



中山大學  
SUN YAT-SEN UNIVERSITY

《移动互联网编程实践》

# 实验报告

(Monkey 测试的调研与实践)

学 院 名 称 : 计算机学院

专业 (班级) : 计算机科学与技术

学 生 姓 名 : 王若琪

学 号 : 18340166

时 间 : 2020 年 12 月 23 日

# 目录

1. 调研的基础知识.....	2
1.1 Monkey 测试简介.....	2
1.2 Monkey 的工作原理.....	2
1.3 Monkey 参数详解.....	3
1.4 Monkey 测试特点.....	4
1.5 Monkey 日志分析方法.....	4
1.6 基础 Monkey 的缺陷.....	5
1.7 智能 Monkey 方法简述.....	6
2. 实验环境条件.....	6
2.1 设备平台.....	6
2.2 Monkey 环境.....	6
2.3 安卓模拟器.....	7
2.4 目标应用.....	8
3. 实验详细过程.....	8
3.1 基础 Monkey 命令尝试.....	8
3.2 智能 Monkey 的实现.....	11
3.2.1 Adb 控制相关函数.....	11
3.2.2 智能 Monkey 控制部分.....	12
3.2.3 运行智能测试方法.....	14
4. 测试效果.....	15
4.1 基础 Monkey 测试时 APP 界面变化举例.....	15
4.2 智能化 Monkey 测试时 APP 界面变化举例.....	16
4.3 智能化 Monkey 控制端输出效果.....	18
4.4 对比分析.....	19
5. 总结与反思.....	19
6. 参考网页.....	20

# 1. 调研的基础知识

## 1.1 Monkey 测试简介

Monkey 是一款 app 的自动化测试工具，Monkey 是猴子的意思，所以从原理上说，它的自动化测试就类似猴子一样在软件上乱敲按键，猴子什么都不懂，就爱捣乱。Monkey 原理也是类似，通过向系统发送伪随机的用户事件流（如按键输入、触摸屏输入、滑动 Trackball、手势输入等操作），来对设备上的程序进行测试，检测程序长时间的稳定性，多久的时间会发生异常。

Monkey 工具存在 Android 系统中，使用 Java 语言写成，jar 包在 Android 文件系统中的存放路径是：`/system/framework/monkey.jar`；Monkey.jar 程序是由一个名为“monkey”的 Shell 脚本来启动执行，shell 脚本在 Android 文件系统中的存放路径是：`/system/bin/monkey`。

Monkey 需要通过 adb 来唤醒，即通过在 cmd 窗口中执行：`adb shell monkey {+命令参数}` 来进行 Monkey 测试。

## 1.2 Monkey 的工作原理

在 Monkey 运行的时候，它会生成事件，并把它们发给系统。同时，Monkey 还对测试中的系统进行监测，对下列三种情况进行特殊处理：

(1) 如果限定了 Monkey 运行在一个或几个特定的包上，那么它会监测试图转到其它包的操作，并对其进行阻止；

(2) 如果应用程序崩溃或接收到任何失控异常，Monkey 将停止并报错；

(3) 如果应用程序产生了应用程序不响应 ANR(application not responding)的错误，Monkey 将会停止并报错；

按照选定的不同级别的反馈信息，在 Monkey 中还可以看到其执行过程报告和生成的事件。

## 1.3 Monkey 参数详解

### (1) `-p` 参数

参数`-p`用于约束限制。

### (2) `-v` 参数

定义执行 Monkey 的时候的日志显示详情。

### (3) `-s` [`-s SEED`]

在多次执行 Monkey 的时候，就算指令完全一样，发出的随机事件及顺序不一样，在每次执行的时候，可以都要给相同的一个种子数（随机的数字），发出的随机事件及顺序就一样。

主要使用的场景：复现之前 Monkey 执行时候出现的问题。

### (4) `--throttle` [`-throttle MILLISEC`]

在每次事件发送的时间间隔，以一定频率发送事件。

### (5) [`-pct-touch PERCENT`]

PERCENT 表示占总事件的百分之多少。

调整发送的触摸事件的比例功能，点击一个位置，再松开（`action_down/action_up`）。

### (6) [`-pct-motion PERCENT`]

PERCENT 表示占总事件的百分之多少。

调整发送的拖动/移动事件的比例功能，点击一个位置，拖到另一个位置，再松开（`action_down/ACTION_MOVE/action_up`）。

### (7) [`-pct-syskeys PERCENT`]

PERCENT 表示占总事件的百分之多少。

模拟设备的电话、主页、音量加、减等操作。

### (8) `--pct-appswitch PERCENT`

PERCENT 表示占总事件的百分之多少。

覆盖你指定包（app）activity，切换不同的页面。

#### (9) [- ignore-crashes]

忽略 Monkey 执行过程中 crash 这种问题，继续执行 Monkey。

通常，应用发生崩溃或异常时 Monkey 会停止运行。如果设置此项，Monkey 将继续发送事件给系统，直到事件计数完成。

#### (10) - ignore-timeouts

通常，应用程序发生任何超时错误（如“Application Not responding”对话框）Monkey 将停止运行，设置此项，Monkey 将继续发送事件给系统，直到事件计数完成。

注：常用。

#### (11) - ignore-security-exception

通常，当程序发生许可错误（例如启动一些需要许可的 Activity）导致的异常时，Monkey 将停止运行。设置此项，Monkey 将继续发送事件给系统，直到事件计数完成。

