

Android 稳定性测试框架

Monkey 简要使用说明

【附录中包含 Monkey 部分命令和脚本语言】

By [超级闹钟小组] 王磊 沙金锐

◎ 安装 Monkey

Monkey 包括在 Android SDK 中，安装 Android SDK 后直接使用（Android SDK 的安装自行百度），官方文档在 Android SDK 文件夹下\docs\tools\help\monkey.html

◎ 环境配置

将 AndroidSDK 文件夹下的 platform-tools 文件夹加入环境变量 path

◎ 建立测试

1. Monkey 的基本使用

使用前请保证手机或安卓模拟器已与电脑连接。

进入命令行，输入 `adb shell monkey <time_event>` (其中 time_event 表示随机事件数)

结果如下，同时会看到你的手机已经自己运行起来，胡乱打开了某些 APP，这表示命令行向手机发送了随机操作。

```
C:\Users\dphs>adb shell monkey 50
// activityResuming(com.qihoo.cleandroid_cn)
Events injected: 50
## Network stats: elapsed time=1479ms (0ms mobile, 0ms wifi, 1479ms not connecte
d)
```

2. Monkey 的脚本测试

注：以下内容截图中的命令 `nox_adb` 是由于使用夜神安卓模拟器所改写的，手机真机或其他模拟器一般只要 `adb` 即可

№1 先看两个基本命令

① 运行 Monkey 脚本

```
adb shell monkey -f <scriptfile> <event-count>
```

<scriptfile>表示脚本文件名，<event-count>表示脚本运行次数

② 打印日志信息

```
adb shell monkey -v -v -v <time_event>
```

注：其中-v 的个数代表信息的详细程度，逐级详细，但最多支持三级

Eg. -v 的情况

```
D:\software\Nox\bin>nox_adb shell monkey -v 10
Monkey: seed=1481409606081 count=10
IncludeCategory: android.intent.category.LAUNCHER
IncludeCategory: android.intent.category.MONKEY
// Event percentages:
// 0: 15.0%
// 1: 10.0%
// 2: 2.0%
// 3: 15.0%
// 4: -0.0%
// 5: 25.0%
// 6: 15.0%
// 7: 2.0%
// 8: 2.0%
// 9: 1.0%
// 10: 13.0%
Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.vphone.helper/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.vphone.helper/.MainActivity } in package com.vphone.helper
:Sending Touch (ACTION_DOWN): 0:(774.0,501.0)
:Sending Touch (ACTION_UP): 0:(759.9808,521.2685)
Events injected: 10
:Sending rotation degree=0, persist=false
:Dropped: keys=0 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=1385ms (0ms mobile, 0ms wifi, 1385ms not connected)
// Monkey finished
```

