App 测试实训教程

V3.0

目录

1 APP 基础

1.1 什么是 APP

APP(应用程序，外语缩写：App，外语全称：Application) 。APP 指的是智能手机的第三方应用程序。现在主流的 APP 应用格式有：

- 苹果的 iOS 系统，app 格式有 ipa，pxl，deb，这里的 APP 都是用在 iphone 系列的手机和平板电脑上。
- Android 格式：apk， 这里 APP 主要用在使用安卓系统的智能手机上，这类的手机在中国市场上的占用率最高。
- 微软的 WindowsPhone7、WindowsPhone8 系统，app 格式为 xap。我们可以在应用市场下载。假如你是开放者，你也可以开发出 APP 提交到 APP 应用商店供别人下载。通告收费下载，或者在你的 APP 里面放置广来盈利。

目前比较著名的 App 商店有 Apple 的 iTunes 商店里的 App Store;android 的 Google Play Store、360 手机应用商店、百度 Android 应用中心、豌豆荚、91 手机助手等。APP 创新性开发，始终是用户的关注焦点。与趋于成熟的美国市场相对比，我国开发市场正处于高速生长阶段。对于很多有志于开发手机应用的朋友来说，他们最关心的是开发各种 APP 所要使用的编程语言，现在开发手机 APP 所要使用的主要编程语言有：

- iOS 平台开发语言为 Objective-C; Swift
- 安卓 Android 开发语言为 java;
- 微软 Windows phone7 开发语言是 C#;

1.2 APP 技术

目前主流应用程序大体分为三类：Web App、Hybrid App、 Native App。

首先，我们来看看什么是 Web App、Hybrid App、 Native App。

1. Web APP

Web App 指采用 Html5 语言写出的 App，不需要下载安装。类似于现在所说的轻应用。生存在浏览器中的应用，基本上可以说是触屏版的网页应用。WebApp 是基于 Web 的系统和应用，其作用是向广大的最终用户发布一组复杂的内容和功能。说得简单一点就是因为移动互联网特别火爆，很多企业公司也都想拥有一个属于自己的 app，但是因为原生 app 开发的成本比较高，而且后期维护比较困难，所以就寻求一种方式--web app！所谓的 web app 就是给 web 站打了个包加了个壳，我们看起来像是一个 app 可以上传到应用商店，可以从上面下载，但是我们在手机上打开之后看到的实际上还是网页，只不过写成的是自适应的网页能够在手机上显示的也比较好，能够唬人，做到跟原生 app 神似！但是 web app 获取不到手机里面的底层功能，比如说打开摄像头、打开相册、获取我们的地理位置信息、支付等都是做不到的。如果你想开发一款 app 需要用到原生底层能力的话，web app 是满足不了的，如果你想开发一款 app 上传到苹果的应用商店的话是不行的，苹果审核是非常严格的，因为在苹果看来 web app 根本都不是属于真正意义上的 app，作为企业产品展示还可以。

Web app 的优点：

A.开发成本低；

B.更新快；

C.更新无需通知用户，不需要手动升级；

D.能够跨多个平台和终端；

Web app 的缺点：

A.临时性的入口；

B.无法获取系统级别的通知，提醒，动效等等；

C.用户留存率低；

D.用户体验较差；

Web app 中的 H5 技术

H5 是一种超文本标记语言，需要 web 解释器对语言进行翻译，也就是说它必然要依靠 web 解释器（例如浏览器）,而浏览器依靠的是 Android 系统，所以，在 H5 解释器这方面的 app（或者说一种对于 H5 起支撑作用的 App）是不会被取代的。H5 性能需要改进，不论是电脑还是手机，它和原生的性能差距很大。很多原生的功能，H5 是做不到的，因为它不能直接调用底层硬件。H5 其本质是网页，换页时，基本要加载整个页面，就像是浏览器打开一个新页面一样，展示会比较慢，而原生系统则只加载变化的部分。如果 APP 用户常见页面频换（比如一些活动页面）那么用 H5，维护起来更容易。

2. Native App

Native APP 指的是原生程序，一般依托于操作系统，有很强的交互，是一个完整的 App，可拓展性强。需要用户下载安装使用。

优点：

（1）打造完美的用户体验

（2）性能稳定

（3）操作速度快，上手流畅

（4）访问本地资源（通讯录，相册）

（5）设计出色的动效，转场，

（6）拥有系统级别的贴心通知或提醒

（7）用户留存率高

缺点：

（1）开发成本高（不同平台用不同的开发语言和界面适配）

（2）维护成本高（例如一款 App 已更新至 V5 版本，但仍有用户在使用 V2， V3， V4 版本，需要更多的开发人员维护之前的版本）

（3）更新缓慢，根据不同平台，提交 - 审核 - 上线 等等不同的流程，需要经过的流程较复杂

3. Hybrid App

Hybrid APP 指的是半原生半 Web 的混合类 App。因为采用了原生应用的一部分、Web 应用的一部分，所以必须是部分在设备上运行、部分在 Web 上运行。Hybrid app 需要下载安装，看上去类似 Native App，但只有很少的 UI Web View，访问的内容是 Web 。例如 Store 里的新闻类 APP，视频类 APP 普遍采取的是 Native 的框架，Web 的内容。Hybrid App 极力

去打造类似于 Native App 的体验，但仍受限于技术，网速，等等很多因素，尚不完美。对于固定格式、页面速度要求高的模块采用原生开发，对于新闻、大段文字、资讯类的用 H5 页面来加载，将其嵌入到原生框架中可以达到比较良好的体验。混合应用中比例很自由，比如 Web 占 90%，原生占 10%，或者各占 50%都可以。

4. Web App、Hybrid App、Native App 技术特性

	Native	Html5	Hybrid
APP 特性			
图像渲染	本地API渲染	Html，Canvas，CSS	混合
性能	快	慢	慢
原生界面	原生	模仿	模仿
发布	App Store	Web	App store
本机设备访问			
照相机	支持	不支持	支持
系统通知	支持	不支持	支持
定位	支持	支持	支持
网络要求			
网络要求	支持离线	大部分依赖网络	大部分依赖网络

Native 是使用原生系统内核的，相当于直接在系统上操作。是我们传统意义上的软件，更加稳定。但是 H5 的 APP 先得调用系统的浏览器内核，相当于是在网页中进行操作，较原生 APP 稳定性稍差，似乎还没有百万级用户量的 H5 APP。但是 H5 最大的优点是可以跨平台，开发容易。native 需要用 ANDROID 的语言和 IOS 的语言各自写，H5 只要开发一套

1.3 APP 页面类型

APP 页面的四种类型：

用户打开 APP，是为了完成某项任务。打开电商类 app 是为了购物或者查看物流信息；打开简书为了写文章或者浏览文章；打开微信是为了查看朋友圈、即时聊天.....而这些任务都是基于 APP 的每个页面去完成的。不同的页面有着不同的作用，按照作用划分，可以将 APP 页面大致分为四种类型：聚合类、列表类、内容类、功能类。

- **聚合类**

聚合类多见于 APP 的首页，用于功能入口的聚合展示。例如京东和得到 APP 将很多功能聚合在首页，用户可以在首页找到想要功能入口，点击就能进入该功能模块。聚合类相当于

分流的作用，用户打开 APP，进入首页，再通过首页的各个功能入口进入到其他的页面。在设计聚合类页面的时候，根据用户的需求和产品功能的优先级进行排列，将优先级高的功能入口放在靠前的位置。

- **列表类**

列表类是 APP 最常见的页面类型，目的是展示同类别的信息，供用户筛选。虽然我们将这类页面命名为列表页，但是其展示形式是多样的，可以是列表的形式、也可以是网格的形式、还可以是卡片的形式。它们的本质都是展示多条信息，供用户选择。当设计列表页时，应根据展现信息的不同特点选择恰当的展示形式（列表型、网格型、卡片型），或者干脆提供多种浏览方式来供用户选择。淘宝、京东的商品列表页提供了列表和网格两种展示方式。同时将用户关心的维度放在列表中展示出来（如标题、图片、销量、价格等），用户会根据这些维度的信息来进行选择，从而决定要不要点击进入下级页面。

- **内容类**

打开 APP 进入京东首页（聚合页），搜索“电脑”进入搜索结果页（列表页），在列表页看中了苹果的“iMac”，点击进入商品详情页（即内容页）。内容页是用来展示具体信息的页面，购物 APP 中的商品详情页，读书 APP 中的书籍阅读页，资讯 APP 中的正文页.....这些都属于内容页。值得注意的是，一个页面可能会包含两种不同的页面类型，例如美团首页，第一屏是聚合页，下面则是列表页。

- **功能类**

为了完成某个功能而存在的页面，常见的有发布状态页面、收货信息的填写页面等等。

按照作用把页面分为这四类有什么用呢？

1. 目前的产品都是由一个个页面组成的，知道了页面的四种类型，会加深你对页面的理解。
2. 多一个角度思考产品设计，可以根据聚合类、列表类、内容类、功能类，分别思考每个类型的页面应该有什么样的特点，测试时应该注意什么。

2 App 测试基础

2.1 什么是 APP 测试

移动端测试是指对移动应用进行的测试，即实体的特性满足需求的程度。比如针对手机，ipad 等平台上的各种 app 功能和性能展开的测试。相较于传统的 web 端、PC 客户端产品的测试，移动端的测试受手机屏幕大小、内存、CPU、网络特性，操作系统、用户使用习惯的差异，有其自身的特点，所以对移动端产品测试就需要充分考虑测试差异而单独分列出来。App 测试就是符合多种网络，不同系统不同分辨率下发现软件缺陷，并保证提高软件质量的过程。

2.2 APP 测试究竟关注什么

- 业务逻辑
  - 不同登录状态的访问权限
  - 不同操作后页面的跳转
  - 与其他业务逻辑的关联
  - 操作时返回上一步
  - 操作错误情况
- 数据显示
  - 存在多条数据时的上拉与下拉加载显示
  - 筛选数据后的上拉与下拉加载显示
  - 列表页与详情页数据是否显示相同
  - 图片在详情页的显示
  - 文本信息在详情页的显示
  - 数据列表的排序
  - 无数据情况下界面的显示
- 手机权限与应用本身
  - 有权限（如：拍照、存储）时的运行与无权限时的运行
  - 第三方平台安装
  - 卸载情况
  - 异常退出后的重新运行
- 上传/下载文件的格式
  - 上传文件/图片的格式限制
  - 上传文件/图片的大小限制
  - 下载文件超过手机剩余存储空间的情况
- 输入数据
  - 输入特殊字符或标签的情况
  - 输入空值的情况
  - 输入字符串长度超过限制
  - 输入数据不符合业务显性或隐性的需求（如：价格输入负数或 0 等）
  - 数据的边界值情况
- 控件
  - 点击按钮页面的跳转情况
  - 多次点击按钮的情况
- 中断
  - 支付过程中中断操作
  - 安装过程中中断操作
  - 应用运用时锁屏
  - 杀死进程后的再次进入 App 的情况
- 界面
  - 文字描述的正常性

- 界面显示情况（如：界面显示错位或与效果图不符）

➢ 其他

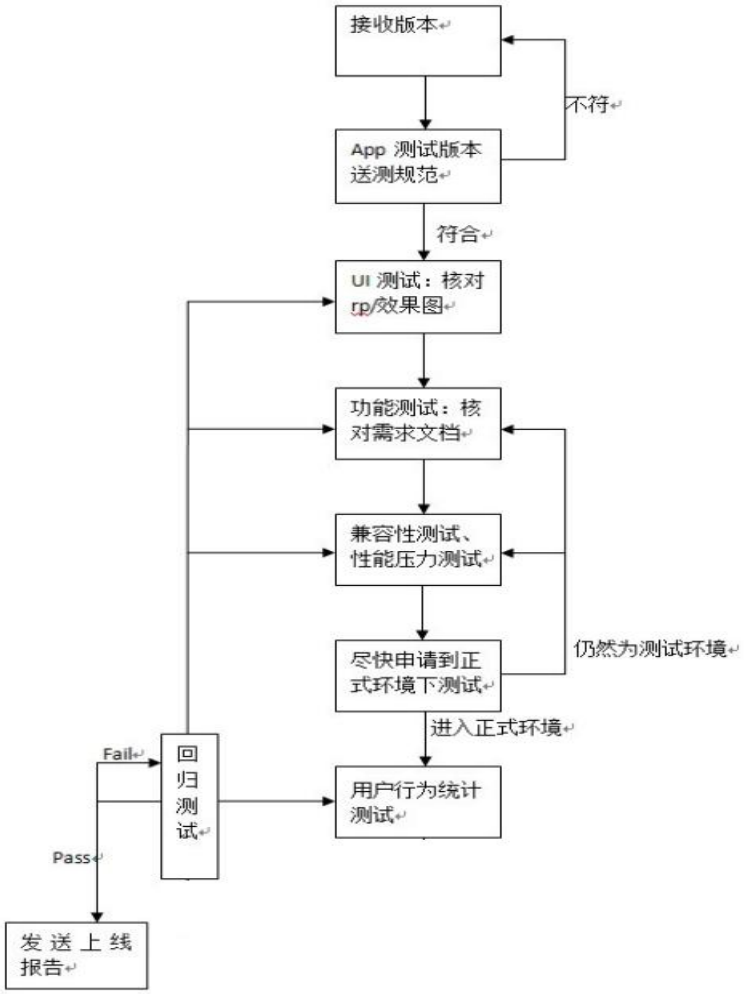
- 推送的方式，次数
- 测试客户端与 pc 端的交互
- pc 端与客户端数据一致