### 1.4 Monkey 测试特点

- (1) 测试的对象仅为应用程序包，有一定的局限性。
- (2) Monkey 测试使用的事件流数据流是随机的，不能进行自定义。
- (3) 可对 MonkeyTest 的对象，事件数量，类型，频率等进行设置。

### 1.5 Monkey 日志分析方法

#### (1) 初步分析

- ① 找到是 Monkey 里面的哪个地方出错
- ② 查看 Monkey 里面出错前的一些事件动作，并手动执行该动作
- ③ 若以上步骤还不能找出，可以使用之前执行的 Monkey 命令再执行一遍，注意 seed 值要一样

#### (2) 一般结果分析

- ① ANR 问题：在日志中搜索“ANR”。
- ② 崩溃问题：在日志中搜索“Exception”。

### (3) 详细日志分析

首先需要查看 Monkey 测试中是否出现了 ANR 或者异常，具体方法如上述。将执行 Monkey 生成的 log，从手机中导出并打开查看该 log；在 log 的最开始都会显示 Monkey 执行的 seed 值、执行次数和测试的包名。具体方法如上述。然后我们要分析 log 中的具体信息，查看 log 中第一个 Switch，主要是查看 Monkey 执行的是那一个 Activity。在下一个 switch 之间的，如果出现了崩溃或其他异常，可以在该 Activity 中查找问题的所在。

## 1.6 基础 Monkey 的缺陷

### (1) 遍历界面有限

因为 Monkey 是不断输入伪随机事件流来测试应用，所以无法控制输入的事件，这导致有些较深的界面很难进入。最终导致测试遍历的界面有限。

### (2) 路径回环

在随机测试的过程中，如果程序中存在大量的环路，就会容易出现一直重复相同路径的现象，造成路径循环。

### (3) 很难有针对性测试

因为 Monkey 对于测试事件和数据都是随机的，自定义程度不高，无法对指定界面和指定动作进行测试，所以有很大的局限性。

### (4) 无意义动作过多

当某界面可操作按钮较少时，

### (5) 容易在某些界面卡死

在 Monkey 测试中，如果 app 在某个界面难以退出，或者难以找到进一步搜索的方法时，就会导致 app 在这个界面上卡死。

## 1.7 智能 Monkey 方法简述

(1) 针对基础 Monkey 测试中容易在某一界面卡死、遍历界面有限、无意义动作过多的问题，有以下解决方案：

可以人为设定一个时间限度，获取记录在某一界面的停留时间来判断是否在同一截面停留时间过长，如果超过这一期限，就随机打开一个人为设定好的界面（可以包括初始界面或功能比较多的界面等）。

如果使用这种方法，可以避免在某一界面卡死，从而有更多机会和时间去遍历其他页面，并且可以减少在此类界面上的无意义重复动作，一举三得。

(2) 针对缺乏针对性的问题，可以通过写脚本、调整各个参数等来指定特定界面和操作方式，从而有针对性地对某一软件、某一功能进行测试。

## 2. 实验环境条件

### 2.1 设备平台

安卓（安卓模拟器）+ Windows 10 + python 3.7

### 2.2 Monkey 环境

JDK+SDK

JDK 是 Java 语言的软件开发工具包，主要用于移动设备、嵌入式设备上的 java 应用程序。JDK 是整个 java 开发的核心，它包含了 JAVA 的运行环境（JVM+Java 系统类库）和 JAVA 工具。

Android SDK 包含了许多可以帮助你开发 Android 平台应用的工具。这些工具分为两类：一是 SDK 工具；二是平台工具。SDK 工具独立于平台，任何开发 Android 应用的平台都需要配置。平台工具是定制的适应

最新的 Android 平台特性。

### 搭建环境过程

先下载并安装 JDK，过程参考网址如下，这里不做赘述：

<https://blog.csdn.net/write6/article/details/79136388>

再下载安装安卓 SDK，过程参考网址如下，这里不做赘述：

<https://blog.csdn.net/u011541946/article/details/77142045>

## 2.3 安卓模拟器

由于没有安卓手机，所以本实验采用效果完全一致的安卓模拟器来模拟。为此我下载安装了夜神安卓模拟器，采用夜神安卓模拟器进行 adb 连接时，需要进入设置，双击“版本号”来启动开发者模式，之后再在“开发者选项”中打开 USB 调试选项。



之后，将 SDK\platform-tools 下的 adb.exe 文件，替换成夜神模拟器 bin 目录下的 adb.exe 文件，即可进行 ADB 连接，连接过程如下：

先在夜神模拟器 bin 目录下的命令行中输入：

```
nox_adb.exe connect 127.0.0.1:62001
```

再在任意目录下的命令行输入：

```
adb connect 127.0.0.1:62001
```

即可成功进行对夜神安卓模拟器的 adb 连接。



## 2.4 目标应用

我选择“普通移动应用”中的新浪微博平台进行测试，因为新浪微博界面功能较多，并且拥有多种功能模块，如小视频、直播等，所以测试价值较高。打开新浪微博应用主界面如下：



## 3. 实验详细过程

### 3.1 基础 Monkey 命令尝试

安装配置好环境，并且使用 adb 连接成功之后，先用 Monkey 命令做一些最基本的测试：

先打开安卓模拟器，进行 adb 连接，并启动要测试的新浪微博：

```
C:\Program Files (x86)\Nox\bin>nox_adb.exe connect 127.0.0.1:62001  
already connected to 127.0.0.1:62001
```

```
C:\Users\wangr>adb connect 127.0.0.1:62001  
already connected to 127.0.0.1:62001
```