Eg. 2 个 -v 的情况（明显复杂了好多）

```
D:\software\Nox\bin>nox_adb shell monkey -v -v 10
Monkey: seed=1481409887946 count=10
IncludeCategory: android.intent.category.LAUNCHER
IncludeCategory: android.intent.category.MONKEY
// Selecting main activities from category android.intent.category.LAUNCHER
// + Using main activity com.android.contacts.activities.PeopleActivity (from package com.android.contacts)
// + Using main activity com.google.android.gms.app.settings.GoogleSettingsActivity (from package com.google.android.gms)
// + Using main activity com.android.vending.AssetBrowserActivity (from package com.android.vending)
// + Using main activity com.android.settings.Settings (from package com.android.settings)
// + Using main activity com.bignox.app.store.hd.StoreActivity (from package com.bignox.app.store.hd)
// + Using main activity com.android.browser.BrowserActivity (from package com.android.browser)
// + Using main activity com.cyanogenmod.filemanager.activities.NavigationActivity (from package com.cyanogenmod.filemanager)
// + Using main activity com.android.gallery3d.app.GalleryActivity (from package com.android.gallery3d)
// + Using main activity com.android.camera.Camera (from package com.android.camera)
// + Using main activity com.vphone.launcher.Launcher (from package com.vphone.launcher)
// + Using main activity com.android.providers.downloads.ui.DownloadList (from package com.android.providers.downloads.ui)
// + Using main activity com.google.android.gms.games.ui.destination.main.MainActivity (from package com.google.android.play.games)
// + Using main activity com.vphone.helper.MainActivity (from package com.vphone.helper)
// + Using main activity superalarm.firsttry.MainActivity (from package superalarm.firsttry)
// + Using main activity com.tencent.qqmusic.activity.AppStarterActivity (from package com.tencent.qqmusic)
// + Using main activity com.tencent.sc.activity.SplashActivity (from package com.qzone)
// + Using main activity com.tencent.mobileqq.activity.SplashActivity (from package com.tencent.mobileqq)
// + Using main activity com.tencent.mm.ui.LauncherUI (from package com.tencent.mm)
// + Using main activity com.t2kports.nba2k14android.MainActivity (from package com.t2kports.nba2k14android)
// Selecting main activities from category android.intent.category.MONKEY
// + Using main activity com.android.settings.Settings$RunningServicesActivity (from package com.android.settings)
// + Using main activity com.android.settings.Settings$StorageUseActivity (from package com.android.settings)
// + Using main activity com.vphone.launcher.Launcher (from package com.vphone.launcher)
// + Using main activity com.vphone.launcher.Launcher (from package com.vphone.launcher)
Seeded: 1481409887946
Event percentages:
0: 15.0%
1: 10.0%
2: 2.0%
3: 15.0%
4: -0.0%
5: 25.0%
6: 15.0%
7: 2.0%
8: 2.0%
9: 1.0%
10: 13.0%
Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.android.providers.downloads.ui.DownloadList;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.android.providers.downloads.ui.DownloadList } in package com.android.providers.downloads.ui
Sleeping for 0 milliseconds
:Sending Key (ACTION_DOWN): 20 // KEYCODE_DPAD_DOWN
Sleeping for 0 milliseconds
// allowing start of Intent { act=android.provider.action.MANAGE_ROOT dat=content://com.android.providers.downloads.documents/root/downloads cap=com.android.documentsui/DocumentsActivity } in package com.android.documentsui
// activityResuming(com.android.documentsui)
:Sending Key (ACTION_UP): 20 // KEYCODE_DPAD_DOWN
Sleeping for 0 milliseconds
:Sending Key (ACTION_DOWN): 184 // KEYCODE_PROG_GREEN
:Sending Key (ACTION_UP): 184 // KEYCODE_PROG_GREEN
Sleeping for 0 milliseconds
:Sending Key (ACTION_DOWN): 20 // KEYCODE_DPAD_DOWN
:Sending Key (ACTION_UP): 20 // KEYCODE_DPAD_DOWN
Sleeping for 0 milliseconds
:Sending Key (ACTION_DOWN): 109 // KEYCODE_BUTTON_SELECT
:Sending Key (ACTION_UP): 109 // KEYCODE_BUTTON_SELECT
Sleeping for 0 milliseconds
:Sending Touch (ACTION_DOWN): 0:(51.0,317.0)
Events injected: 10
:Sending rotation degree=0, persist=false
:Dropped: keys=0 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=486ms (0ms mobile, 0ms wifi, 486ms not connected)
// Monkey finished
```

No2 Monkey 脚本编写与运行

①脚本前缀

```
#Start Script
type = user
count = 10
speed = 1.0
start data >>|
```

脚本前五行固定格式如上，**start data >>**中间空格不能省略（教训啊啊啊）其中 **type**、**count**、**speed** 的值改变对结果并无影响。

② 从例子看起

```
#此部分为注释
#下面所有语句分号可加可不加
#monkey脚本无格式要求，txt、java等甚至无格式均可
#脚本所用机型分辨率为1280x720

#Start Script
type = user
count = 10
speed = 1.0
start data >>    #以上为脚本前缀部分

#打开superalarm.firsttry包内的主界面MainActivity
LaunchActivity(superalarm.firsttry,superalarm.firsttry.MainActivity);
UserWait(2000);    #等待2000ms，使机器完全反应

#模拟手指点击操作，点击坐标为（623,1212）
Tap(623,1212);
UserWait(2000);
Tap(623,1212);
UserWait(2000);

#在文本框内输入字符串111
captureDispatchString(111);

#点击另一文本框，坐标（280,589）
Tap(280,589);
captureDispatchString(555);

#输入键值为66的键，66代表Enter
captureDispatchPress(66);
UserWait(2000);
Tap(623,1212);
UserWait(2000)

#输入键值为KEYCODE_BACK（该键值数值为4）的键，代表返回键
captureDispatchPress(KEYCODE_BACK);
UserWait(2000)|
```