2.3 APP 测试流程

测试任务开始前，要检查各项测试资源，如产品功能需求文档，产品原型图及产品效果图等。其他的，例如有秒杀专题的项目、需要规划秒杀时间表、有优惠券使用的项目、需要申请添加优惠券数据、支付宝/银联支付功能的项目、需要提前申请支付宝/银联账户等等。

- 产品功能需求文档；
- 产品原型图/产品效果图；
- 行为统计分析定义文档；
- 测试设备（IOS/Android）

APP 测试流程图



- **接收版本**

A)接收测试版本的同时，需要查看程序是否符合《app 测试版本提交质量规范》，若符合则开始测试任务，若不符合规范，可拒绝测试。

B)日常接收版本时需要注意测试版本规范，如不符合，请开发人员重新修改合适的版本后再次提交测试

- **UI 测试**

A）确保手头的原型图与效果图为当前最新版本。

B）确保产品 UI 符合产品经理制定的原型图与效果图。

C）一切界面问题以效果图为准，若有用户体验方面的建议，必须先以邮件或口头的形式询问产品经理。

D）由于测试环境中的数据为模拟数据，测试时必须预先考虑到正式环境中可能出现的数据类型。

- **功能测试**

A） 确保手头的功能需求文档为当前最新版本。

B） 确保所有的软件功能都已实现且逻辑正常。

C） 一切功能问题以需求文档为准，若有用户体验方面的建议，必须先以邮件或口头的形式询问产品经理。用户体验方面的建议，优先级放在修复 bug 之后。

D）若有些功能在技术上难以实现或者由于排期的原因无法在短时间内实现，必须得到产品经理的确认，而不是单单只听开发人员的技术解释。此处确认最好以邮件形式存在。

E）所有的“外部原因”问题，都需要尽早地督促开发人员与客户服务端人员联系协调解决。并在之后的测试报告中予以体现。

F）所有的“设计如此”、“延期处理”问题，都需要和产品经理确认后再进行验证。并在之后的测试报告中予以体现。

G）测试下单时，注册的测试账号必须符合公司规范；收货地址必须包含“测试”关键字，最好每次下单的名称中含有日期，以便查询；在正式环境中下单后必须取消该订单等。

- **兼容测试/性能测试**

A） 确保软件在所有兼容机型上都能正常使用；

B） 对于低端性能兼容机上独有的问题，若在技术上难以修改或者由于排期的原因无法在短时间内改进，必须在测试日报中注明，并得到技术平台主管、产品经理以及运营人员的确认，最好以邮件的形式得到确认；

C） 性能测试方面必须满足硬件压力条件下的测试需要（例如多线程，用户常用的 app 都要后台运行的环境中测试。）

D） 网络响应用户体验方面的性能测试，需要保证在 WIFI、4g、3g 网络下的切换效果。比如 WIFI 切换到 3g，网络响应的速度以及切换界面。

- **用户行为统计测试**

A）确保产品经理在文档中所定义的页面在该产品中都是存在的。

B）尽可能真实地模拟用户行为。

C）核对统计日志，确保各项操作所对应的页面 ID 以及操作 ID 都是正确的。

- **回归测试**

A）软件最终上线前，需对产品进行回归测试，测试内容包含之前所有的测试项目

B）回归测试不再对细节进行测试，而是类似于对产品进行验收，从客户正常使用的角度对产品进行再一轮的整体测试。

C）只有在回归测试通过之后，才对产品进行提交。

- **测试日报及产品上线报告**

产品上线前，测试人员每天需对所测项目发送测试日报。

测试日报所包含的内容为：

A）对当前测试版本质量进行分级。

B）对较严重的问题进行例举，提示开发人员优先修改。

C）对版本的整体情况进行评估。

App 的测试周期一般为两周，根据项目情况以及版本质量可适当缩短或延长测试时间。正式测试前先向主管或产品经理确认项目排期

2.4 APP 测试要点

APP 测试的时候，建议让开发打好包 APK 和 IPA 安装包，测试人员自己安装应用，进行测试。在测试过程中需要注意的测试点如下：

2.4.1 功能测试

根据软件说明或用户需求验证 App 的各个功能实现，采用如下方法实现并评估功能测试过程：

1）采用时间、地点、对象、行为和背景五元素或业务分析等方法分析、提炼 App 的用户使用场景，对比说明或需求，整理出内在、外在及非功能直接相关的需求，构建测试点，并明确测试标准。

2）根据被测功能点的特性列出相应类型的测试用例对其进行覆盖，如：设计输入的地方需要考虑等价、边界、负面、异常、非法、场景回滚、关联测试等测试类型对其进行覆盖。 3）在测试实现的各个阶段跟踪测试实现与需求输入的覆盖情况，及时修正业务或需求理解错误。

1．运行

1) App 安装完成后的试运行，可正常打开软件。
2) App 打开测试，是否有加载状态进度提示。
3) App 页面间的切换是否流畅，逻辑是否正确。
4) 注册 ：同表单编辑页面；用户名密码长度；注册后的提示页面；前台注册页面和后台的管理页面数据是否一致；注册后，在后台管理中页面提示；
   登录：使用合法的用户登录系统；系统是否允许多次非法的登录；是否有次数限制；使用已经登录的账号登录系统是否正确处理；用户名、密码，错误或漏填时能否登录；删除或修改后的用户，原用户名登录 ；不输入用户口令和重复点“确定/取消”按钮，是否允许登录 登录后，页面中登录信息页面中有注销按钮 登录超时的处理
5) 注销: 注销原模块，新的模块系统能否正确处理、终止注销能否返回原模块，原用户、注销原用户，新用户系统能否正确处理、使用错误的账号、口令、无权限的被禁用的账号进行注销

2. 应用的前后台切换

A） App 切换到后台，再回到 App，检查是否停留在上一次操作界面,比如提示框是否还存在，有时候会出现应用自动跳过提示框的缺陷。

B）App 切换到后台，再回到前台时，注意程序是否崩溃，功能状态是否正常，尤其是对于从后台切换回前台数据有自动更新的时候

C）手机锁屏解锁后进入 App 注意是否会崩溃，功能状态是否正常。

D）当 App 使用过程中有电话进来中断后再切换到 App，功能状态是否正常

E）当杀掉 App 进程后，再开启 App，App 能否正常启动 。

F）对于有数据交换的页面，每个页面都必须要进行前后台切换、锁屏的测试，这种页面最容易出现崩溃。

3. 免登录

很多应用提供免登录功能，当应用开启时自动以上一次登录的用户身份来使用 App。

考虑无网络情况时能否正常进入免登录状态。

A) 切换用户登录后，要校验用户登录信息以及数据内容是否相应更新，确保原用户退出。

B）设备同时登录的情况，通常一个账户只允许登录一台机器。所以，需要检查一个账户登录多台手机的情况,原手机里的用户需要被退出，给出友好提示。

C) 密码更换后，检查有数据交换时是否进行了有效身份的校验。

D) 支持自动登录的应用在进行数据校验时，检查系统是否能自动登录成功并且数据操作无误。

E) 检查用户主动退出登录后，下次启动 App，应停留在登录界面。

4. 离线浏览

很多应用会支持离线浏览，即在本地客户端会缓存一部分数据供用户查看。

A) 在无线网络情况可以浏览本地数据。

B) 退出 App 再开启 App 时能正常浏览。

C) 切换到后台再回到前台可以正常浏览。

D) 锁屏后再解锁回到应用前台可以正常浏览。

E) 在服务器端数据更新时会给予离线的相应提示。

5. 定位、照相机服务

A) App 有用到相机，定位服务时，需要注意系统版本差异。

B) 有用到照相机服务的地方，需要进行前后台的切换测试，检查应用是否正常。

C) 测试照相机服务时，需要采用真机进行测试。

功能测试要点：

- 多分辨率测试
- 多系统测试
- 用户不同的使用习惯
- 网络的不稳定性
- 安装/卸载测试
- 升级测试
- 并发测试
- 数据来源
- 推送
- 分享跳转

2.4.2 UI 测试

1、测试用户界面（如菜单、对话框、窗口和其他控件）布局、风格是否满足要求、文字是否正确、页面是否美观、文字、图片组合是否完美、操作是否友好等。

2、UI 测试的目标是确保用户界面会通过测试对象的功能来为用户提供相应的访问或浏览功能。确保用户界面符合公司或行业的标准。包括用户友好性、人性化、易操作性测试。

1. **导航测试**

A）按钮、对话框、列表和窗口等；

B) 在不同的连接页面之间需要导航;

C) 是否易于导航，导航是否直观；

D) 是否需要搜索引擎；

E) 导航与页面结构、菜单、连接页面的风格是否一致；

Android 应用导航主要分为以下三类：

A) 应用程序内的导航

- 通过多个入口进入到界面
- 界面内视图间的导航
- 同一层级的界面间导航

B) 应用程序外的导航

- 间接通知（通知栏内的消息、mail 内的日历等）
- 弹出通知

C) 应用程序间的导航

当上一个查看的界面是当前界面的父层级时,点击“返回”按键和点击“向上”按键的结果是一样的。然而,与“向上”按键不同的是, “向上”按键可以确保用户停留在应用程序中,而“返回”按键可以让用户回到系统首页,甚至会回到另一个应用户程序。

2. **图形测试**

A）横向比较，各控件操作方式统一。

B）自适应界面设计，内容根据窗口大小自适应。

C）页面标签风格是否统一。

D）页面是否美观。

E）页面的图片应有其实际意义而要求整体有序美观。

3. **内容测试**

A）输入框说明文字的内容与系统功能是否一致。

B）文字长度是否加以限制。

C）文字内容是否表意不明。

D）是否有错别字。

E）信息是否为中文显示

2.4.3性能测试

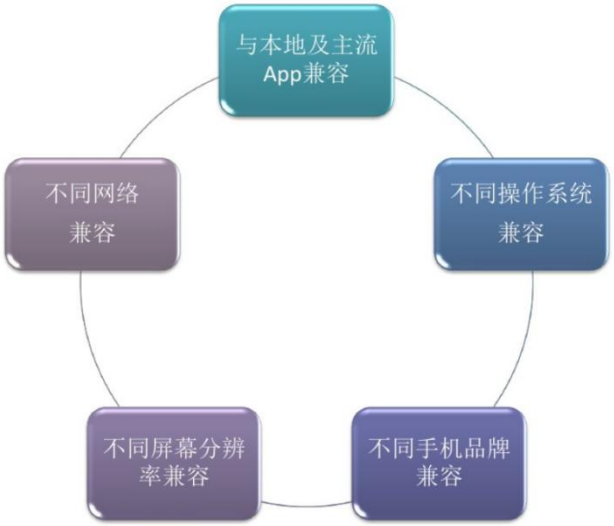
App 性能测试分客户端和服务端，为什么要这样分呢，其实我们要先了解 App 的架构，一般原生的 App，当用户请求 App 的某个页面时，App 会向服务器发送请求，服务端返回请求数据（一般是 json 数据），App 再将数据与模板合并渲染出页面，展示给用户。（这个

过程其实和浏览器本身差不多，只是我们在测试通过浏览器访问的应用性能时，并不关心浏览器本身的性能，因为这个由浏览器厂家完成了）。服务端的性能可以通过接口或者 web 网页模拟用户输入进行测试，和普通的 PC 端性能测试方法一样；客户端性能需要借助一些专门的工具来测试，App 性能的关注点主要有耗电量、流量占用、启动退出耗时、响应时延、流畅度、CPU 内存等。