检查是否连接成功:

```
C:\Users\wangr>adb devices
List of devices attached
127.0.0.1:62001 device
```

查看要测试的安装包名:

```
adb shell logcat|findstr "Displayed"
```

```
C:\Users\wangr>adb shell logcat|findstr "Displayed"
12-20 13:54:12.255 2169 2189 I ActivityManager: Displayed com.android.settings/.FallbackHome: +191ms
12-20 13:54:12.868 2169 2189 I ActivityManager: Displayed com.vphone.launcher/.Launcher: +310ms
12-20 14:24:33.352 2169 2189 I ActivityManager: Displayed com.android.settings/.Settings: +377ms
12-20 14:24:45.092 2169 2189 I ActivityManager: Displayed com.android.settings/.Settings$DeviceInfoSettingsActivity: +243ms
12-20 14:24:49.363 2169 2189 I ActivityManager: Displayed android.com.android.internal.app.PlatformLogoActivity: +242ms
12-20 14:25:34.961 2169 2189 I ActivityManager: Displayed com.android.settings/.Settings$WirelessSettingsActivity: +159ms
12-20 14:25:42.992 2169 2189 I ActivityManager: Displayed com.android.settings/.Settings$DevelopmentSettingsActivity: +151ms
12-20 14:27:04.584 2169 2189 I ActivityManager: Displayed com.android.settings/.Settings$DeviceInfoSettingsActivity: +120ms
12-20 15:00:10.674 2169 2189 I ActivityManager: Displayed com.android.settings/.Settings: +308ms
12-20 15:00:12.296 2169 2189 I ActivityManager: Displayed com.android.systemui/.recents.RecentsActivity: +154ms
12-20 15:00:21.534 2169 2189 I ActivityManager: Displayed com.sina.weibo/.SplashActivity: +852ms
12-20 15:00:23.519 2169 2189 I ActivityManager: Displayed com.sina.weibo/.MainTabActivity: +298ms (total +319ms)
12-20 15:02:56.079 2169 2189 I ActivityManager: Displayed com.sina.weibo/.feed.DetailWeiboActivity: +123ms
12-20 15:04:07.051 2169 2189 I ActivityManager: Displayed com.sina.weibo/.feed.DetailWeiboActivity: +123ms
```

如上图, 其中红框中的内容就是要测试的 APP 相关内容, 其中斜杠前面的内容 com.sina.weibo 是指包名, 斜杠后面的内容是 appActivity (后文进行智能化 Monkey 化测试时会用到)。

接下来在指定这个包, 输入如下的命令, 此时指定软件 (新浪微博) 就会被打开然后开始随机点击、滑动等:

```
adb shell monkey -p com.sina.weibo -v -v -v 100
```

输入指令后, 会自动在新浪微博应用里进行种类和比例如下的操作:

```
// Event percentages:
// 0: 15.0%
// 1: 10.0%
// 2: 2.0%
// 3: 15.0%
// 4: -0.0%
// 5: -0.0%
// 6: 25.0%
// 7: 15.0%
// 8: 2.0%
// 9: 2.0%
// 10: 1.0%
// 11: 13.0%
```

其中各个序号对应的事件和操作如下:

- 0: 触摸事件百分比, 参数--pct-touch
- 1: 滑动事件百分比, 参数--pct-motion
- 2: 缩放事件百分比, 参数--pct-pinchzoom
- 3: 轨迹球事件百分比, 参数--pct-trackball
- 4: 屏幕旋转事件百分比, 参数--pct-rotation
- 5: nav 导航事件, 就是上下左右, 参数--pct-nav

- 6: 基本导航事件百分比, 参数--pct-nav
- 7: 主要导航事件百分比, 参数--pct-majornav
- 8: 系统事件百分比, 参数--pct-syskeys
- 9: Activity 启动事件百分比, 参数--pct-appswitch
- 10: 键盘翻转事件百分比, 参数--pct-flip
- 11: 其他事件百分比, 参数--pct-anyevent

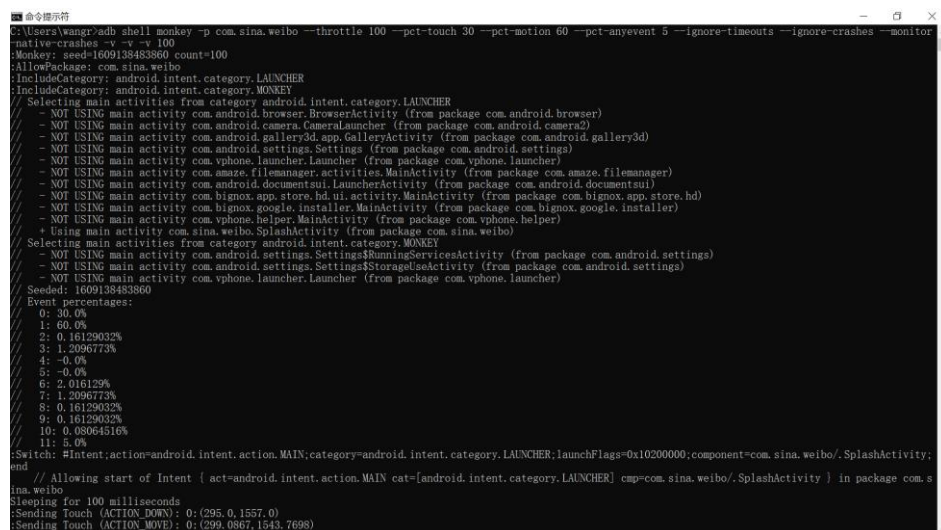
接下来再专门有针对性地设置一些参数的值, 进行尝试, 命令如下:

```
adb shell monkey -p com.sina.weibo --throttle 100 --pct-touch 30 --pct-motion 60 --pct-anyevent 5 --ignore-timeouts --ignore-crashes --monitor-native-crashes -v -v -v 100
```