③ 脚本运行

1、将脚本文件复制到手机或运行 **adb push <scriptfile> <path>**

(scriptfile 为脚本文件名，path 为手机路径)

2、运行

adb shell monkey -f <scriptfile> [-v -v -v] <event-count>

(event-count 为运行次数，-v 部分依是否需要查看运行日志及详细程度而定)

例:

```
D:\nox-emulator\Nox\bin>nox_adb push firstmonkey.txt /mnt/sdcard/
0 KB/s (1415 bytes in 5.095s)

D:\nox-emulator\Nox\bin>nox_adb shell monkey -f /mnt/sdcard/firstmonkey.txt -v 2
:Monkey: seed=1481467567417 count=2
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
Replaying 0 events with speed 1.0
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x1
0200000;component=superalarm.firsttry/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=
superalarm.firsttry/.MainActivity } in package superalarm.firsttry
:Sending Touch (ACTION_DOWN): 0:(623.0,1212.0)
:Sending Touch (ACTION_UP): 0:(623.0,1212.0)
// Allowing start of Intent { cmp=superalarm.firsttry/.PersonalInformation } in package superalarm.first
try
:Sending Touch (ACTION_DOWN): 0:(623.0,1212.0)
:Sending Touch (ACTION_UP): 0:(623.0,1212.0)
// Allowing start of Intent { cmp=superalarm.firsttry/.WelcomeActivity } in package superalarm.firsttry
// activityResuming(superalarm.firsttry)
// Shell command input text 111 status was 0
:Sending Touch (ACTION_DOWN): 0:(280.0,589.0)
:Sending Touch (ACTION_UP): 0:(280.0,589.0)
// Shell command input text 555 status was 0
:Sending Touch (ACTION_DOWN): 0:(623.0,1212.0)
:Sending Touch (ACTION_UP): 0:(623.0,1212.0)
// Allowing start of Intent { cmp=superalarm.firsttry/.PresonalInformationHaveLogin } in package superal
arm.firsttry
// activityResuming(superalarm.firsttry)
// activityResuming(superalarm.firsttry)
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x1
0200000;component=superalarm.firsttry/.MainActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=
superalarm.firsttry/.MainActivity } in package superalarm.firsttry
:Sending Touch (ACTION_DOWN): 0:(623.0,1212.0)
:Sending Touch (ACTION_UP): 0:(623.0,1212.0)
// Allowing start of Intent { cmp=superalarm.firsttry/.PersonalInformation } in package superalarm.first
try
:Sending Touch (ACTION_DOWN): 0:(623.0,1212.0)
:Sending Touch (ACTION_UP): 0:(623.0,1212.0)
// Allowing start of Intent { cmp=superalarm.firsttry/.WelcomeActivity } in package superalarm.firsttry
// activityResuming(superalarm.firsttry)
// Shell command input text 111 status was 0
:Sending Touch (ACTION_DOWN): 0:(280.0,589.0)
:Sending Touch (ACTION_UP): 0:(280.0,589.0)
// Shell command input text 555 status was 0
:Sending Touch (ACTION_DOWN): 0:(623.0,1212.0)
:Sending Touch (ACTION_UP): 0:(623.0,1212.0)
// Allowing start of Intent { cmp=superalarm.firsttry/.PresonalInformationHaveLogin } in package superal
arm.firsttry
// activityResuming(superalarm.firsttry)
// activityResuming(superalarm.firsttry)
Events injected: 42
:Sending rotation degree=0, persist=false
:Dropped: keys=0 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=27561ms (0ms mobile, 0ms wifi, 27561ms not connected)
// Monkey finished
```