- 响应能力测试：测试 App 中的各类操作是否满足用户响应时间要求。 App 安装、卸载的响应时间 App 各类功能性操作的响应时间
- 压力测试：反复/长期操作下，系统资源是否占用异常。App 反复进行安装卸载，检查系统资源是否正常，其他功能反复进行操作，检查系统资源是否正常
- app 占用的存储空间
- app 对于设备的电量消耗
- app 对于流量的消耗
- App 对于客户端 CPU 和内存的消耗
- API 测试

2.4.4兼容测试

主要测试内部和外部兼容性
1）与本地及主流 App 是否兼容
2） 与各种设备是否兼容，若有跨系统支持则需要检验在多个系统下，各种行为是否一致。 不同手机屏幕分标率的兼容性、不同手机品牌的兼容性



品牌/系列	主流机型	分辨率（像素）	系统版本（Android）

品牌/系列	主流机型	分辨率（像素）	系统版本（Android）
三星	Galaxy S25 / S25 Ultra	2340×1080（S25） 3120×1440（Ultra）	Android 15
小米	小米 15 Ultra / Redmi Note 15	2720×1220（Ultra） 2400×1080（Note）	Android 15（MIUI 17）
华为	Mate 70 Pro / P80	2720×1224（Mate） 2520×1080（P80）	鸿蒙 OS 5.0（兼容 Android）
OPPO	Find X8 Pro / Reno 13	3200×1440（Find） 2412×1080（Reno）	Android 15（ColorOS 16）
vivo	X100 Pro / iQOO 13	3040×1440（X100） 2400×1080（iQOO）	Android 15（OriginOS 5）
荣耀	Magic7 Pro / 荣耀 100	2848×1312（Magic） 2400×1080（荣耀 100）	Android 15（Magic UI 8.0）
苹果	iPhone 16 / 16 Plus	2556×1179（16） 2796×1290（Plus）	iOS 19
苹果	iPhone 16 Pro / Pro Max	2796×1290（Pro） 2992×1344（Pro Max）	iOS 19
苹果	iPhone 15 系列（存量）	2532×1170（15） 2796×1290（15 Pro Max）	iOS 18 / 19
苹果	iPhone SE 5	1792×828	iOS 19

测试覆盖：

优先级排序：

必测：Android 旗舰（小米 15 Ultra、三星 S25）+ 中端（Redmi Note 15）、iPhone 16 系列、iPad Pro

选测：折叠屏（华为 Mate X6）、入门机型（Tecno Spark 21）、小屏（iPhone SE 5）

核心关注点：

分辨率：优先覆盖 2400×1080（FHD+）和 2700×1200 以上（2K）

系统版本：Android 15/14、iOS 19/18（覆盖 90%以上用户）

特殊场景：折叠屏的状态切换适配、平板的横屏布局、小屏的 UI 压缩问题

覆盖主流分辨率：

主流机型：2556×1179 ~ 2992×1344（19.5:9 比例）

小屏机型：1792×828（16:9 比例，iPhone SE 系列）

IOS 操作系统：
主流版本：iOS 19（占比约 55%）
存量版本：iOS 18（占比约 35%），iOS 17（占比约 8%）
Android 系统：
主流版本：Android 15（占比约 65%）
存量版本：Android 14（占比约 25%），Android 13（占比约 8%）

2.4.5 安装更新卸载测试

验证 App 是否能正确安装、运行、卸载、以及操作过程和操作前后对系统资源的使用情况

- **安装**

A）软件安装后是否能够正常运行，安装后的文件夹以及文件是否写到了指定的目录里。
B）软件安装各个选项的组合是否符合概要设计说明。
C）软件安装向导的 UI 测试
D）安装后没有生成多余的目录结构和文件。

- **卸载**

A）测试系统直接卸载程序是否有提示信息。
B）测试卸载后文件是否全部删除所有的安装文件夹。
C）卸载是否支持取消功能，单击取消后软件卸载的情况。
D）系统直接卸载 UI 测试，是否有卸载状态进度条提示。

- **升级更新**

1）当客户端有新版本时，有更新提示。
2）当版本为非强制升级版时，用户可以取消更新，老版本能正常使用。用户在下次启动 App 时，仍出现更新提示。
3）当版本为强制升级版，而用户没有做更新时，退出客户端。下次启动 App 时，仍出现强制升级提示。
4）当客户端有新版本时，直接检查是否能正常更新，检查更新后的客户端功能是否是新版本。
5）更新完成之后，用户信息是否被清除了。
6）考虑网络的影响，3G/4G/WIFI 下是否都能正常升级或者能够基于流量的影响进行智能下载
7）考虑中断下载后是否可以继续或者重新下载
8）考虑断电和内存不足的问题，能够继续进行相关升级
9）考虑应用权限问题，如果新版本对于应用权限有了扩展，需要进行权限确认
10）考虑不同机型，升级测试需要对各种机型进行覆盖测试
11）选择升级情况下旧版本的兼容性

如果不是强制升级，那新旧版本的 app 同时运行时必不可少的，此时需要考虑新旧版本并行时后台接口的兼容性。在进行旧版本功能兼容性验证时，可以进行主要流程的测试和变更的接口影响到的功能详细验证，这样可以缩小测试范围，减少测试时间同时又能保证相应的变更都进行了测试。

12）覆盖升级后新版本的使用情况

a. 除了新版本自身的新功能验证之外，要进行主要业务流程的验证。

b. 在覆盖升级前，需要模拟使用旧版本的用户进行缓存数据的创建，然后进行升级，确认缓存数据升级后可以正常显示，相关功能工作正常。

2.4.6交叉事件测试

交叉测试又叫事件或冲突测试，是指一个功能正在执行过程中，同时另外一个事件或操作对该过程进行干扰的测试。如：App 在前/后台运行状态时与来电、文件下载、音乐收听等关键运用的交互情况测试等。

交叉事件测试非常重要，能发现很多应用中潜在的性能问题。

1）多个 App 同时运行是否影响正常功能。

2）App 运行时前/后台切换是否影响正常功能。

3）App 运行时拨打/接听电话。

4）App 运行时发送/接收信息。

5）App 运行时发送/收取邮件。

6）App 运行时浏览网络。

7）App 运行时使用蓝牙传送/接收数据。

8）App 运行时使用相机、计算器等手机自带设备。

2.4.7推送测试

app 消息推送是指 app 开发者通过第三方工具将自己想要推的消息推送给用户，让用户被动的接收。进行推送测试主要进行以下几个方面的测试：

1) 检查 Push 消息是否按照指定的业务规则发送。
2) 当用户设置不接收推送消息时，不会再接收到 Push 消息。
3) 如果用户设置了免打扰的时间段，检查在免打扰时间段内，用户接收不到 Push。在非免打扰时间段内，用户能正常收到 Push。
4) 当 Push 消息是针对登录用户的时候，需要检查收到的 Push 与用户身份是否相符，没有错误的将其他人的消息推送过来。一般情况下，只对手机上最后一个登录用户进行消息推送
5) 测试 Push 时，需要采用真机进行测试。
6) 推送的消息包括两大类：运营人员手动编辑、推送的公告、活动等，与用户行为（比如交易）相关的通知，这部分的消息是在代码执行过程中自动生成、推送。

确保成功集成和调用第三方 app

测试 app 分享功能

测试 app 显示外部链接的功能

测试 app 使用社交媒体等账号登录的功能

测试 app 推送服务

测试 app 和输入法等 app 交互的功能

2.4.8安全测试

1.数据安全性

1) 当将密码或其它的敏感数据输入到应用程序时，其不会被存储在设备中，同时密码也不会被解码。
2) 输入的密码将不以明文形式进行显示。
3) 密码、信用卡明细或其他的敏感数据将不被存储在它们预输入的位置上。
4) 当应用程序处理信用卡明细或其它的敏感数据时，不以明文形式将数据写到其他单独的文件或者临时文件中。以防止应用程序异常终止而又没有删除它的临时文件，文件可能遭受入侵者的袭击，然后读取这些数据信息。
5) 应用程序应考虑用户提示信息或安全警告
6) 在数据删除之前，应用程序应当通知用户或者提供一个“取消”命令的操作。
7) 应用程序应当能够处理当不允许应用软件连接到个人信息管理的情况。
8) 当进行读或写用户信息操作时，应用程序将会向用户发送一个操作提示信息。
9) 在没有用户明确许可的前提下不损坏删除个人信息管理应用程序中的任何内容。
10) 如果数据库中重要的数据正要被重写，应及时告知用户。
11) 能合理的处理出现的错误。
12) 意外情况下应提示用户。

2.通讯安全性

1) 在运行软件过程中，如果有来电、短信、蓝牙等通讯或充电时，是否能暂停程序，

优先处理通信，并在处理完毕后能正常恢复软件，继续其原来的功能。

2) 当创立连接时，应用程序能够处理因为网络连接中断，进而告诉用户连接中断的情况。

3) 应能处理通讯延时或中断。

4) 应用程序关闭网络连接不再使用时应及时关闭，断开。

3.人机接口安全测试

1) 返回菜单应总保持可用。

2) 应用程序必需利用目标设备适用的全屏尺寸来显示内容。

3) 声音的设置不影响使用程序的功能。

4) 应用程序必须能够处理不可预知的用户操作，例如错误的操作和同时按下多个键

2.4.9硬件环境测试

主要涉及的是与硬件相关的测试，包括手势操作测试和网络环境测试

- **手势操作测试**

1）手机开锁屏对运行中的 App 的影响

2）切换网络对运行中的 App 的影响

3）App 运行时关机

4）App 运行时重启系统

5）App 运行时充电

6）App 运行时 kill 掉进程再打开

- **网络环境测试（弱网环境测试）**

手机的网络目前主要分为 2G、3G、4G、WIFI。目前 2G 的网络相对于比较慢,基本可以不考虑。

1) 无网络时，执行需要网络的操作，给予友好提示，确保程序不出现 crash。

2) 内网测试时，要注意选择到外网操作时的异常情况处理。

3) 在网络信号不好时，检查功能状态是否正常，确保不因提交数据失败而造成 crash。

4) 在网络信号不好时，检查数据是否会一直处于提交中的状态，有无超时限制。如遇数据交换失败时要给予提示。

5)在网络信号不好时，执行操作后，在回调没有完成的情况下，退出本页面或者执行其他操作的情况，有无异常情况，此问题也会经常出现程序 crash。

服务器宕机或出现 404、502 等情况下的测试

后台服务牵涉到 DNS、空间服务商的情况下会影响其稳定性

如：当出现域名解析故障时，你对后台 API 的请求很可能就会出现 404 错误，抛出异常。这时需要对异常进行正确的处理，否则可能会导致程序不能正常工作

2.4.10 客户端数据库测试

1）一般的增、删、改、查测试。
2）当表不存在时是否能自动创建，当数据库表被删除后能否再自建，数据是否还能自动从服务器中获取回来并保存。
3）在业务需要从服务器端取回数据保存到客户端的时候，客户端能否将数据保存到本地。
4）当业务需要从客户端取数据时，检查客户端数据存在时，App 数据是否能自动从客户端数据中取出，还是仍然会从服务器端获取？检查客户端数据不存在时，App 数据能否自动从服务器端获取到并保存到服务器端。
5）当业务对数据进行了修改、删除后，客户端和服务器端是否会有相应的更新。

2.4.11 用户体验测试

以主观的普通消费者的角度去感知产品或服务的舒适、有用、易用、友好亲切程度。通过不同个体、独立空间和非经验的统计复用方式去有效评价产品的体验特性，提出修改意见提升产品的潜在客户满意度。

1）是否有空数据界面设计，引导用户去执行操作。
2）是否滥用用户引导。
3）是否有不可点击的效果，如：你的按钮此时处于不可用状态，那么一定要灰掉，或者拿掉按钮，否则会给用户误导。
4）菜单层次是否太深。
5）交互流程分支是否太多。
6）相关的选项是否离的很远。
7）一次是否载入太多的数据。
8）界面中按钮可点击范围是否适中。
9）标签页是否跟内容没有从属关系，当切换标签的时候，内容跟着切换。
10）操作应该有主次从属关系。
11）是否定义 Back 的逻辑。涉及软硬件交互时，Back 键应具体定义。
12）是否有横屏模式的设计，应用　一般需要支持横屏模式，即自适应设计。