-p 是允许的包名为 com.sina.weibo, --throttle 100 意为用户操作的延时为 100 毫秒, --pct-touch 30 是指触摸事件的百分比为 30, --pct-motion 60 是指滑动事件的百分比为 60, --pct-anyevent 5 是指其他事件的百分比为 5。

--ignore-timeouts --ignore-crashes --monitor-native-crashes -v -v -v 100, 表示测试时如果有发生 crashes 或者 timeouts, monkey 仍会正常继续, 但是会监控 native-crashes。-v -v -v : 日志级别 2, 最详细的日志, 包括测试中选中/未选中的 Activity 信息。100 是事件数, 表示进行 100 次随机动作。

在命令行输入这条命令后, 会进行按照参数设置要求的 Monkey 测试:



```
C:\Users\wangr>adb shell monkey -p com.sina.weibo --throttle 100 --pct-touch 30 --pct-motion 60 --pct-anyevent 5 --ignore-timeouts --ignore-crashes --monitor-native-crashes -v -v -v 100
Monkey: seed=160913843860 count=100
AllowPackage: com.sina.weibo
IncludeCategory: android.intent.category.LAUNCHER
IncludeCategory: android.intent.category.MONKEY
// Selecting main activities from category android.intent.category.LAUNCHER
- NOT USING main activity com.android.browser.BrowserActivity (from package com.android.browser)
- NOT USING main activity com.android.camera.CameraLauncher (from package com.android.camera2)
- NOT USING main activity com.android.gallery3d.app.GalleryActivity (from package com.android.gallery3d)
- NOT USING main activity com.android.settings.Settings (from package com.android.settings)
- NOT USING main activity com.vphone.launcher.Launcher (from package com.vphone.launcher)
- NOT USING main activity com.amaze.filemanager.activities.MainActivity (from package com.amaze.filemanager)
- NOT USING main activity com.android.documentsui.LauncherActivity (from package com.android.documentsui)
- NOT USING main activity com.bignox.app.store.hd.ui.activity.MainActivity (from package com.bignox.app.store.hd)
- NOT USING main activity com.bignox.google.installer.MainActivity (from package com.bignox.google.installer)
- NOT USING main activity com.vphone.helper.MainActivity (from package com.vphone.helper)
+ Using main activity com.sina.weibo.SplashActivity (from package com.sina.weibo)
Selecting main activities from category android.intent.category.MONKEY
- NOT USING main activity com.android.settings.Settings$RunningServicesActivity (from package com.android.settings)
- NOT USING main activity com.android.settings.Settings$StorageUseActivity (from package com.android.settings)
- NOT USING main activity com.vphone.launcher.Launcher (from package com.vphone.launcher)
Seeded: 160913843860
Event percentages:
0: 30.0%
1: 60.0%
2: 0.16129032%
3: 1.2096773%
4: -0.0%
5: -0.0%
6: 2.016129%
7: 1.2096773%
8: 0.16129032%
9: 0.16129032%
10: 0.08064516%
11: 5.0%
Switch: H:Intent:action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x10200000;component=com.sina.weibo/.SplashActivity;end
// Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com.sina.weibo/.SplashActivity } in package com.sina.weibo
Sleeping for 100 milliseconds
Sending Touch (ACTION_DOWN): 0: (295.0, 1557.0)
Sending Touch (ACTION_MOVE): 0: (299.0867, 1543.7698)
```

【相关实验效果具体分析请见“测试效果”模块。】

## 3.2 智能 Monkey 的实现

实现了上文中提到的第一种智能解决方案，即能够解决测试时遍历界面有限、无意义动作过多、在某界面卡死这三方面问题，具体方案和程序关键代码描述如下：

### 3.2.1 Adb 控制相关函数

首先介绍有关 adb 控制的部分，下文抽选几个关键部分进行描述，其余部分完整代码在源文件 adbfunc.py 中。

在 python 中控制执行 adb 指令的函数 call\_adb(command)能够让系统执行 command 中预先设置好的命令，关键代码如下：

```
1. def call_adb(command):
2.     command_result = ''
3.     command_text = f'adb {command}'
4.     print("执行命令: " + command_text)
5.     results = os.popen(command_text, "r")
6.     while 1:
7.         line = results.readline()
8.         if not line:
9.             break
10.        command_result += line
11.    results.close()
12.    return command_result
```