小组项目初版运行结果示例



④ 脚本中包名及主界面的确定

1 如果所测试的 APP 是自己 team 所写，那么包名和主界面名就是自己所设置的包名和主界面名

2 如果测试他人的 APP，可通过如下方式确定

注：该方式需要超级用户权限，部分手机的开发者模式也可进行，但本人的手机开发者模式并不能用，简单测试可考虑虚拟机

命令行运行 `adb shell`，进入所连接设备的控制台，输入 `ls data/data`，查看设备上安装的所有包，找到要测试的包名

```
D:\nox-emulator\Nox\bin>nox_adb shell
root@android:/ # ls data/data
ls data/data
com.android.backupconfirm
com.android.browser
com.android.camera
com.android.certinstaller
com.android.contacts
com.android.defcontainer
com.android.documentsui
com.android.externalstorage
com.android.gallery3d
com.android.inputdevices
com.android.keychain
com.android.keyguard
com.android.location.fused
com.android.noisefield
com.android.onetimeinitializer
com.android.packageinstaller
com.android.pacprocessor
com.android.phasebeam
com.android.phone
com.android.providers.calendar
com.android.providers.contacts
com.android.providers.downloads
com.android.providers.downloads.ui
com.android.providers.media
com.android.providers.settings
com.android.providers.telephony
com.android.providers.userdictionary
com.android.provision
com.android.proxyhandler
com.android.settings
com.android.sharedstoragebackup
com.android.shell
com.android.soundrecorder
com.android.systemui
com.android.vending
com.android.vpndialogs
com.android.wallpaper.livpicker
com.android.wallpapercropper
com.bignox.app.store.hd
com.cyanogenmod.filemanager
com.example.android.softkeyboard
com.google.android.backuptransport
com.google.android.configupdater
com.google.android.feedback
com.google.android.gms
com.google.android.gsf
com.google.android.gsf.login
com.google.android.partnersetup
com.google.android.play.games
com.google.android.syncadapters.contacts
com.svox.pico
com.vphone.helper
com.vphone.launcher
Superalarm.firsttry
```

然后输入 `logcat | busybox grep START`，同样为实时监控，这时打开要测试的 APP，显示如下内容，即可确定主界面为哪一个 Activity。

```
I/ActivityManager( 506): START u0 {act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0x10200000 cmp=superalarm.firsttry/.GuideActivity bnds=[8,379][149,600] (has extras)} from pid 704
I/ActivityManager( 506): START u0 {cmp=superalarm.firsttry/.MainActivity} from pid 2613
```

⑤ 脚本中对绝对坐标的确定

上例中点击操作使用的是绝对坐标，不同分辨率的手机坐标值是不同的，因此只适用于对一台手机进行测试，转换手机测试时需要利用公式改变坐标，这也是Monkey不尽如人意之处。

1 确定手机坐标的范围

命令行输入 `adb shell getevent -p`

```
add device 1: /dev/input/event9
name: "NOX via VirtualBox seamless mouse"
events:
  KEY (0001): 0110
  ABS (0003): 0000 : value 0, min 0, max 65535, fuzz 0, flat 0, resolution 0
               0001 : value 0, min 0, max 65535, fuzz 0, flat 0, resolution 0
input props:
<none>
could not get driver version for /dev/input/mouse4, Not a typewriter
add device 2: /dev/input/event8
name: "NOX via VirtualBox PS/2 mouse"
events:
  KEY (0001): 0110 0111 0112 0113 0114
  REL (0002): 0000 0001 0006 0008
input props:
<none>
could not get driver version for /dev/input/mouse3, Not a typewriter
add device 3: /dev/input/event7
name: "Android Input"
events:
  KEY (0001): 0001 0002 0003 0004 0005 0006 0007 0008
               0009 000a 000b 000c 000d 000e 000f 0010
               0011 0012 0013 0014 0015 0016 0017 0018
               0019 001a 001b 001c 001d 001e 001f 0020
               0021 0022 0023 0024 0025 0026 0027 0028
               0029 002a 002b 002c 002d 002e 002f 0030
               0031 0032 0033 0034 0035 0036 0037 0038
               0039 003a 003b 003c 003d 003e 003f 0040
               0041 0042 0043 0044 0045 0046 0047 0048
               0049 004a 004b 004c 004d 004e 004f 0050
               0051 0052 0053 0054 0055 0056 0057 0058
               0059 005a 005b 005c 005d 005e 005f 0060
               0061 0062 0063 0064 0065 0066 0067 0068
               0069 006a 006b 006c 006d 006e 006f 0070
               0071 0072 0073 0074 0075 0076 0077 0078
               0079 007a 007b 007c 007d 007e 007f 0080
               0081 0082 0083 0084 0085 0086 0087 0088
               0089 008a 008b 008c 008d 008e 008f 0090
               0091 0092 0093 0094 0095 0096 0097 0098
               0099 009a 009b 009c 009d 009e 009f 00a0
               00a1 00a2 00a3 00a4 00a5 00a6 00a7 00a8
               00a9 00aa 00ab 00ac 00ad 00ae 00af 00b0
               00b1 00b2 00b3 00b4 00b5 00b6 00b7 00b8
               00b9 00ba 00bb 00bc 00bd 00be 00bf 00c0
               00c1 00c2 00c3 00c4 00c5 00c6 00c7 00c8
               00c9 00ca 00cb 00cc 00cd 00ce 00cf 00d0
               00d1 00d2 00d3 00d4 00d5 00d6 00d7 00d8
               00d9 00da 00db 00dc 00dd 00de 00df 00e0
               00e1 00e2 00e3 00e4 00e5 00e6 00e7 00e8
               00e9 00ea 00eb 00ec 00ed 00ee 00ef 00f0
               00f1 00f2 00f3 00f4 00f5 00f6 00f7 00f8
               00f9 00fa 00fb 00fc 00fd 00fe 00ff 0110
               0145 014a
  REL (0002): 0006 0008
  ABS (0003): 0035 : value 0, min 0, max 720, fuzz 0, flat 0, resolution 0
               0036 : value 0, min 0, max 1280, fuzz 0, flat 0, resolution 0
               003a : value 0, min 0, max 0, fuzz 0, flat 0, resolution 0
```