2.5 Android 测试和 IOS 测试的区别

1. **手机操作系统：**Android 较多，ios 较少且不能降级，只能单向升级；新的 ios 系统中的资源库不能完全兼容低版本中的 ios 系统中的应用，低版本 ios 系统中的应用调用了新的资源库，会直接导致闪退（Crash）；
2. **操作习惯：**Android，Back 键是否被重写，测试点击 Back 键后的反馈是否正确；应用数据从内存移动到 SD 卡后能否正常运行等；
3. **push 测试：**Android 点击 home 键，程序后台运行时，此时接收到 push，点击后唤醒应用，此时是否可以正确跳转；ios 点击 home 键关闭程序和屏幕锁屏的情况；
4. **安装卸载测试：**Android 的下载和安装的平台和工具和渠道比较多，ios 主要有 app

store，iTunes 和 testflight 下载；

5. 升级测试：可以被升级的必要条件：新旧版本具有相同的签名；新旧版本具有相同的包名；有一个标示符区分新旧版本（如版本号），对于 Android 若有内置的应用需检查升级之后内置文件是否匹配（如内置的输入法）

2.6 App 测试与 Web 测试的区别

从功能测试方面讲，Web 测试与 App 测试在测试用例设计和测试流程上没什么区别。而两者的主要区别体现在如下几个方面：

1. **系统结构方面**
   - Web 项目，B/S 架构，基于浏览器的；Web 测试过程中，客户端会随服务器端同步更新，所以只需更新服务器端即可
   - App 项目，C/S 架构，基于客户端的；App 测试过程中，只要修改了服务端，那么客户端用户所有核心版本都需要进行回归测试一遍
2. **性能方面**
   - Web 项目，需要监测响应时间、CPU、Memory，另外则还需系统能支持多少用户同时在线；超过最大用户数，系统会给出什么样的反映
   - App 项目，需要监测响应时间、CPU、Memory，另外则还需监测流量、电量等
3. **兼容方面**
   - Web 项目：首先，考虑操作系统兼容（Windows7、Windows10、Linux 等）；其次，考虑浏览器兼容（IE8、IE10、Firefox、Chrome、360 等）
   - App 项目：首先，考虑设备系统兼容（Android【华为、联想、小米、三星等】、 iOS【ipad、iphone】、Windows【Win7、Win8】、OSX【Mac】）；其次，考虑手机设备的大小、型号、分辨率的兼容。
4. **测试工具方面**
   - Web 测试，自动化工具通常使用 Selenium，性能测试工具通常使用 LoadRunner / JMeter
   - App 测试，自动化工具通常使用 Appium / Monkey，性能测试工具通常使用 JMeter
5. **专项测试方面**

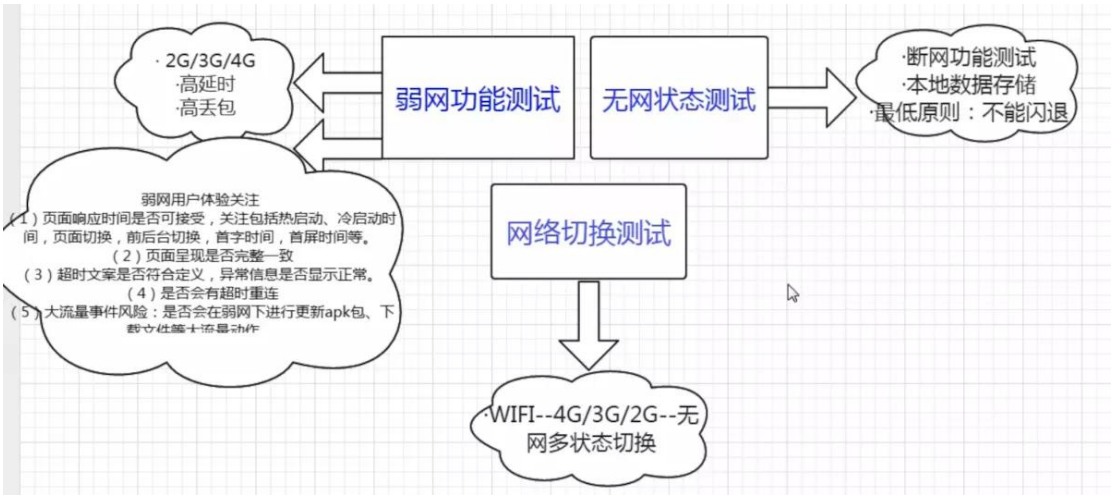
- Web 专项测试
  - 链接测试：主要是保证链接的可用性和正确性，考虑链接的页面是否存在？是否按指示链接到了相应链接的页面？是否存在空白页面等
  - 图形测试：首先，确保图形有明确用途，图片或动画不要过于紧凑，以免浪费传输时间；其次，验证文字回绕是否正确
  - 打印测试：考虑网页打印是否正常（显示的图片和文本的对齐方式可能与打印出来的东西不一样，是否完整打印）
- App 专项测试

- 安装、更新、卸载：
  安装：需考虑安装时的中断、弱网以及安装后删除安装文件等情况
  更新：分强制更新、非强制更新、增量包更新、断点续传、弱网状态下更新等几种情况
  卸载：需考虑卸载后 App 相关文件是否删除干净
- 权限测试：设置某个 App 是否可以获取该权限，比如是否可访问短信、读取联系人、相册、照相机、位置信息等
- 安全测试：安装包是否可反编译代码、安装包是否签名、权限设置
- 边界测试：可用存储空间少、飞行模式、系统时间有误、第三方登录（QQ、微信、微博登录）以及没有 SD 卡/双 SD 卡等
- 界面操作：关于手机端测试，需注意横竖屏切换、多点触控、手势、事件触发区、前后台的切换（从后台回到 App，检查是否停留在上次操作界面、功能和应用状态是否一样）等
- 干扰测试：电话响应（接通、呼叫挂断、呼叫保持）、收发短信、中断（插拔数据线、手机锁屏、闹钟、蓝牙等）、电量不足、关机、重启、死机等
- 网络测试：首先，弱网络测试（模拟 2G、3G、4G、WiFi 网络状态以及丢包情况，重点要考虑回退和刷新是否会造成二次提交）；其次，网络切换测试（网络断开后重连、3G 切换到 4G/WiFi 等）

3 APP 弱网测试

3.1 什么是弱网测试

弱网测试知识网络测试的一部分，整个 app 的网络测试包含了弱网测试，网状态测试，以及网络切换测试。



弱网测试主要是对 2G/3G/4G 以及高延迟高丢包的这几种状态做模拟然后看下 app 的容错性如何。那么弱网测试的主要关注点在哪里呢？

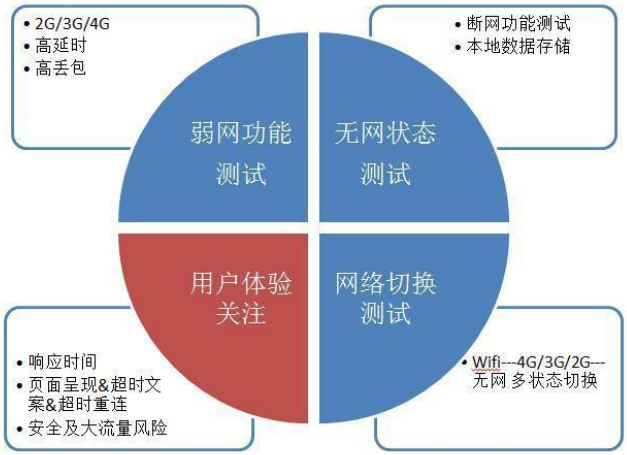
（1）页面相应时间是否可接受，关注包括热启动，冷启动时间，页面切换，前后台切换，首字时间，首屏时间等。

（2）页面成仙是否完整一致

（3）超时文案是否符合定义，一场信息是否显示正常。
（4）是否会有超时重连
（5）大流量时间风险：是否会在弱网下进行更新 Apk 包，下载文件等大流量动作

3.2 弱网测试策略

弱网测试作为健壮性测试的重要部分，对于 APP 来说必不可少。目前国内所处的网络环境并非全是流畅的 WIFI 环境，仍然有相当的用户使用 4G、3G、2G 等网络。另外因为移动端产品的使用场景多变，在地铁里、公交上，甚至是电梯、车库等场景使用 APP，这些场景因为环境原因，网络覆盖度较低，网络环境差，也就是我们常说的弱网环境。因此我们需要针对这些场景的弱网环境下，验证出现丢包、网络延时的情况下，APP 的展示及容错机制，避免因用户体验不友好造成用户的流失。



如上图所示，弱网测试主要进行特殊网络状态下的功能测试同时关注用户体验，具体来说，弱网测试包括弱网功能测试、无网状态测试、网络切换测试等，测试的同时关注用户体验的诸多方面。

1.弱网功能测试

这一部分主要是在各种非 WIFI 网络环境下进行的功能测试，同时模拟高延时和高丢包的异常网络环境进行健壮性测试。2G/3G/4G 的网络可以通过使用电话卡移动 / 联通 / 电信等网络进行模拟，关注页面的响应时间、页面呈现是否完整一致等。

弱网功能测试建议将整体的功能测试用例在弱网环境下进行一轮测试，相同模块下的功能可以分多个网络条件进行测试。这部分发现的问题可能会有：页面图片在弱网环境下加载不出来（图片加载逻辑需优化）、需要模版的页面版式结构混乱（模版文件在弱网环境的加载需优化）、页面响应时间较长没有任何显示（页面显示逻辑待优化、重试机制加入）等。

常见场景举例：

1） 结合 APP 本身的业务属性

比如社交类 APP（聊天、抢红包）对网络环境依赖性大且用户关注度高，弱网环境下需要重点关注。

比如互联网金融 APP，申购流程中创建订单后是否支付成功，用户关注度最高（涉及扣费）。例如 弱网环境，创建订单失败，用户关注是否被扣费；创建订单成功后支付失败，再次支付是否重复扣费等。

2) 使用频率和易遇到弱网的场景

比如视频 APP 观看小视频，用户在碎片时间极易观看小视频（APP 用户喜欢使用碎片化时间进行娱乐操作）；还有就是刷微博，看新闻等。

比如金融 APP，用户在碎片化时间使用金融 APP，领取奖品、查看理财类新闻、查看收益。

因弱网引起的一些常见问题及原因分析：

1) 连续发送请求的场景

问题：搜索时输入关键字会连续发请求，停下时，显示最终的关键字搜索结果，但很快又会被前面的关键字搜索结果覆盖了；

原因：中间的请求返回较慢，显示了最终的结果后，之前的请求返回的数据应不做处理。这种情况对应支付、下单环节等也要注意，不要因为网络问题造成请求堆积，当网络恢复时产生多个订单或者多次扣费。

2) 弱网环境操作可用控件

问题：因弱网导致连接时间过长，期间界面上有些控件可以操作，但是这些操作依赖服务器返回的数据，那么操作时会因请求未成功而出现异常；

原因：弱网络环境下，请求的数据返回时间较长，等待的过程中，如果页面上的相关控件仍然可以操作，则容易出现异常（闪退现象、报数据错误等）。

3) 弱网环境登录

问题：在弱网络环境下容易出现登录不上或者登录后立即掉线；

原因：登录没有缓冲机制，而请求超时时间的设置没有区分同网络情况，比如 WIFI 因为网速快，临时出现无法连接，再次重连时重试时间应该较短，而比如 2G/3G/4G 网络，可能因运营商的原因网络会长时间保持弱网状态，那么就需要相比 WIFI 要有更长的重试和超时机制。需要根据不同的网络环境设置重试机制与超时机制。

4) 上传大图或者多图

问题：上传大图或者多图时，在弱网络环境下出现进度条走到一半卡住然后又从头开始；

原因：采用分段上传方式，直至请求超时，分段传输没有结束，代码逻辑不对，导致每次重试都重头上传，一直循环。

2.无网状态测试

无网状态测试则是在切段网络的情况下进行的测试，主要关注页面的显示与交互、本地数据的存储、断网功能的使用等，经常该部分也需要与网络切换部分协同进行。

通常来说，包含如下情况：

1) 断网情况下请求一个非本地数据的页面需要设定一定的时间等待上限，及时提示网络异常以及提示重试；
2) 断网情况下请求一个部分本地数据的页面需要观察本地数据的部分是否加载显示正常，待请求的部分是否符合交互给的缺省样式一致；
3) 断网情况下请求一个完全本地数据的页面是否显示正常。这里还需考虑本地数据存储的情况，有些需要联网后上报服务器的数据本地是否正确存储，联网后这些数据能否正常上报。

无网状态测试建议按照页面划分进行，针对每个页面单独测试无网状态的显示，页面间跳转的显示，页面内功能的点击和显示，同时关注无网到有网时的页面恢复显示状态、数据上报情况是否正常。