判断测试 app 是否在某个界面，通过判断 package\_name 和 module\_key 与获取到的当前 activity 是否一致，来确定是否 app 仍然停留在当前页面。这个函数能够在程序开始 Monkey 测试之前，用来确定手机中打开的 app 是否和 config 中设定的 app 一致。关键代码如下：

```
1. def if_top(self, pkg_name, moduleKey):
2.     result = self.current_activity()
3.     if result == '':
4.         return "the process doesn't exist."
5.     print(result)
```

```

6.  if (pkg_name in result and moduleKey in result and "leakcanary" not in
    result) or "WkBrowserActivity" in result or "CordovaWebActivity" in re
    sult:
7.      return True
8.  else:
9.      return False

```

判断是否测试中 app 在某个界面停留时间过久，这是本实验较为核心的模块，通过判断当前时间减去本界面开始时间的值是否大于设定值，来确定是否此界面的停留时间过长或是是否“卡死”。本模块关键代码如下：

```

1.  def if_dead_lock(self, start_time, current_activity, max_time):
2.      while current_activity == self.current_activity():
3.          end_time = time.time()
4.          print(f"同一页面停留时间: {(end_time - start_time)}")
5.          if end_time - start_time > max_time:
6.              print("超过限定时间: 随机打开设定的 activity")
7.              return True
8.          else:
9.              return False
10. else:
11.     return False

```

### 3.2.2 智能 Monkey 控制部分

接下来介绍 test.py 中的功能，也就是和智能 Monkey 程序运行流程相关的部分：

建立一个 log 文件夹，储存每次测试的结果日志，每 log 文件夹中存放以时间命名的子目录，代表那一时间的运行记录。每个以时间命名的子目录下有三个文件，分别是 anr\_traces.log，logcat.log，monkey.log，分别用来存放 ANR traces、Monkey 测试时手机日志和 monkey 记录。部分关键代码如下：

```

1.  logDir = os.path.join("log", f"{datetime.datetime.now().strftime('%Y%m%
    d_%p_%H%M%S')}")

```

```

2. os.makedirs(logDir)
3. adbLogFileName = os.path.join(logDir, "logcat.log")
4. monkeyLogFile = os.path.join(logDir, "monkey.log")
5. monkeyConfig.monkeyCmd = f"adb -
   s {device} shell {monkeyConfig.monkeyCmd + monkeyLogFile}"

```

之后开始 Monkey 测试，并在此过程中对其进行智能控制：

```

1. # 开始测试
2. rt_monkey(monkeyConfig.monkeyCmd, logDir)

```

这其中的 monkeyConfig.monkeyCmd 设置如下：

```

1. monkeyCmd = f"monkey -p {package_name} --throttle 300 " \
2.     "--pct-appswitch 5 --pct-touch 30 --pct-motion 60 --pct-
   anyevent 5 " \
3.     "--ignore-timeouts --ignore-crashes --monitor-native-
   crashes -v -v -v 3000 > "

```

这条指令的参数解释为：

-p 是允许的包名为 {package\_name}，--throttle 300 意为用户操作的延时为 300 毫秒，--pct-appswitch 5 是指 Activity 启动的百分比是 5，--pct-touch 30 是指触摸事件的百分比为 30，--pct-motion 60 是指滑动事件的百分比为 60，--pct-anyevent 5 是指其他时间的百分比为 5。

--ignore-timeouts --ignore-crashes --monitor-native-crashes -v -v -v 3000 > 表示测试时如果有发生 crashes 或者 timeouts，monkey 仍会正常继续，但是会监控 native-crashes。-v -v -v : 日志级别 2，最详细的日志，包括测试中选中/未选中的 Activity 信息。3000 是事件数，表示进行 3000 次随机动作。

启动 Monkey 测试之后正式进入智能控制。整个智能测试控制是在一个循环内部，条件一直为 True，直到判断结束事件发生跳出循环。循环中内容介绍如下：

在循环的开始，先判断当前 app 的 module 是否符合 config 中的设定，然后再打开设定好的第一个界面（微博主界面），相关代码如下：

```
1. # 判断测试的 app 的 module 是否在 top
2. if AndroidDebugBridge().if_top(monkeyConfig.package_name,
3.                                 monkeyConfig.module_key) is False:
4.     # 打开第一个设置好的 activity（微博主界面）
5.     adb.open_app(monkeyConfig.package_name, monkeyConfig.activity[0], device)
```

之后对当前界面进行判断，判断是否有在一个界面卡死的现象存在，这里设定最长时间范围是 10 秒之内，否则就认为在当前界面停留过久，然后随机打开下一个在 config 中预先定义存在的界面，并继续 Monkey 测试工作。关键代码如下：

```
1. # 判断测试时是否有在一个界面卡死的现象，设置时间范围为 10
2. AndroidDebugBridge().if_dead_lock(start_activity_time, current_activity
3.                                     , 10):
4.     adb.open_app(monkeyConfig.package_name, random.choice(monkeyConfig.activity), device)
5.     start_activity_time = time.time()
```

最后判断是否跳出智能控制循环：

```
1. # 判断是否结束
2. with open(monkeyLogFile, "r", encoding='utf-8') as monkeyLog:
3.     if monkeyLog.read().count('Monkey finished') > 0:
4.         print(f"{device}\n 测试结束")
5.         break
```

