



Android* 资源



自动 Android* 应用测试

作者：[Roman Khatko \(Intel\)](https://software.intel.com/zh-cn/user/780674) (<https://software.intel.com/zh-cn/user/780674>)，更新时间：2014 年 2 月 16 日

[翻译](#)

测试是应用开发流程中的重要环节。对于 Android，这尤为重要，因为 Android 设备彼此之间差异很大，主要体现在以下几个方面：

- 屏幕尺寸和分辨率
- Android 版本
- 外形
- 处理器的指令集
- 是否有前置摄像头、NFC、外置键盘等。

您应在多台设备上对 Android 应用进行测试。

应用测试流程包括多种测试。让我们来了解一下手动功能测试。测试者需要认真检查所有功能并将设备重置为初始状态。测试者在每个应用和每部设备上重复上述操作。该流程是手动完成的，因此非常耗时。

自动功能测试可按计划执行而无需额外成本。例如，测试一个 build，每天晚上在所有设备上进行测试，早上分析结果并修复漏洞。

在本文中，我们将回顾几种适用于自动功能测试的工具。我仅对 Android SDK 中包括的或在开源许可协议下发布的工具进行介绍。

自动测试的概念

我们的目标是将手动执行的操作自动化，以实现最大精度。让我们了解一下这些操作。我们将使用多个应用和多部 Android 设备。

对于每个应用和每部设备，我们应按照下列步骤来操作：

1. 在设备上安装应用
2. 启动应用
3. 使用指定方法测试应用
4. 卸载应用
5. 将设备重置为初始阶段

在每个步骤中，您需要收集和分析数据，如日志和截屏。在下文中，我们将讨论将这些步骤自动化的工具。

控制 Android 设备

首先，您需要选择用于运行自动化测试的电脑，并在该电脑上安装 Android SDK。我将以运行 Linux* 系统的台式电脑为例。您需要在每台设备上禁用锁屏并将“进入睡眠模式的时间”调整到最大值。对于一些方法 您需要禁用屏幕方向调整。

Android SDK 中有两种实用程序可以控制 Android 设备：ADB 和 monkeyrunner*。下面，我将具体介绍如何将手动测试中的操作自动化。

使用 ADB 控制 Android 设备

ADB（Android Debug Bridge）是控制 Android 设备的命令行工具。ADB 主页是：

<http://developer.android.com/tools/help/adb.html> (<http://developer.android.com/tools/help/adb.html>)

ADB 工具位于 <android_sdk>/platform-tools/ 目录下。您需要将该目录存放到 PATH 环境变量下。

检查 ADB 安装

安装并设置 Android SDK，然后将 Android 设备连接至您的电脑并运行下列命令：

```
adb devices
```

该命令将出现在所有插入的 Android 设备上。如果设备列表不是空白，那么 ADB 便可正常运行。

在多台设备上运行

您需要使用“-s”参数指定 ADB 应使用哪台设备。

```
adb -s [serial_number] [command]
```

例如：

```
adb -s [serial_number] logcat
```

设备的序列号可通过输入 «adb devices» 命令后生成的输出中获得。参数 -s 支持您同时使用多台互联设备。

基本的 ADB 命令

在设备上打开控制台：

```
adb shell
```

在设备上运行命令：

```
adb shell [command]
```

Android 中包含许多标准的 Linux 实用程序：ls、cat、dmesg ...

从 apk 文件安装应用：

```
adb install example.apk
```

卸载应用：

```
adb uninstall [package]
```

从 apk 文件获取程序包名称：

```
aapt dump badging example.apk | grep "Package"
```

从设备将文件下载至电脑：

```
adb pull [path-on-device] [file]
```

从设备将文件上传至电脑：

```
adb push [file] [path-on-device]
```

注：

Android 设备上的大多数目录仅支持读取访问。 /sdcard（但是您无法从该目录下运行程序）和 /data/local/tmp 支持写入访问。

启动应用：

```
adb shell am start -n [package]/[activity]
```

运行指定活动。

您可以从 apk 文件抽取活动名称：

```
aapt dump badging example.apk | grep "launchable-activity"
```