3.网络切换测试

这部分主要是进行几个不同网络场景的切换，包括 wifi-3G/4G/5G、wifi-无网、3G/4G/5G-wifi、3G/4G/5G-无网、无网-3G/4G/5G、无网-wifi 等。主要关注页面的显示与交互，尤其是弱网到 wifi，wifi 到弱网的情况，是否会有页面的 crash 以及显示的错乱、session 是否一致、请求堆积处理等。

4.用户体验关注

弱网测试的目的就是尽可能保证用户体验，关注的关键点包括：

1) 页面响应时间是否可接受，关注包括热启动、冷启动时间，页面切换，前后台切换，首屏时间等；
2) 页面呈现是否完整一致；
3) 超时文案是否符合定义，异常信息是否显示正常，不要出现错误码、什么 API 请求异常之类的术语，普通用户是看不懂的带来不好的体验；
4) 是否会有超时重连（不同模块，开发对请求处理不同。测试前可了解，代码是否支持自动重复请求，自动重发请求的频率是什么？）
5) 不会因网络连接不正常而导致客户端 ANR（ANR 即 Application Not Responding,应用程序无响应） 或者 crash；
6) 有超时限制，不能无限重连，比如有些图书 app，有很多本地资源可用，在弱网情况下，用户想使用本地资源，但是一直处于重连状态，导致无法操作，用户体验很差；
7) 在网络连接过程中，网络连接恢复 ，请求正常提交。客户端正常反应；
8) 超时时间不宜过长，及时让用户知晓网络情况，用户对于等待时间的容忍度非常低；
9) 安全角度：是否会发生 DNS 劫持、登录 IP 更换频繁、单点登录异常等；
10) 大流量事件风险：是否会在弱网下进行更新 apk 包、下载文件等大流量动作。

3.3 模拟弱网环境测试的原理

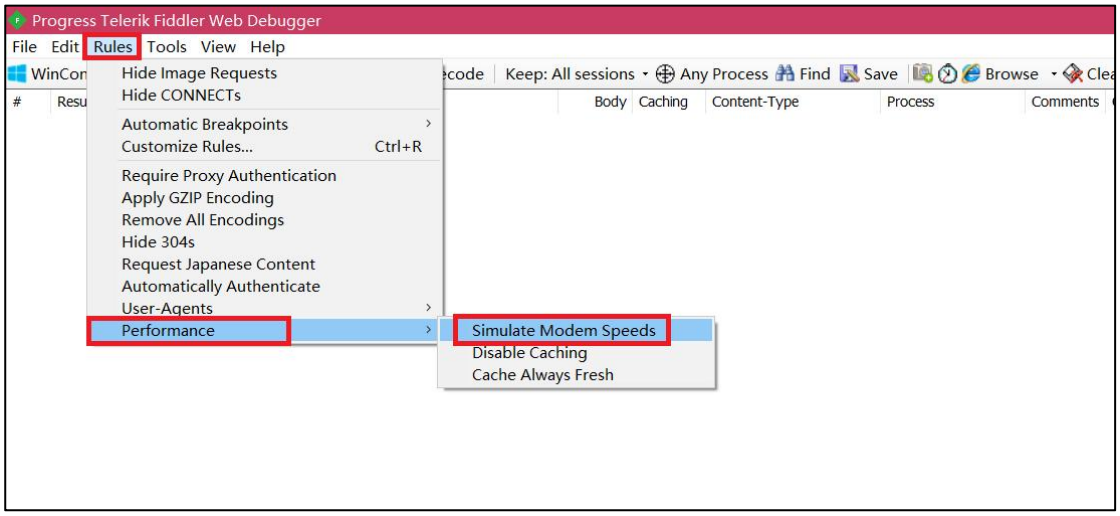
我们可以通过抓包工具，比如 Fiddler、Charles 来模拟限速，因为 Fiddler 本来就是个代理，它提供了客户端请求前和服务器响应前的回调接口，我们可以在这些接口里面自定义一些逻辑。Fiddler 的模拟限速正是在客户端请求前来自定义限速的逻辑，此逻辑是通过延迟发送数据或接收的数据的时间来限制网络的下载速度和上传速度，从而达到限速的效果。

3.4 Fiddler 模拟弱网环境测试

Fiddler 提供了一个功能，让我们模拟低速网路环境，启用方法如下：

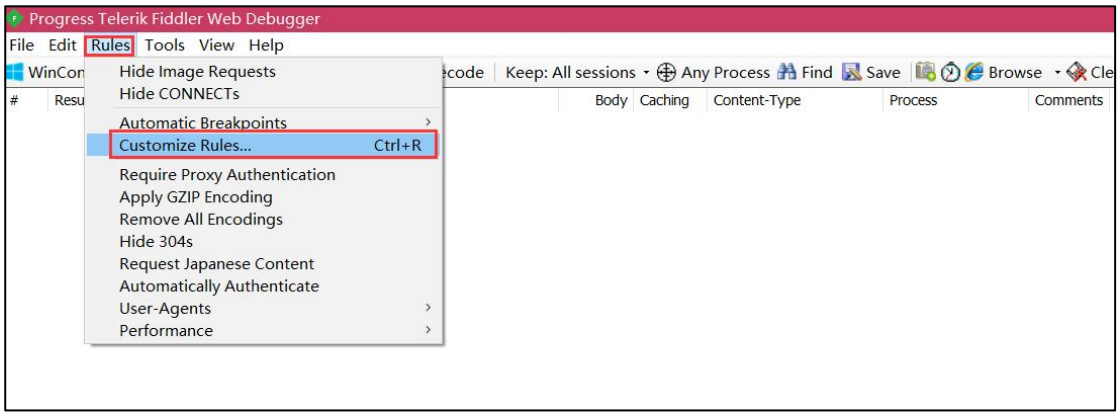
第一步：启用模拟调制解调器速度选项

点击 Rules > Performances > Simulate Modem Speeds

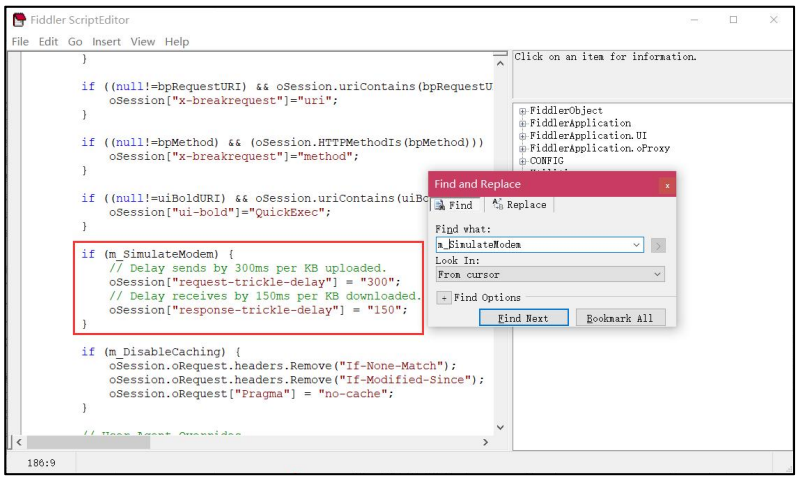


第二步：设置速度模拟配置

点击 Rules > Customize Rules 或者使用快捷键 Ctrl+R 打开用户配置文件；



使用快捷键 Ctrl+F，找到配置文件中的 m_SimulateModem；



代码如下：

```
if (m_SimulateModem) {
  // Delay sends by 300ms per KB uploaded.每延迟300ms发送1kb的数据，也就是每1s发送3kb的数据
  oSession["request-trickle-delay"] = "300";
  // Delay receives by 150ms per KB downloaded.每延迟150ms下行1kb的数据
  oSession["response-trickle-delay"] = "150";
}
```

通过设置上行和下行速率，模拟不同网络速度。

注意：如果一旦发生修改，那么一定要重新勾选第一步。

网络设置参考：

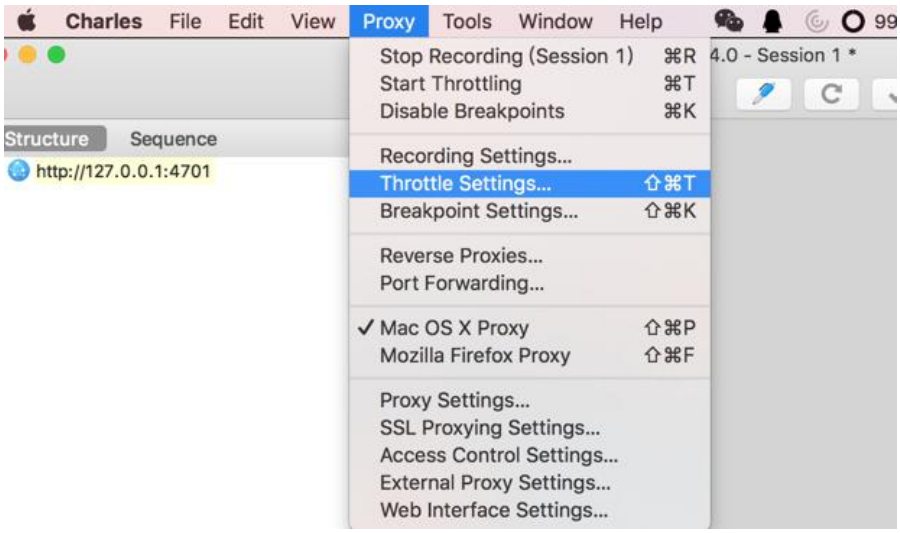
网络环境	上/下行带宽(kbps)	上/下行丢包率(%)	上/下行延迟(ms)	DNS延迟(ms)	备注
2G	20/50	0/0	500/400	0	
3G	330/2000	0/0	100/100	0	
4G	40000/80000	0/0	15/10	0	
wifi	33000/40000	0/0	1/1	0	
带宽有限环境	32/32	0/0	200/100	0	
低丢包率、低时延的环境（上行）	33000/40000	10/0	100/100	200	
低丢包率、高时延的环境（上行）	33000/40000	10/0	350/350	350	
低丢包率、低时延的环境（下行）	33000/40000	0/10	100/100	200	
低丢包率、高时延的环境（下行）	33000/40000	0/10	350/350	350	
低丢包率、低时延的环境	33000/40000	10/10	100/100	200	
低丢包率、高时延的环境	33000/40000	10/10	350/350	350	
高丢包率的环境（上行）	33000/40000	90/0	100/100	200	
高丢包率的环境（下行）	33000/40000	0/90	100/100	200	
高丢包率的环境	33000/40000	90/90	100/100	200	
网络超时（响应）	33000/40000	0/100	100/100	200	
网络超时（请求）	33000/40000	100/0	100/100	200	
网络超时（完全丢包）	33000/40000	100/100	100/100	200	
无网（飞行模式或关闭网络）					

wifi环境下即可设置测试

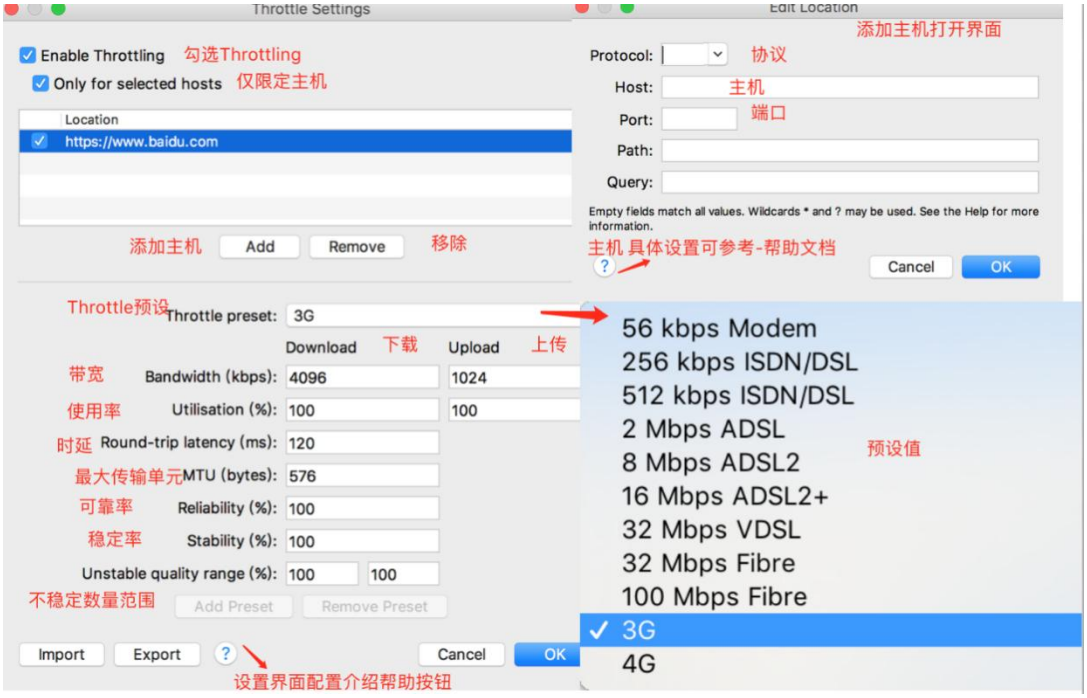
3.5 Charles 模拟弱网环境测试

以 charles 4.0 版本为例

打开 Proxy > Throttle Settings



打开 Throttle Settings 界面如下



Charles 的预设已经有常用的网速模拟设置，比如 56kbps Modem、3G、4G 等，根据需要选择即可，如果上述预设都不能满足，则可以根据需要自己修改预设值。