### 3.2.3 运行智能测试方法

① 打开安卓模拟器，并打开待测试的 app 界面（本实验为新浪微博），并按照前文描述的方法进行 adb 连接。

② 在源代码文件夹的 cmd 命令行中输入命令 `python test.py`，进行智能 Monkey 测试。

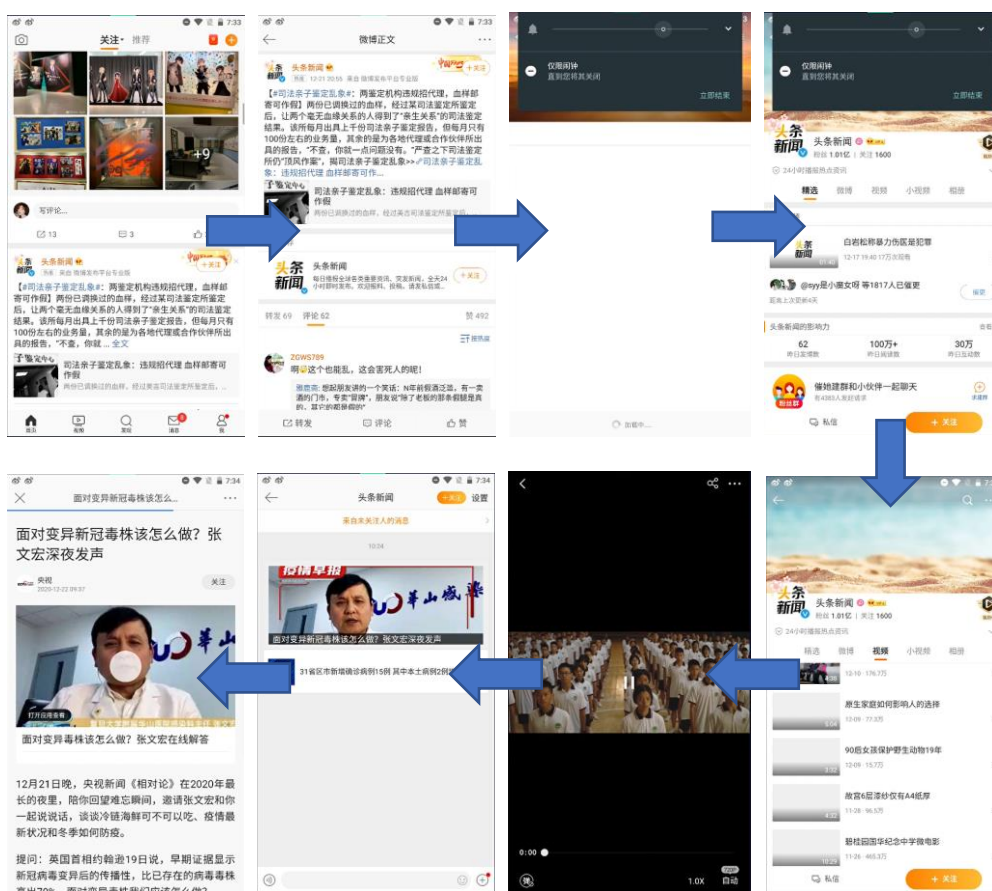


③ 查看并分析 log 文件夹下的测试记录和日志。

## 4. 测试效果

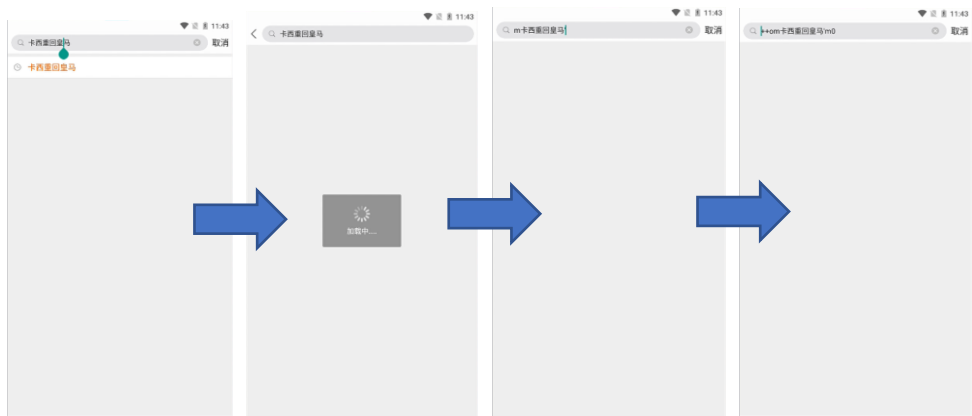
### 4.1 基础 Monkey 测试时 APP 界面变化举例

随机基础测试时，新浪微博的很多界面会被遍历到，测试部分过程截图举例如下：



但是当基础 Monkey 测试进入到某些界面，如搜索框的时候，就会因为难以随机触发退出而一直在搜索框内输入乱码，造成在这个界面停留时间过长，出现在当前界面“卡死”现象，举例如下：