读取日志

Logcat 是从 Android 设备读取日志的命令。

Logcat 主页：<http://developer.android.com/tools/help/logcat.html>
(<http://developer.android.com/tools/help/logcat.html>)

从设备读取日志（按下 Ctrl-C 解除阻止）：

```
adb logcat
```

清除设备上的日志缓存：

```
adb logcat -c
```

将日志缓存转储到设备上（显示当前的缓存内容，未阻止）：

```
adb logcat -d
```

示例：

```
1 | adb logcat -c # clear the buffer log
2 | # Action
3 | adb logcat -d > file.log # save the current contents of the log buffer
```

```
| to file.log
```

使用 screencap 截取屏幕

screencap 实用程序将当前的屏幕内容保存为图形文件：

```
1 adb shell screencap /sdcard/screen.png
2 adb pull /sdcard/screen.png screen.png
3 adb shell rm /sdcard /screen.png
```

screencap 实用程序可在安装了 Android 4.x 和更高版本的手机上使用。在低于 Android 4.x 的版本上，您可以使用 monkeyrunner 来截取屏幕。

使用 ADB 运行 BASH 脚本来测试应用

脚本：[app_test.sh \(http://software.intel.com/sites/default/files/article/365437/app-test.zip\)](http://software.intel.com/sites/default/files/article/365437/app-test.zip)

使用 MonkeyRunner 控制 Android 设备

monkeyrunner 工具可为脚本提供控制 Android 设备的 API。您可以使用 monkeyrunner 编写 Python* 脚本来安装、启动 Android 应用，模拟用户操作，获取截图并将其保存至电脑。Monkeyrunner 使用 Jython* 运行脚本。

monkeyrunner 主页和 API 参考：http://developer.android.com/tools/help/monkeyrunner_concepts.html
(http://developer.android.com/tools/help/monkeyrunner_concepts.html)

使用 monkeyrunner 读取日志

File log.py：

```
01 | # coding: utf-8
```

```
02 from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
03
04 def log(fn, device):
05     msg = device.shell('logcat -d')
06     f_log = open(fn, 'at')
07     if msg is None:
08         msg = 'None'
09     f_log.write(msg.encode('utf-8'))
10     f_log.close()
11     device.shell('logcat -c')
12
13 if __name__ == '__main__':
14     device = MonkeyRunner.waitForConnection()
15     device.shell('logcat -c') # Clear logs buffer
16     # ...
17     log('example.log', device) # Write logs
```

开始：

monkeyrunner log.py

脚本将会把日志写入当前目录下名为“文件示例.log”的文件。

使用 MonkeyRunner 捕捉截图

File screen.py：

```
1 # coding: utf-8
2 from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
3
4 if __name__ == '__main__':
5     device = MonkeyRunner.waitForConnection()
6     image = device.takeSnapshot()
7     image.writeToFile('screenshot.png', 'png')
```

开始：

```
monkeyrunner screen.py
```

脚本捕捉截图并在当前目录下将其保存为名为 filescreenshot.png 文件。

示例：使用 monkeyrunner 控制设备

脚本：[monkeyrunner_test.py \(http://software.intel.com/sites/default/files/article/365437/monkeyrunner-test.zip\)](http://software.intel.com/sites/default/files/article/365437/monkeyrunner-test.zip)

开始：

```
monkeyrunner monkeyrunner_test.py
```

自动测试方法

使用 Monkey* 进行测试

试想将正在测试的设备交给一只非常敏捷且极富创造力的猴子 — Monkey（猴子）工具的设计便是模拟这种情形。Monkey 工具是 Android SDK 的组成部分，可发送一连串的随机用户事件。命令行参数可指定用户操作次数、每类事件的比率和程序包的名称（因此，Monkey 的操作不会超出所测试应用的限度，且不会向地址簿中的任何联系人发送 SMS）。

Monkey 主页上提供了诸多示例和参数列表：<http://developer.android.com/tools/help/monkey.html>
(<http://developer.android.com/tools/help/monkey.html>)