4 App Monkey 测试

4.1 Android adb 常用命令

- ✧ adb 连接
  命令：adb connect 设备 IP 或名称：端口号
- ✧ adb 连接成功验证
  检查设备命令：adb devices
  检查 adb 版本命令：adb version
  进入设备命令：adb shell
- ✧ 使用 adb 安装、卸载应用
  安装命令：adb install apk 包名.apk
  卸载命令：adb uninstall apk 包名.apk

4.2 Android 测试环境搭建

4.2.1 安装 Java JDK 环境

1、首先要安装 java 的 JDK；
2、安装好 JDK 之后需要配置环境变量，在“系统变量”中设置 3 项属性，JAVA_HOME,PATH,CLASSPATH(大小写无所谓),若环境变量中已存在这个变更则点击“编辑”，不存在则点击“新建”；

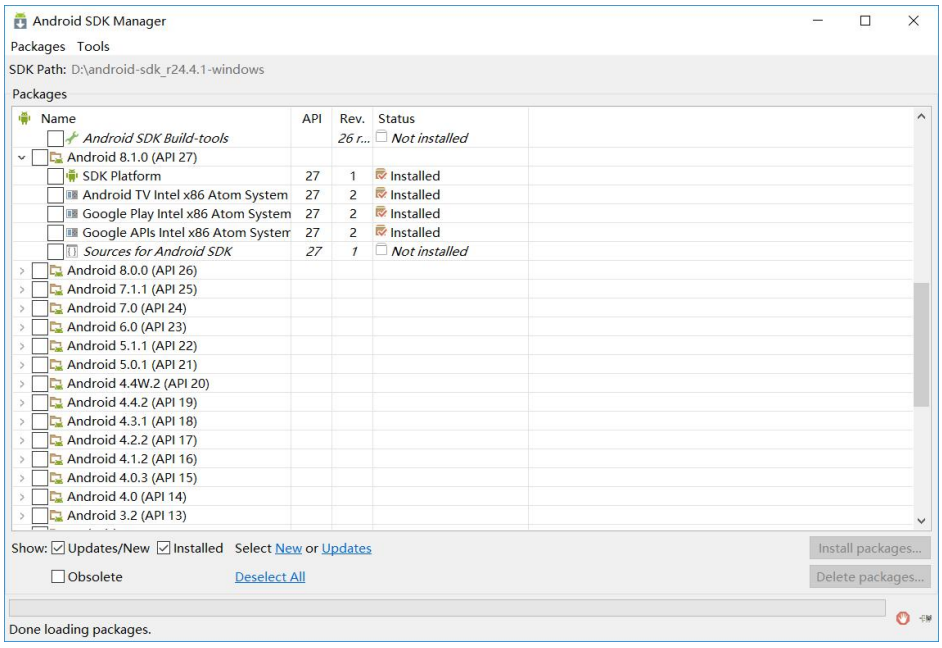
- JAVA_HOME 指明 JDK 安装路径

- PATH 使得系统可以在任何路径下识别 java 命令，设为：
  %JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
- CLASSPATH 为 java 加载类(class or lib)路径，只有类在 CLASSPATH 中，java 命令才能识别，设为：.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar (要加.表示当前路径)
  %JAVA_HOME%就是引用前面指定的 JAVA_HOME；

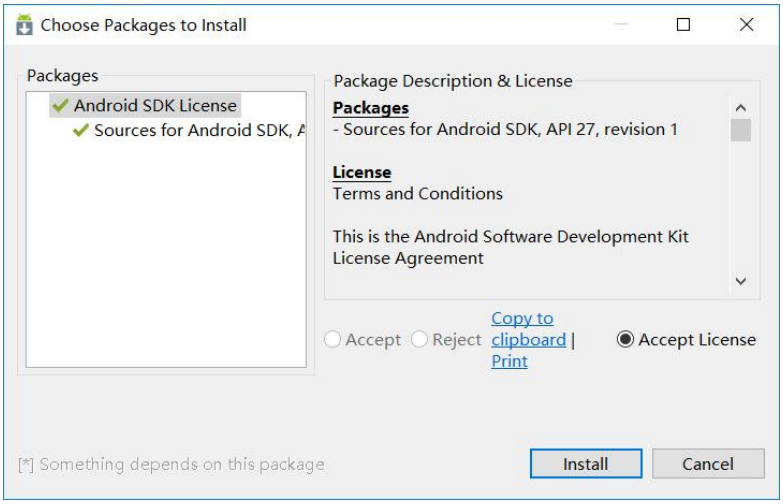
3、“开始”—>;“运行”，键入“cmd”；输入命令“java -version”，“java”，“javac”几个命令，出现画面，说明环境变量配置成功；

4.2.2 安装 Android SDK 环境

1、下载最新的 Android SDK ；
2、解压 Android SDK 文件，里面有两个应用程序：“SDK Manager.exe”（负责下载或更新 SDK 包） 和 ″AVD Manager.exe″（负责创建管理虚拟机）。先运行“SDK Manager.exe”进行 SDK 下载。
3、运行后出现下面的界面，选择自己想安装的 Android 版本，然后点击“Install X packages”安装。

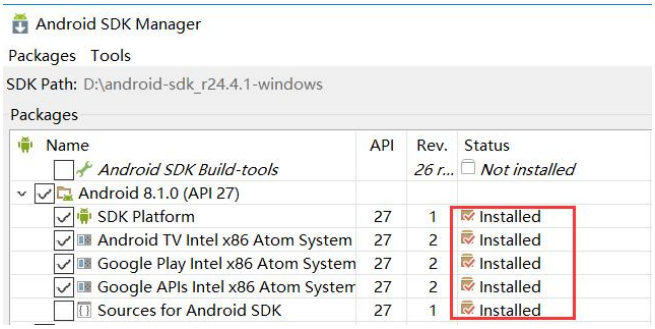


4、在新出现的界面上，选择接受并遵守所有许可内容(Accept license)，再点击“Install”。



然后 Android SDK 管理器就开始下载并安装你所选的包了，等上一段时间就好了。

5、安装好后，在 Android SDK 管理器界面上，你所选的包后面会显示“Installed”，表示已经安装好了，如图。



4.2.3 安装 Android 模拟器

App 测试还需要移动端设备，可以使用真机进行测试或模拟器进行测试，模拟器和真机有巨大差异，建议使用真机做测试，如果没有真机，可以下载安装虚拟模拟器。

- **常用模拟器有:**

  AVD Manager.exe、Genymotion、夜神模拟器、逍遥安卓模拟器、腾讯手游助手、iphone 模拟器电脑版、雷电模拟器、iphone 手游模拟器电脑版、ios 模拟器、BlueStacks 模拟器、天天模拟器

- **夜神模拟器【推荐】**

1) 下载并安装夜神模拟器;
2) 启动打开夜神模拟器 ;
3) 打开 CMD 命令窗口进入到模拟器的安装路径,默认安装位置:C:\Program Files\Nox\bin
4) 输入 nox_adb.exe connect 127.0.0.1:62001 或者 adb connect 127.0.0.1:62001（夜神模拟器的默认端口号为 62001，或者 52001）即可以连接到模拟器 ;
5) 打开夜神模拟器，选择一个需要调试的应用

6) cmd 命令进入到 DOS 窗口 输入 adb devices，可以显示到连接上的设备

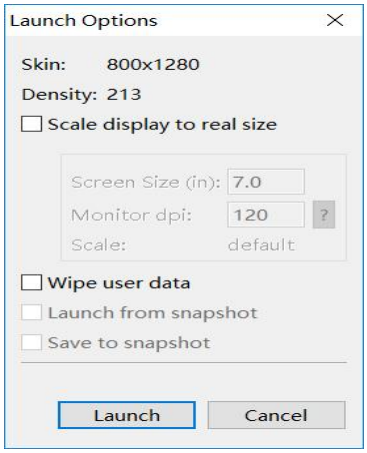
➢ AndroidSDK 自带的 AVD 创建模拟器

1、在 Android SDK 安装文件夹下，运行 AVD Manager.exe。

2、打开 AVD Manager.exe 后，点击“New”创建新的模拟器；

3、创建一个新的 Android Virtual Device (AVD)：输入 AVD 名称、选择 Target、输入模拟的 SD Card 的容量大小、以及选择外观皮肤 Skin。然后 “Create AVD”。

4、开始运行你新建的模拟器 Virtual Device：选中这个虚拟设备并点击“Start”，在出现的界面上直接点击“Launch”就可以启动 Android 模拟器了。



注意：Android SDK 自带的 AVD 实在不争气，出现各种问题：

- AVD 模拟器奇卡无比；
- 使用 USB 数据线链接手机经常无法识别设备；
- Log 日志输出不全；

➢ 安装 Genymotion 模拟器

1、下载并安装最新版的 Genymotion 模拟器；

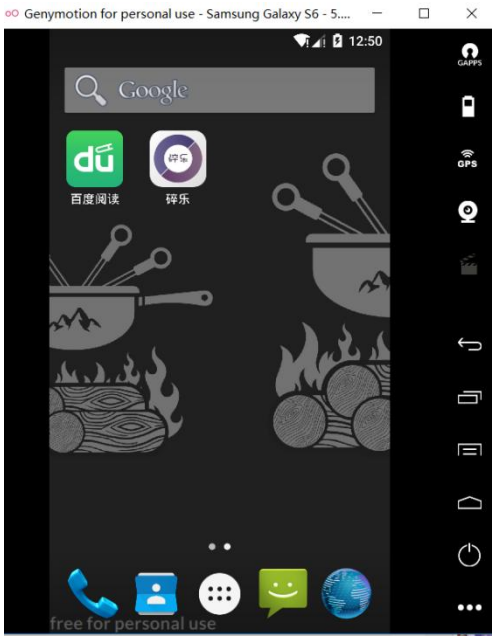
Genymotion 官网 http://www.genymotion.net/ ；

2、Genymotion 中安装 SDK6.0 和 7.0 的设备(Genymotion 支持 SDK 6.0 7.0)；

3、启动对应的模拟器,cmd 窗口中输入 adb devices；

4、可直接输入 adb shell 进入到模拟器；

1. 安装应用，如:碎乐.app(直接拖动 apk 到模拟器即可)
2. 应用安装成功，如下图：



注意：Genymotion 安装 apk 无法安装：

1、下载 ARM_Translation_Marshmallow6.0.zip 文件，存放到本地的盘符根目录下：如 D 盘根目录：D:\ARM_Translation_Marshmallow6.0.zip；

2、将下载的 ARM_Translation_Marshmallow6.0.zip 直接拖拽到模拟器中安装；

3、安装完成后重启模拟器即可；

4.2.4 启动 Monkey 测试环境

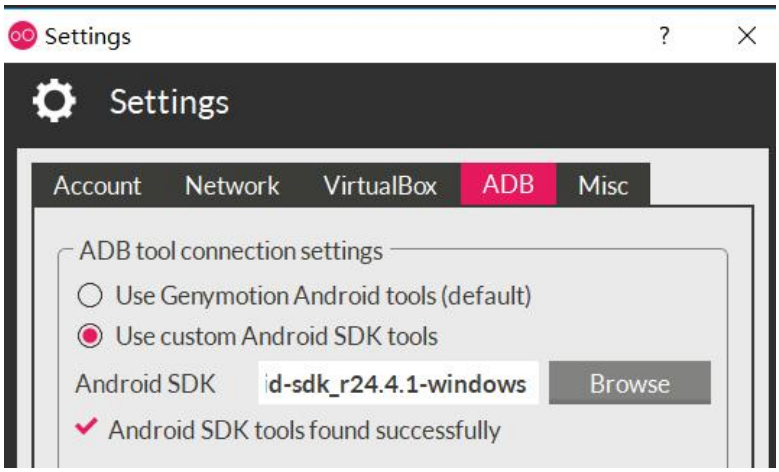
1.启动模拟器后，再启动命令行，进入 Dos 模式；

2.进入 SDK 安装包下的 adb.exe 文件所在的路径，输入”adb shell”进入手机（模拟器），然后输入“adb devices”，显示出连接的手机（模拟器）型号。

```
D:\android-sdk_r24.4.1-windows\adb>adb shell
root@vbox86p:/ # adb devices
List of devices attached
emulator-5554   device
```