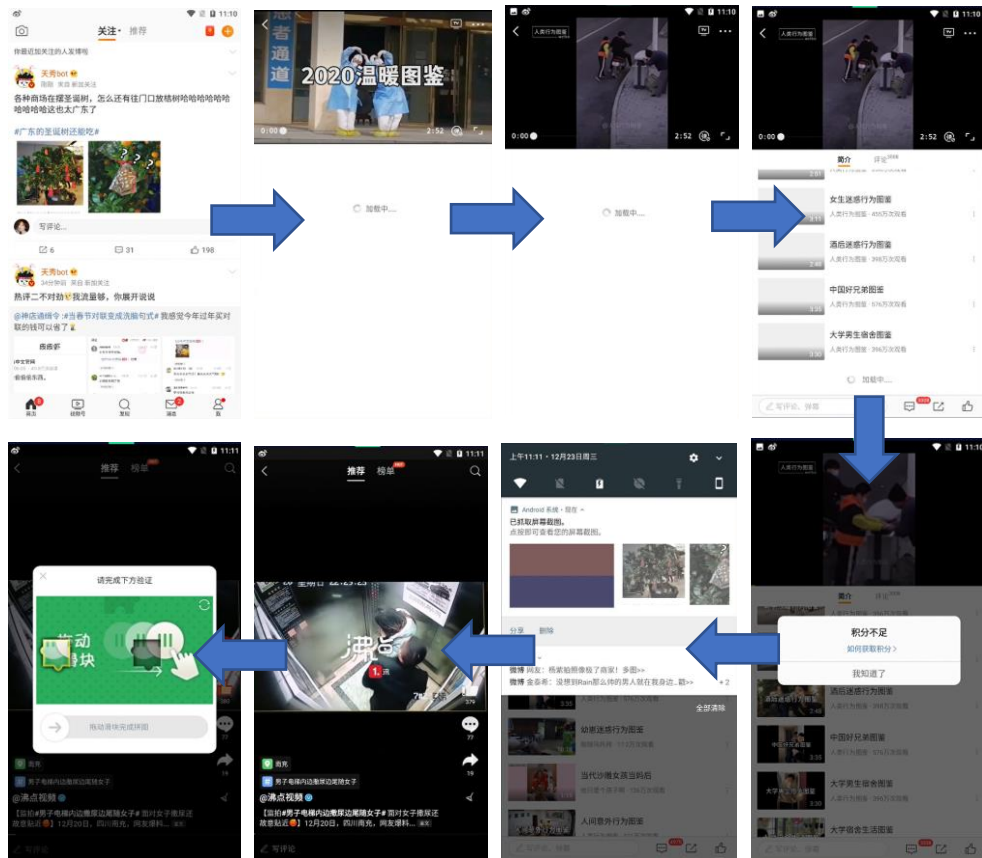
将搜索框内容放大展示：



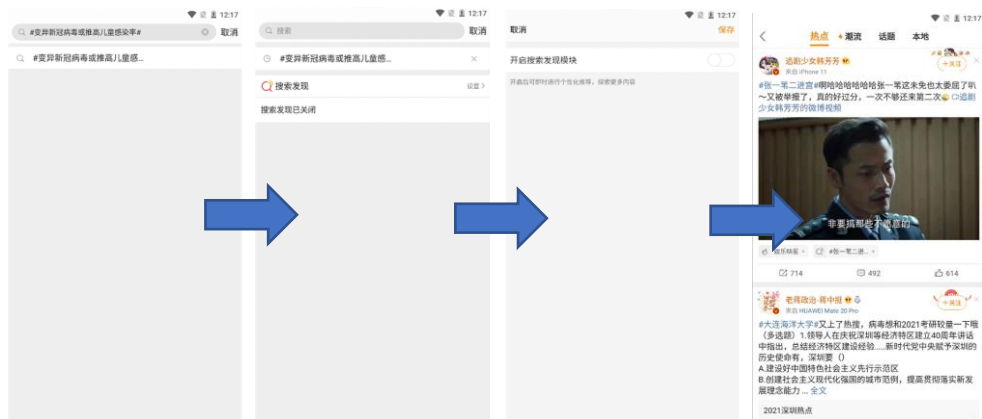
如上所示一系列测试截图，当新浪微博进入搜索界面时，Monkey 随机操作总是会在搜索框内继续输入随机内容，很难返回，导致测试一直停留在这一界面，进行许多无意义的动作，浪费时间。所以需要智能化 Monkey 来改进，改进效果见后文。

## 4.2 智能化 Monkey 测试时 APP 界面变化举例

首先，测试智能 Monkey 在没有卡死情况下的性能，可见由于在 python 源文件 (config.py) 中定义好了操作动作的比例，使其更加适用于新浪微博 APP，所以无意义动作相比于基础 Monkey 测试的较少，界面遍历也较为丰富：



针对基础 Monkey 测试容易在某些界面卡死的现象（以上文中的搜索界面为例），智能化 Monkey 有很大的改进，效果展示如下：



可见在智能 Monkey 测试中，当在搜索界面停留超过限定时间时，会自动切换到其他界面，避免了“卡死”现象。

在智能 Monkey 测试中，当在搜索界面停留超过 10 个单位时间时，就会自动跳转到其他界面，所以在相同时间，相同随机操作数内，智能

Monkey 遍历的界面明显多于基础 Monkey，并且智能 Monkey 无意义动作占比也明显低于基础 Monkey。

## 4.3 智能化 Monkey 控制端输出效果

智能化 Monkey 测试时，程序会实时输出当前的界面 (Activity)，并且如果没有检测到“卡死”，就正常继续；如果有“卡死”的现象（即在同一界面停留时间超过 10 个单位时间不改变），就随机跳转至 config 中设定好的界面继续进行 Monkey 测试。