Monkey 工具的主要优点是零维护成本。而且，压力测试还可检测出 non-trivial 漏洞。

使用 Monkey 工具进行测试的缺点：

- Monkey 无法模拟复杂的工作负载，如身份验证。因此，应用的功能性无法进行测试。
- 需要复杂控制（快速反应和复杂手势）的游戏应在开始时完成，或不要执行该类应用的测试。

- Monkey 发现的错误很难再现。
- Monkey 无法在测试过程中检查应用状态。

对于任何应用而言，开始都可以先使用 Monkey 进行自动测试。这种方法可为具体应用提供充足的结果。如果测试质量较低，您需要使用其他测试方法。

使用 MonkeyRunner 进行测试

使用 MonkeyRunner，您不仅可以开发 Android 设备控制脚本，还可以编写脚本在特定设备上测试应用。

优点：

- 灵活。

缺点：

- 编写脚本较复杂 — 即使是在简单的情况下。

开发 MonkeyRunner 需要花费很长的时间，因此通常这种方法不太理想。但是在某些情况下可以使用这种方法。

使用 getevent 和 sendevent 进行测试

Getevent 和 sendevent 实用程序支持用户记录事件序列并将其再现。运行这些工具不需要根许可。

优点：

- 在手动测试下，无需花费额外成本便可记录事件序列（如果执行）。
- 记录事件序列不需要编程技能。

缺点：

- 需单独为每个应用和每部设备记录序列。如果您更改了一个应用界面，需要重新记录所有已记录的操作。
- 这种方法无法在测试过程中检查应用状态。如果应用响应延迟（如网页加载），测试结果将出现错误。
- 演示快速、复杂的序列所需的时间比记录的时间长。因此，有时候这种方法不适合测试响应时间较为关键的动态游戏（dynamic game）。

记录事件序列：

```
01 # Record event sequence
02 # Do actions on the device, press Ctrl-C to finish
03 adb shell getevent -t > events.txt
04 # Convert event sequence to script
05 ./decode_events.py events.txt > events.sh
06 # Load the script to the device
07 adb push events.sh /data/local/tmp/
08 # Set the permissions
09 adb shell chmod 755 /data/local/tmp/events.sh
10 # Run script
11 adb shell sh /data/local/tmp/events.sh
```

脚本：[decode_events.py \(http://software.intel.com/sites/default/files/article/365437/decode-events.zip\)](http://software.intel.com/sites/default/files/article/365437/decode-events.zip)

在设备上再现记录的事件序列。

使用 Robotium* 进行测试

Robotium 未包含在 Android SDK 中，但是它是在开源协议下发布的产品。Robotium 主页是：
<http://code.google.com/p/robotium/> (<http://code.google.com/p/robotium/>)。

Robotium 脚本可在应用 UI 层面而非输入设备层面定义操作。

例如，脚本需要点击 «OK» 按钮。monkeyrunner 脚本将按照“点击屏幕点 (x0, y0)”执行。Robotium 脚本将按照“按下带有文本“OK”的按钮”执行。

当在界面层面描述操作时，测试脚本将不受界面布局、界面分辨率和方向的影响。

此外，Robotium 还支持您检查应用对操作的响应。例如，点击 «OK» 按钮后，将会出现带有“Item 1”的列表项目。您还可以使用 Robotium 查看列表元素的名称。如果您可以在每一步之后检查应用状态，便很容易找到错误出现在哪一步。

缺点：

- 您需要在 Java* 中为每个应用开发一个测试脚本。这需要编程技巧，也较为费时。
- 应用界面更改后，必须重新安排事件序列。
- 相比使用 getevent / sendevent 而言，编写 Robotium 脚本更为困难。

一般而言，Robotium 支持您以适度的成本开发最高质量的测试案例。

测试方法比较

测试方法	优点	缺点
Monkey — 一连串随机用户操作	无维护成本。 不受设备影响。 压力测试可检查 non-trivial 错误。	测试质量在不同的应用上会有所差异。 无法再现缺陷报告。 Monkey 无法在测试过程中检查应用状态。
monkeyrunner — 设备控制脚本	灵活性	编写脚本较复杂 — 即使针对简单的应用。