1. 此时就可以执行 Monkey Test 命令了

注：如果启动失败，请检查模拟器的设置，重新加载 sdk



4.3 Monkey 测试概述

Monkey 在英文里的含义是“猴子”，在测试行业对应有一个术语叫“猴子测试”，那么什么是“猴子测试”？

“猴子”测试是指没有测试经验的人甚至是对计算机不了解的人（就像一只猴子一样）不需要知道程序的任何用户交互方面的知识，如果给他一个程序，他就会针对他看到的任何界面进行操作，当然其操作也是毫无目的的，乱按乱点，这种测试往往在产品周期的早期阶段会找到很多很好的缺陷，为用户节省不少时间。

4.3.1 Monkey 简介

Monkey 是 Android 系统 SDK 中附带的一个命令行工具，可以运行在模拟器里或实际设备中。Monkey 可以向被测应用程序发送伪随机的事件流（如按键、触屏、手势等），实现对应用程序进行压力测试目的。Monkey 测试是 android 自动化测试的一种手段，它非常简单易用，在模拟器或设备上运行的时候，如果用户触发了点击、触摸、手势等操作，它就会产生随机脉冲信号。因此我们可以通过 Monkey 用随机重复的方法来对应用程序进行一些稳定性、健壮性方面的测试。Monkey 测试是一种测试软件稳定性、健壮性的快速有效的方法。

该工具用于进行**压力测试，**然后开发人员结合 monkey 打印的日志和系统打印的日志，分析测试中的问题

4.3.2 Monkey 测试的特点

Monkey 测试,所有的事件都是随机产生的，不带任何人的主观性。

1、测试的对象仅为应用程序包，有一定的局限性。

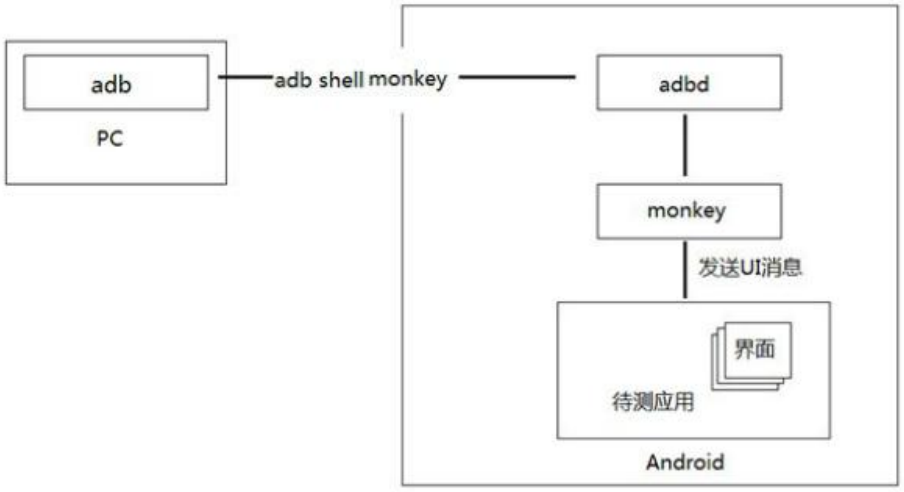
2、Monkey 测试使用的事件数据流是随机的，不能进行自定义。

3、可对 Monkey 测试的对象，事件数量，类型，频率等进行设置。

4.3.3Monkey 测试原理

利用 socket 通讯（Android 客户端和服务器以 TCP/UDP 方式）的方式来模拟用户的按键输入、触摸屏输入、手势输入等。 Monkey 的测试对象仅为应用程序包，有一定的局限性。测试使用的事件流数据流是随机的，不能进行自定义。 可对 MonkeyTest 的对象、事件数量、类型、频率等进行设置

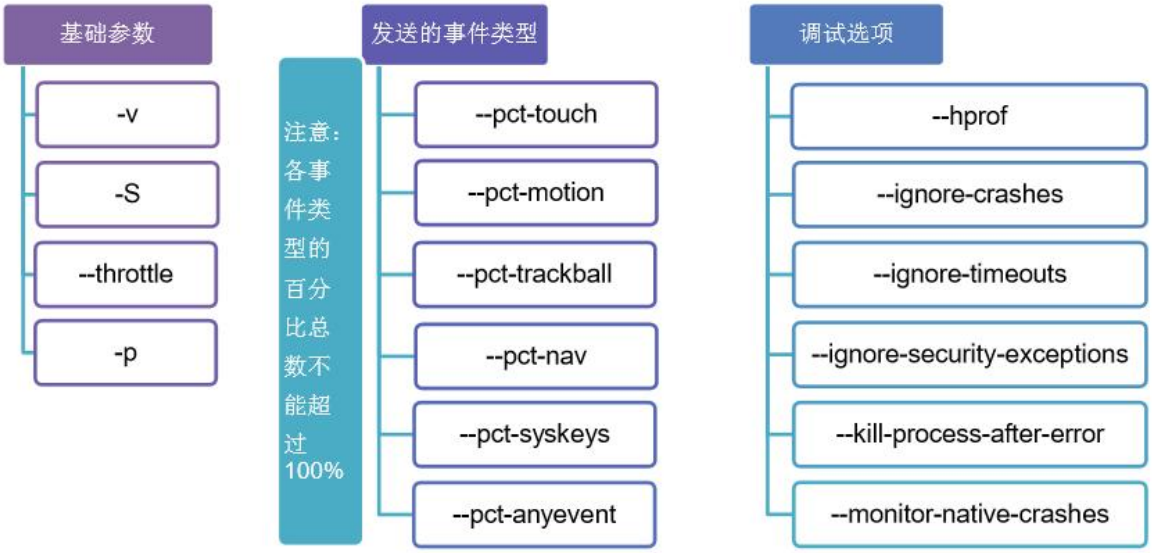
Monkey 用 Java 语言写成，Monkey.jar 程序是由一个名为“monkey”的 Shell 脚本来启动执行，shell 脚本在 Android 文件系统中的存放路径是：/system/bin/monkey；这样就可以通过在 shell 窗口中执行: adb shell monkey {+命令参数}来进行 Monkey 测试了



从PC上启动monkey的执行示意图

4.4 Monkey 命令参数选项

Monkey 参数大全



4.4.1常用选项

- -help：打印帮助信息

```
C:\Users\Administrator> adb shell monkey -help
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

➢ -v：指定打印信息的详细级别

用于指定反馈信息级别（信息级别就是日志的详细程度），总共分 3 个级别，一个 -v 增加一个级别 ， 默认级别为 0 。-v -v -v 为最详细日志。分别对应的参数如下表所示：

级别	参数	说明	示例
Level 0	-v	说明缺省值，仅提供启动提示、测试完成和最终结果等少量信息	adb shell monkey -p com.android.caculators2 -v 100
Level 1	-v -v	提供较为详细的日志，包括每个发送到 Activity 的事件信息	adb shell monkey -p com.android.caculators2 -v -v 100
Level 2	-v -v -v	提供最详细的日志，包括了测试中选中/未选中的 Activity 信息	adb shell monkey -p com.android.caculators2 -v -v -v 100

4.4.2 约束选项

➢ -p：指定有效的 package

用此参数指定一个或多个包（Package，即 App），一个-p 对应一个有效 package。指定包之后，monkey 将只允许系统启动指定的 APP,如果不指定包，将允许系统启动设备中的所有 APP.

✧ 指定一个包： adb shell monkey -p com.android.caculators2 10

✧ 指定多个包：adb shell monkey -p com.android.caculators2 -p cn.emoney.wea -p com.android.caculators2 100

✧ 不指定包：adb shell monkey 100

- -c：activity 必须至少包含一个指定的 category

activity 必须至少包含一个指定的 category，才能被启动，否则启动不了；

4.4.3 事件选项

- -s：指定产生随机事件种子值，相同的种子值产生相同的事件序列

用于指定伪随机数生成器的 seed 值，如果 seed 相同，则两次 Monkey 测试所产生的事件序列也相同的。

Monkey 测试 1：adb shell monkey -p com.android.caculators2 -s 10 -v 100

Monkey 测试 2：adb shell monkey -p com.android.caculators2 -s 10 -v 100

两次测试的效果是相同的，因为模拟的用户操作序列（每次操作按照一定的先后顺序所组成的一系列操作，即一个序列）是一样的。**注意：不能完全重复上次操作，但是具体操作事件是相同的。比如点击->向左拖动->释放这几个事件这是相同的，但具体涉及的坐标是不相同的**。

- --throttle<毫秒>：指定用户操作（即事件）间的时延，单位是毫秒；

--throttle：每个事件结束后的间隔时间——降低系统的压力（如不指定，系统会尽快的发送事件序列）。

如：adb shell monkey -p com.android.caculators2 --throttle 5000 -v 100

- --pct-｛+事件类别｝｛+事件类别百分比｝

用于指定每种类别事件的数目百分比（在 Monkey 事件序列中，该类事件数目占总事件数目的百分比）

参数	说明
--pct-touch｛+百分比｝	调整触摸事件的百分比(**触摸事件**是一个 down-up 事件，它发生在屏幕上的某单一位置)
--pct-motion ｛+百分比｝	调整动作事件（滑动事件）的百分比(**动作事件**由屏幕上某处的一个 down 事件、一系列的伪随件机事和一个 up 事件组成)
--pct-trackball ｛+百分比｝	调整轨迹事件的百分比(**轨迹事件**由一个或几个随机的移动组成，有时还伴随有点击)
--pct-nav ｛+百分比｝	调整“基本”导航事件的百分比(**导航事件**由来自方向输入

	设备的 up/down/left/right 组成)
--pct-majornav {+百分比}	调整“主要”导航事件的百分比(这些导航事件通常引发图形界面中的动作，如：回退按键 back ke、菜单按键 menu key)
--pct-syskeys {+百分比}	系统按键事件的百分比（系统按键事件如：Home、Back、startCall、endCall、volumeControl)
--pct-appswitch 、 --pct-anyevent	--pct-appswitch （activity 之间的切换）、 --pct-anyevent （任意事件）

示例:

adb shell monkey -p com.android.caculators2 --pct-touch 10 1000

4.4.4调试选项

- --dbg-no-events：初始化启动的 activity，但是不产生任何事件。
- --hprof：指定该项后在事件序列发送前后会立即生成分析报告 —— 一般建议指定该项。
- --ignore-crashes：忽略崩溃
- --ignore-timeouts：忽略超时
- --ignore-security-exceptions：忽略安全异常
- --kill-process-after-error：发生错误后直接杀掉进程
- --monitor-native-crashes：跟踪本地方法的崩溃问题
- --wait-dbg：直到连接了调试器才执行 monkey 测试。

--ignore-crashes

- 用于指定当应用程序崩溃时（Force& Close 错误），Monkey 是否停止运行。如果使用此参数，即使应用程序崩溃，Monkey 依然会发送事件，直到事件计数完成。

  adb shell monkey -p com.android.caculators2 --ignore-crashes 1000

- 测试过程中即使程序崩溃，Monkey 依然会继续发送事件直到事件数目达到 1000 为止

  adb shell monkey -p com.android.caculators2 1000

  测试过程中，如果 caculators2 程序崩溃，Monkey 将会停止运行

- --ignore-timeouts

  - 用于指定当应用程序发生 ANR（Application No Responding）错误时，Monkey 是否停止运行。如果使用此参数，即使应用程序发生 ANR 错误，Monkey 依然会发送事件，直到事件计数完成。

    adb shell monkey -p com.android.caculators2 --ignore-timeouts 1000

- --ignore-security-exceptions

用于指定当应用程序发生许可错误时（如证书许可，网络许可等），Monkey 是否停止运行。如果使用此参数，即使应用程序发生许可错误，Monkey 依然会发送事件，直到事件计数完成。

adb shell monkey -p com.android.caculators2 --ignore-security-exception 1000

- --kill-process-after-error

用于指定当应用程序发生错误时，是否停止其运行。如果指定此参数，当应用程序发生错误时，应用程序停止运行并保持在当前状态

（注意：应用程序仅是静止在发生错误时的状态，系统并不会结束该应用程序的进程）。

adb shell monkey -p com.android.caculators2 --kill-process-after-error 1000

- --monitor-native-crashes

用于指定是否监视并报告应用程序发生崩溃的本地代码。

adb shell monkey -p com.android.caculators2 --monitor-native-crashes 1000

adb 全称是 Android Debug Bridge ，就是起个桥梁连接的作用，是 Android SDK 提供的 Debug 工具。借助 ADB 工具，我们可以管理终端设备或模拟器的状态，进行很多操作，如安装 APP、升级、运行 Shell 命令登录。如果是终端设备，ADB 就是连接 Android 手机与 PC 的桥梁，可以让用户在电脑上对手机进行全面操作；如果是模拟器，adb 也可以对模拟器进行操作。

4.5 Monkey 常用命令

4.5.1Monkey 的基本用法

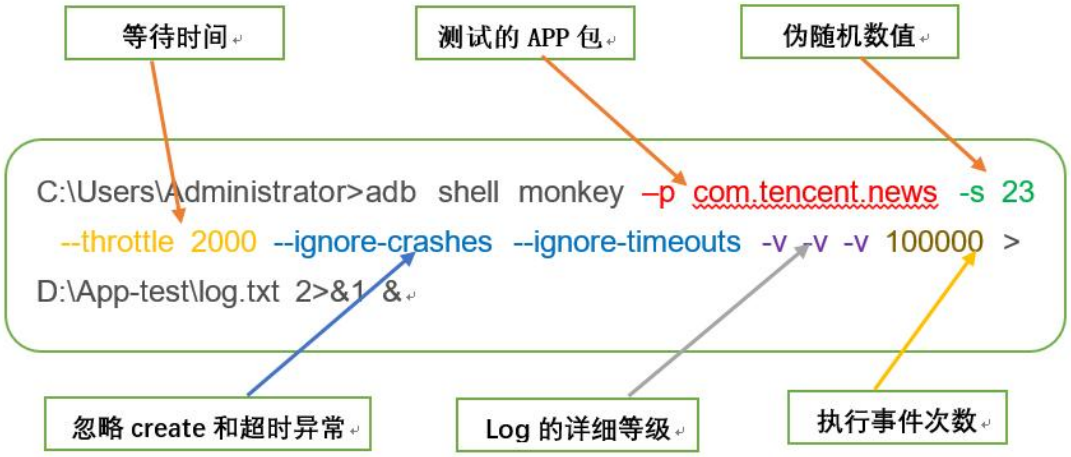
基本语法如下：