可以看到上面有 device1 对应/dev/input/event9, device2 对应 event8, device3 对应 event7, 但我们并不知道自己的手机是哪个 device

接着在命令行运行 `adb shell getevent`, 此命令会实时监控你在 device 上的操作, 然后点击手机上任意地方, 多出如下内容

```
name: VirtualBox USB Tablet
/dev/input/event7: 0001 014a 00000001
/dev/input/event7: 0003 003a 00000001
/dev/input/event7: 0003 0035 0000017e
/dev/input/event7: 0003 0036 00000271
/dev/input/event7: 0000 0002 00000000
/dev/input/event7: 0000 0000 00000000
/dev/input/event7: 0000 0002 00000000
/dev/input/event7: 0000 0000 00000000
/dev/input/event7: 0001 014a 00000000
/dev/input/event7: 0003 003a 00000000
/dev/input/event7: 0003 0035 ffd5ae32
/dev/input/event7: 0003 0036 ffdb6107
/dev/input/event7: 0000 0002 00000000
/dev/input/event7: 0000 0000 00000000
```

可知要测试的手机对应 event7, 即 device3, 在上面 add device3 下找到如下部分

```
ABS (0003): 0035 : value 0, min 0, max 720, fuzz 0, flat 0, resolution 0
            0036 : value 0, min 0, max 1280, fuzz 0, flat 0, resolution 0
            003a : value 0, min 0, max 0, fuzz 0, flat 0, resolution 0
```

0035 后表示 x 的值最小为 0, 最大为 720; 0036 表示 y 的值最小为 0, 最大为 1280

2 确定点击位置的绝对坐标

接下来我们确定刚才点击位置的绝对坐标, 从监控运行的部分同样找到 0035 和 0036

```
/dev/input/event7: 0003 0035 0000017e
/dev/input/event7: 0003 0036 00000271
/dev/input/event7: 0000 0002 00000000
```

但会发现会有多个, 这里只考虑第一次出现的 0035 和 0036。可发现 0035 代表的 x 为 17e, 0036 代表的 y 为 271; 此为 16 进制数, 换为 10 进制为 (382,625), 即绝对坐标。