测试时如果在一个界面停留时间没有超过 10 个单位时间时，运行智能 Monkey 的 python 代码控制端输出如下图，其中 mResumedActivity: ActivityRecord{4dd60dcu0 com.sina.weibo/.MainTabActivity t5} 的意思是当前界面是在主界面：

```
同一页面停留时间: 2.8815104961395264
执行命令: adb shell dumpsys activity | findstr mResumedActivity
mResumedActivity: ActivityRecord{4dd60dc u0 com.sina.weibo/.MainTabActivity t5}
```

当测试时遇到在同一个界面停留超过 10 个单位时间没有改变界面时，智能 Monkey 会检测到“卡死”现象，自动跳转实现在运行智能 Monkey 的 python 代码控制端的输出如下图：

```
执行命令: adb shell dumpsys activity | findstr mResumedActivity
执行命令: adb shell dumpsys activity | findstr mResumedActivity
同一页面停留时间: 12.213938474655151
超过限定时间: 随机打开设定的activity
执行命令: adb -s 127.0.0.1:62001 shell am start -n com.sina.weibo/.page.NewCardListActivity
执行命令: adb shell dumpsys activity | findstr mResumedActivity
mResumedActivity: ActivityRecord{1113c20 u0 com.sina.weibo/.page.NewCardListActivity t3}
```

意为在当前界面停留过久，为了避免卡死，从而增加相同时间内遍历界面的数目，同时也减少无意义的操作数，就随机跳转到预先设定好的界面之一，比如上图为在之前界面“卡死”时，进行智能控制干预，使其随机跳转到了预先在 config.py 文件中人为设定好的界面之一 com.sina.weibo/.page.NewCardListActivity。

## 4.4 对比分析

比较指标\测试方法	基础 Monkey	智能 Monkey
有无“卡死”现象	有	无
无意义动作数	多	少
相同时间内遍历界面数	少	多

根据 4.1 至 4.3 所述实验结果进行对比分析，得出上表所示结果。发现在基础 Monkey 测试的过程中，由于总是在某些界面难以退出，导致一直在做无意义的动作，浪费时间，并导致限定时间内遍历界面过少等问题。然而在智能 Monkey 中，由于加入了智能控制，使得测试不会卡在某一界面长时间不变，并且通过调整动作比例来减少无意义动作，最终发现在相同时间段内，智能 Monkey 不会产生测试时“卡死”现象，并且智能 Monkey 遍历的界面明显多于基础 Monkey，智能 Monkey 的无意义动作比例明显少于基础 Monkey，所以，总而言之，智能 Monkey 测试的效率明显高于基础 Monkey。

## 5. 总结与反思

本次实验是我第一次接触游戏测试方面的实践，过程中遇到许多困难和问题，但在解决问题的过程中我也收获了许多。问题与收获总结如下：

（1）由于只有 Window 电脑和苹果手机，难以进行实验，经网络搜索资料，学到可以安装使用安卓模拟器来达到与真机相同的效果，于是我尝试使用安卓模拟器的开发者模式，顺利解决了设备的问题。

（2）安装 JDK 的过程比较顺利，但是安装 SDK 的过程却有些曲折，一开始由于网络问题难以安装成功，后来进行科学上网，也顺利解决此问题。但是原装

SDK 与夜神安卓模拟器的连接又出现了问题，就是刚连接好就会中断，经过搜索解决方案，将 SDK\platform-tools 下的 adb.exe 文件，替换成夜神模拟器 bin 目录下的 adb.exe 文件，就顺利解决了此问题。

(3) 智能 Monkey 实现的过程对我来说是最复杂的，虽然参考了 <https://github.com/TangHuaiZhe/monkeyTest> 中的思路，但是因为其中用到许多我不熟悉的操作，如直接用 python 程序控制系统执行 adb 命令，并获取 adb 命令的反馈到程序中进行进一步处理，所以搞懂这些颇费一番时间，但是我从中学会了一个思路，就是智能 Monkey 的实现，不仅仅可以通过直接控制 Monkey 的指令来实现，还可以通过在执行 Monkey 测试时，对其进度进行监测，并根据监测结果来进行及时的智能干预和智能控制，就能够以此达到智能 Monkey 的实现。

(4) 智能 Monkey 的实现也需要根据测试的 app 的不同特点来进行适当微调，比如本实验中的目标应用是新浪微博，所以可以将判断“卡死”的限定时间设置成较短的几秒，并且 Monkey 测试的不同动作的比例也适当调整为接近于用户使用新浪微博时的动作比例。如果测试的是其他如游戏之类的应用，由于游戏界面加载时间比新浪微博更久，所以可能需要将判断“卡死”时间设置更长一些，动作比例也需要调整。

(5) 本次实验能够顺利解决基础 Monkey 测试中遇到的遍历界面有限、无意义动作较多、容易在某界面“卡死”等问题，但是，针对环路问题的解决方案实现还有待接下来的进一步研究。

## 6. 参考网页

<https://zhuanlan.zhihu.com/p/97335363>

[https://blog.csdn.net/just\\_like/article/details/83757492](https://blog.csdn.net/just_like/article/details/83757492)

<https://blog.csdn.net/xx326664162/article/details/52385720>

[https://blog.csdn.net/qg\\_36350532/article/details/79000653](https://blog.csdn.net/qg_36350532/article/details/79000653)

[https://blog.csdn.net/qq\\_36155340/article/details/78114040](https://blog.csdn.net/qq_36155340/article/details/78114040)

<https://github.com/TangHuaiZhe/monkeyTest>