测试方法	优点	缺点
getevent/sendevent — 记录/再现用户操作	记录事件序列不需要编程技能。	记录的操作序列仅可在一台设备上以固定的界面方向执行。 应用界面更改后，必须重新安排事件序列。 这种方法无法在测试过程中检查应用的状态。
Robotium — 测试脚本 API 以验证状态	操作在应用 UI 层面描述。 脚本可能不会受到界面分辨率和界面方向的影响。 脚本可在一个操作后检查应用状态。	在 Java 编写脚本较复杂。 如果您更改了应用界面，将需要修改脚本。

结果分析

现在，我们需要分析在自动错误测试流程过程中收集的日志和截屏。

日志分析

您可以搜索一下以下的字符串：

- I/DEBUG
- FATAL EXCEPTION
- WIN DEATH

您可以在该列表中添加在手动测试中发现的错误消息。

截屏分析

您可以准备一系列测试关键时刻的截屏，并在自动测试时将其与屏幕内容进行对比。这可以确定自动测试流程是否正常运行。

将最初的截屏与应用启动后的截屏进行对比非常有用。它可以在应用出现静默故障时检查事件。

Monkeyrunner 支持您按照指定的容错性（百分比）对比两个截屏：

```
1 image1 = device.takeSnapshot()
2 # ...
3 image2 = device.takeSnapshot()
4 if image2.sameAs(image1, 0.1):
5     print 'image1 and image2 are the same (10%)'
```

很遗憾，没有 MonkeyImage API 可用于加载文件中的图片。您可以使用，如 [Python* Imaging Library](http://www.pythonware.com/products/pil/) (<http://www.pythonware.com/products/pil/>) 编写一个自定义功能来对比图片。

将设备重置为初始状态

您应在测试后将设备重置会初始状态。这可以通过下列几种方法来实现：

- 多次按下 «Back» 按钮。
- 重启设备。
- 重启 zygote 流程。

通常情况下，第一个选项最合适。

多次按下 Back 按钮

使用 monkeyrunner 按 “Back” 按钮：

```
1 for i in xrange(0, 10):
2     device.press('KEYCODE_BACK', MonkeyDevice.DOWN_AND_UP)
```

```
3 | time.sleep(0.5)
```

在实际情况下这是较好的选择，因为它不需要用户实际操作。

结论

在本文中，我们介绍了几种针对 Android 应用的自动测试方法。我们回顾了自动测试方法的优点和缺点。

此外，我们还讨论了 Android SDK 中的 Monkey 和 monkeyrunner 工具以及 Robotium 工具。

自动测试不能取代其他类型的测试。正确的有组织的测试流程（结合包括自动测试在内的不同测试方法）是高质量应用开发流程的必要部分。

有关编译器优化的更完整信息，请参阅[优化通知 \(/zh-cn/articles/optimization-notice#opt-cn\)](https://software.intel.com/zh-cn/articles/optimization-notice#opt-cn)。

附件	大小
 app-test.zip (https://software.intel.com/sites/default/files/article/365437/app-test.zip)	571 字节
 decode-events.zip (https://software.intel.com/sites/default/files/article/365437/decode-events.zip)	815 字节
 monkeyrunner-test.zip (https://software.intel.com/sites/default/files/article/365437/monkeyrunner-test.zip)	756 字节

- [资源和设计中心](#)
- [购买英特尔产品](#)
- [固件](#)

○ 管理工具

- [下载中心](#)
- [在线服务中心](#)
- [注册中心](#)

- [01.org](#)
- [Clear Linux* 项目](#)
- [Zephyr 项目](#)

○ 随时掌握最新信息

- [论坛](#)
- [最近更新](#)
- [订阅我们的 YouTube 频道](#)
- [新闻简报存档](#)



关注我们：



英特尔公司 京ICP备 14036123号-1 [使用条款](#) [*商标](#) [隐私条款](#) [Cookie](#)