结束上面过程可输入 Ctrl+C

附录

一、Monkey 命令大全

1 常规类命令

1.1 显示帮助信息

adb shell monkey -h

```
C:\Users\dphs>adb shell monkey -h
usage: monkey [-p ALLOWED_PACKAGE [-p ALLOWED_PACKAGE] ...]
              [-c MAIN_CATEGORY [-c MAIN_CATEGORY] ...]
              [--ignore-crashes] [--ignore-timeouts]
              [--ignore-security-exceptions]
              [--monitor-native-crashes] [--ignore-native-crashes]
              [--kill-process-after-error] [--hprof]
              [--pct-touch PERCENT] [--pct-motion PERCENT]
              [--pct-trackball PERCENT] [--pct-syskeys PERCENT]
              [--pct-nav PERCENT] [--pct-majornav PERCENT]
              [--pct-appswitch PERCENT] [--pct-flip PERCENT]
              [--pct-anyevent PERCENT] [--pct-pinchzoom PERCENT]
              [--pkg-blacklist-file PACKAGE_BLACKLIST_FILE]
              [--pkg-whitelist-file PACKAGE_WHITELIST_FILE]
              [--wait-dbg] [--dbg-no-events]
              [--setup scriptfile [-f scriptfile [-f scriptfile] ...]
              [--port port]
              [-s SEED] [-v [-v] ...]
              [--throttle MILLISEC] [--randomize-throttle]
              [--profile-wait MILLISEC]
              [--device-sleep-time MILLISEC]
              [--randomize-script]
              [--script-log]
              [--bugreport]
              [--periodic-bugreport]
              COUNT
```

1.2 打印日志信息

adb shell monkey -v <event-count>

2 事件类命令(向手机发送事件触发行动)

2.1 脚本运行命令 **adb shell monkey -f <scriptfile> <event-count>**

2.2 指定随机数生成器 seed 值 **adb shell monkey -s <seed> <event-count>**

当运行 **adb shell monkey -v 10** 时, seed 被随机赋值, 用以产生随机事件示例:

```
D:\nox-emulator\Wox\bin>nox_adb shell monkey -v 10
Monkey: seed=1481469823572 count=10
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
// 0: 15.0%
// 1: 10.0%
// 2: 2.0%
// 3: 15.0%
// 4: -0.0%
// 5: 25.0%
// 6: 15.0%
// 7: 2.0%
// 8: 2.0%
// 9: 1.0%
// 10: 13.0%
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x1
02000000;component=com.cyanogenmod.filemanager/.activities.NavigationActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=
com.cyanogenmod.filemanager/.activities.NavigationActivity } in package com.cyanogenmod.filemanager
:Sending Touch (ACTION_DOWN): 0:(569.0,1236.0)
:Sending Touch (ACTION_UP): 0:(565.2443,1237.7893)
:Sending Touch (ACTION_DOWN): 0:(257.0,974.0)
Events injected: 10
:Sending rotation degree=0, persist=false
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=
com.android.settings/.Settings } in package com.android.settings
:Dropped: keys=0 pointers=7 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=335ms (0ms mobile, 0ms wifi, 335ms not connected)
// Monkey finished
```


在日志第一行可以看到 seed 值为 1481469823572，那么执行

```
adb shell monkey -s 1481469823572 -v 10
```

```
D:\nox-emulator\Nox\bin>nox_adb shell monkey -s 1481469823572 -v 10
:Monkey: seed=1481469823572 count=10
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
// 0: 15.0%
// 1: 10.0%
// 2: 2.0%
// 3: 15.0%
// 4: -0.0%
// 5: 25.0%
// 6: 15.0%
// 7: 2.0%
// 8: 2.0%
// 9: 1.0%
// 10: 13.0%
:Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.cyanogenmod.filemanager/.activities.NavigationActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.cyanogenmod.filemanager/.activities.NavigationActivity } in package com.cyanogenmod.filemanager
:Sending Touch (ACTION_DOWN): 0:(569.0,1236.0)
:Sending Touch (ACTION_UP): 0:(565.2443,1237.7893)
:Sending Touch (ACTION_DOWN): 0:(257.0,974.0)
Events injected: 10
:Sending rotation degree=0, persist=false
:Dropped: keys=0 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=65ms (0ms mobile, 0ms wifi, 65ms not connected)
// Monkey finished
```

