heap: 3063 MEMORY\_USED: 3063

PAGECACHE\_OVERFLOW: 2266 MALLOC\_SIZE: 50

DATABASES

pgsz dbsz Lookaside(b) Dbname

1 2544 500 launcher.db

/\*\* The proportional set size for dalvik. \*/

public int dalvikPss;

/\*\* The private dirty pages used by dalvik. \*/

public int dalvikPrivateDirty;

/\*\* The shared dirty pages used by dalvik. \*/

public int dalvikSharedDirty;

/\*\* The proportional set size for the native heap. \*/

public int nativePss;

/\*\* The private dirty pages used by the native heap. \*/

public int nativePrivateDirty;

/\*\* The shared dirty pages used by the native heap. \*/

public int nativeSharedDirty;

/\*\* The proportional set size for everything else. \*/

public int otherPss;

/\*\* The private dirty pages used by everything else. \*/

public int otherPrivateDirty;

/\*\* The shared dirty pages used by everything else. \*/

public int otherSharedDirty;

"size" 表示的是总内存大小（kb）。,

"allocated" 表示的是已使用了的内存大小（kb），,

"free"表示的是剩余的内存大小（kb）,

dalvik：是指dalvik所使用的内存。

native：是被native堆使用的内存。应该指使用C\C++在堆上分配的内存。

other:是指除dalvik和native使用的内存。但是具体是指什么呢？至少包括在C\C++分配的非堆内存，比如分配在栈上的内存。puzlle!

private:是指私有的。非共享的。

share:是指共享的内存。

PSS：实际使用的物理内存（比例分配共享库占用的内存）

Pss：它是把共享内存根据一定比例分摊到共享它的各个进程来计算所得到进程使用内存。网上又说是比例分配共享库占用的内存，那么至于这里的共享是否只是库的共享，还是不清楚。

PrivateDirty：它是指非共享的，又不能换页出去（can not be paged to disk ）的内存的大小。比如Linux为了提高分配内存速度而缓冲的小对象，即使你的进程结束，该内存也不会释放掉，它只是又重新回到缓冲中而已。

SharedDirty:参照PrivateDirty我认为它应该是指共享的，又不能换页出去（can not be paged to disk ）的内存的大小。比如Linux为了提高分配内存速度而缓冲的小对象，即使所有共享它的进程结束，该内存也不会释放掉，它只是又重新回到缓冲中而已。

有了这些，调试应用程序的内存问题容易多了

其中size是需要的内存，

而allocated是分配了的内存，

对应的2列分别是native和dalvik，

当总数也就是total这一列超过单个程序内存的最大限制时，

OOM就很有可能会出现了。