可以看出与之前随机操作结果完全相同，该命令用以重复之前的随机操作

2.3 规定操作间隔时间 `adb shell monkey --throttle <time>`

例如 `adb shell monkey -f monkey.txt --throttle 5000 2` 表示两次脚本运行时间间隔两秒

2.4 规定随机事件中某种事件所占百分比

1) 触摸事件百分比

```
adb shell monkey --pct-touch <percent>
```

2) 动作事件百分比

```
adb shell monkey --pct-motion <percent>
```

3) 轨迹球事件百分比

```
adb shell monkey --pct-trackball <percent>
```

4) 基本导航事件百分比

```
adb shell monkey --pct-nav <percent>
```

5) 主要导航事件百分比

```
adb shell monkey --pct-majornav <percent>
```

6) 系统按键事件百分比

```
adb shell monkey --pct-syskeys <percent>
```

7) 应用启动事件百分比

```
adb shell monkey --pct-appswtich <percent>
```

8) 其他类型事件百分比

```
adb shell monkey --pct-anyevent <percent>
```

3 约束类命令（把测试约束在某个包或类下）

3.1 测试包

```
adb shell monkey -p <package_name> <time_event>
```

3.2 测试类

```
adb shell monkey -c <main_category> <time_event>
```

4 调试类命令

4.1 监视应用程序所调用包之间的转换

```
adb shell monkey --dbg-no-events <time_event>
```

4.2 在事件序列前后立刻生成 profiling report

```
adb shell monkey --hprof <time_event>
```

4.3 出错后发出事件的使用

1) 在应用程序崩溃时继续发送事件

```
adb shell monkey --ignore-crashes <time_event>
```

2) 在任何超时错误发生时继续发送事件

```
adb shell monkey --ignore-timeouts <time_event>
```

3) 在应用程序权限错误后继续发送事件

```
adb shell monkey --ignore-security-exceptions <time_event>
```

4) 在应用程序出错后通知系统停止发生的进程

```
adb shell monkey --kill-process-after-error <time_event>
```

5) 监视并报告 monkey 运行时 Android 系统 native code 的崩溃事件

```
adb shell monkey --monitor-native-crashes <time_event>
```

4.4 暂停执行中的 monkey，直到调试器与它相连

```
adb shell monkey --wait-dbg <time_event>
```

二、Monkey 脚本函数（Monkey API 解释）

1 启动应用

`LaunchActivity(String pkg_name, String cl_name)`

2 唤醒屏幕

`DeviceWakeUp()`

3 等待事件

3.1 `UserWait (long time)` （单位 ms）

3.2 `ProfileWait()` 等待 5 秒

4 点击

4.1 `Tap(float x,float y,[long typeDuration)`

(x,y)为坐标，typeDuration 为持续时间

4.2 `captureDispatchPointer (long downTime, long eventTime, int action,`

`float x, float y, float pressure, float size, int metaState, float xPrecision,`

`float yPrecision, int device, int edgeFlags)`

需要关注的主要是 (x,y) 为坐标值，action 为 0 时为按下，1 为抬起；模拟一次点击需要两个 `captureDispatchPointer`，函数中的 `capture` 也可以省略其余变量的说明（实际编写时可以直接参考他人代码保持不动）：

long downTime, 键最初被按下的时间

long eventTime, 事件发生的时间

float pressure, 当前事件的压力，范围 0-1

float size, 触摸的近似值，范围 0-1

int metaState, 当前按下的 meta 键的标识

float xPrecision, x 坐标精确值

float yPrecision, y 坐标精确值

int device, 事件来源，范围 0-x，0 表示不来自物理设备

int edgeFlags, 坐标是否超出了屏幕范围

5 轨迹球事件

`DispatchTrackball (long downTime, long eventTime, int action, float x,`

`float y, float pressure, float size, int metaState, float xPrecision, float`

`yPrecision, int device, int edgeFlags)`

6 输入字符串事件

`DispatchString(String text)` text 不需要引号

7 按下按键

`DispatchPress(int keyCode)`

8 长按按键

8.1 `LongPress(int keyCode)` keyCode 即键值

8.2 `PressAndHold(x,y,pressDuration)` pressDuration 为持续时间

9 发送键值

`DispatchKey(long downTime, long eventTime, int action, int code, int repeat, int metaState, int device, int scanCode)`

repeat 表示重复次数

10 开关软键盘

`DispatchFlip(boolean keyboardOpen)`

keyboardOpen=true 表示打开，false 表示关闭

11 旋转屏幕

`RotateScreen(rotationDegree,persist)`

rotationDegree:四个参数 0 90 180, 270 分别代表 0, 90, 180, 270

persist:两个参数 ^0,0 旋转后固定和旋转后不固定

12 模拟拖拽操作

`Drag(float xStart,float yStart,float xEnd,float yEnd,int stepCount)`