```
adb shell monkey [options]
```

如果不指定 options，Monkey 将以无反馈模式启动，并把事件任意发送到安装在目标环境中的全部包。下面是一个更为典型的命令行示例，它启动指定的应用程序，并向其发送 500 个伪随机事件：

```
adb shell monkey -p your.package.name -v 500
```

4.5.2Monkey 命令语句详解




```
adb shell monkey  -p com.tencent.news -s 23  --throttle 2000 --ignore-crashes --ignore-timeouts -v -v -v 100000 > D:\App-test\log.txt 2>&1 &
```

1. -p 后面接着的对应的包名，如果是整机测试，就不需要 -p package_name,自动随机测试系统中的所有应用包括系统 APP 和用户 APP；
2. -s 后面是对应的种子数，好像就是操作步骤，根据测试的经验，一般种子数在 23。同步测试的结果，一般种子的个数固定为 23，和选择的操作步骤就是同步的。为随机数的事件序列定一个值，若出现问题下次可以重复同样的序列进行排错。

3. --ignore-crashes --ignore-timeouts 这里是在 monkey 测试的过程中遇到 carash 或者 timeout 的情况时忽略掉，一般不设置时，出现 carash 或者 timeout 时，Monkey 测试会终止，这里是防止 Monkey 测试终止。

4. -v 指的是 Monkey 测试时打印 log 级别。

5. 100000 这里是指点击的次数，根据测试的经验，对于单个应用程序这个次数设置在 100000 次就可以了；如果是整机，一般设置在 500000 次。

6. D:\App-test\log.txt 测试的 log 记录在本机上 D:\App-test\下面的 log.txt 里面，这个名字可以自己写。

7. 2>&1 固定的写法，这个也很重要，代表的意思是中间忽略的异常的日志一并输入到指定的文件中。

8. 最后单独的一个"&" 是一旦 Monkey 测试开始了，之后可以拔掉数据线，不会影响 Monkey 测试。

9. 测试所有模块,即不指定 APP 包：

```
adb shell monkey  -s 23 --ignore-crashes --ignore-timeouts  - v -v -v 100000> D:\data\local\tmp\log.txt 2>&1 &
```

4.5.3 Monkey 常用实战命令

- 触摸事件占 30%，忽略 crash 和超时，每个事件间隔 250ms，输出最详细日志，执行 500 万次

```
adb shell monkey -p your.package.name  --pct-touch 30 --ignore-crashes  --ignore-timeouts --throttle 250 -s 12  -v -v -v 5000000
```

- 触摸事件占 30%，滑动事件占 20%，忽略 crash、超时、安全等异常，每个事件间隔 1000ms，输出最详细日志并保存到本地的目录下，执行 10 万次

```
adb shell monkey -p com.tencent.news -s 35 --pct-touch 30 --pct-motion 20 --throttle 1000 --ignore-crashes --ignore-timeouts --ignore-security-exceptions -v -v -v 100000 > d:\app-test\app_monkey_log.log
```

- 触摸事件占 30%，滑动事件占 20%，忽略 crash 和超时异常，每个事件间隔 1000ms，输出基本日志并保存到本地的目录下文件名按照日期时间格式生成，执行 1 万次

```
adb shell monkey -p com.tencent.news --ignore-crashes --ignore-timeouts -v
10000 > d:\App-Test\tencent-%Date:~0,4%%Date:~5,2%%Date:~8,2%%Time:~0,2%%Tim
e:~3,2%%Time:~6,2%.log
```

- 执行 monkey 每个事件间隔 1s，将正确和错误的日志都输出最详细的日志并保存到系统的目录下 ，执行 1 万次

```
adb shell monkey  -p com.android.mms  -s 100 --throttle 1000  -v  -v  -v 150
00 > /mnt/sdcard/monkey_test.txt 2
```

- 屏幕截图脚本

```
adb shell screencap -p "/mnt/shared/shared/%date:~0,4%%date:~5,2%%date:~8,2%%tim
e:~0,2%%time:~3,2%%time:~6,2%.png"
```

4.6 Monkey 的测试策略

- 单个 apk 的验收测试时，使用单一 apk 且不忽略异常的命令执行
- 单个 apk 的解决问题的测试时，使用单一 apk 且忽略异常的命令执行。这样可以在一次执行的过程中发现应用程序中的多个问题。
- 单个 apk 的应用程序的压力/健壮性测试时，主要缩短 monkey 测试中事件与事件之间的延迟时间，验证在快速的事件响应的过程中，程序是否能正常运行。将--throttle 的值设定为 500 或者更小，一般都使用 500 毫秒的延迟事件。
- 在进行 apk 的集合测试（测试对象为多个 APP，手机测试）时，对于高频率使用的 apk、长时间使用的 apk 都要包含在执行的应用程序中间。

4.7 Monkey 测试实例分析

4.7.1Monkey 测试实例

对“计算器”进行测试步骤：

➢ 在测试 app 之前需要知道被测程序的包名：

1. 通过 adb shell 进入模拟器设备，
2. 访问模拟器中 APP 的路径 cd /data/data;
3. 使用 ls 命令查看所有的包名，找到我们需要的包名“com.android.calculator2”即是“计算器”的包名，

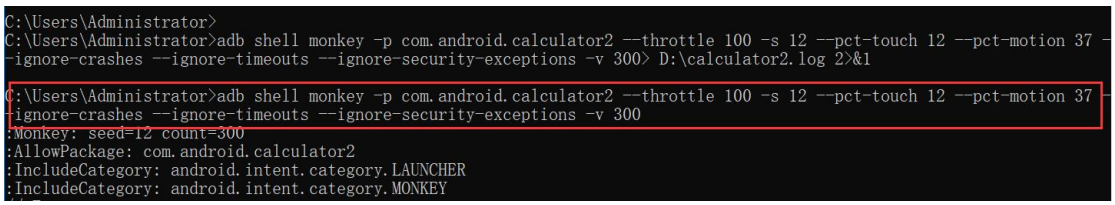
如下图所示。

➢ APP 测试操作：

在控制台输入 monkey 命令,对“计算器”APP 进行 300 次的自动化测试，其中点击事件占 37%，滑动事件占 12%，事件间隔为 100ms,过程中忽略所有异常情况，并将输出的 0 级日志保存到本地；

命令语句：

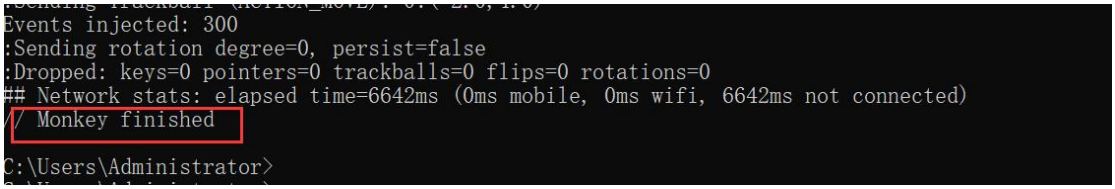
```
adb shell monkey -p com.android.calculator2 --throttle 100 -s 12 --pct-touch 12 --pct-motion 37 --ignore-crashes --ignore-timeouts --ignore-security-exceptions -v 300 > D:\calculator2.log 2>&1
```



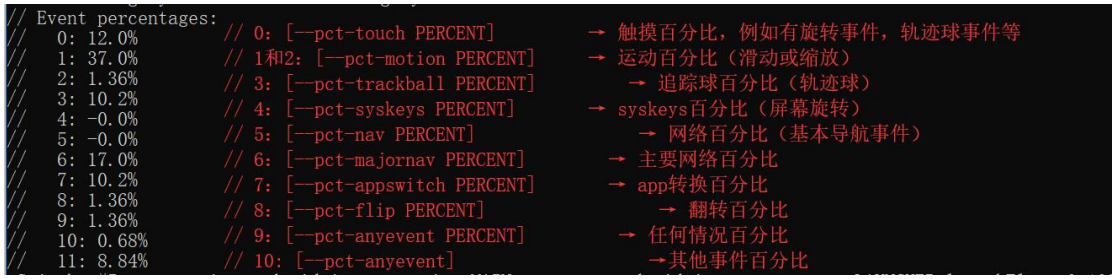
4.7.2Monkey 测试结果分析

➢ 结果初步分析

正常情况，如果 Monkey 测试顺利执行完成，在 log 的最后会打印出当前执行事件的次数和所花费的时间； // Monkey finished 代表执行完成



测试中的事件比例如下：



// 分别代表的意思如下 0~10

// 0：[--pct-touch PERCENT] → 触摸百分比，例如有旋转事件，轨迹球事件等

// 1 和 2：[--pct-motion PERCENT] → 运动百分比（滑动或缩放）

// 3：[--pct-trackball PERCENT] → 追踪球百分比（轨迹球）

// 4：[--pct-syskeys PERCENT] → syskeys 百分比（屏幕旋转）

// 5：[--pct-nav PERCENT] → 网络百分比（基本导航事件）

// 6：[--pct-majornav PERCENT] → 主要网络百分比

// 7：[--pct-appswitch PERCENT] → app 转换百分比

// 8：[--pct-flip PERCENT] → 翻转百分比

// 9：[--pct-anyevent PERCENT] → 任何情况百分比

// 10：[--pct-anyevent] →其他事件百分比

➢ 结果异常分析

Monkey 测试出现错误后，一般的查错步骤为以下几步：

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--复现

一般的测试结果分析：

- ANR 问题：在日志中搜索“ANR”
- 崩溃问题：在日志中搜索“CRASH”
- 异常问题：在日志中搜索“Exception” Force Close
- 错误问题：在日志中搜索“Error”，“Failed”，“Warning”

一般在如下几种情况会产生 log 文件:

1) 程序异常退出，uncausedexception （Fatal）

2) 程序强制关闭，ForceClosed (简称 FC) （Fatal）

3) 程序无响应，ApplicationNo Response（简称 ANR)

ANR 出现的情况有以下两种：

一、 界面操作按钮的点击等待响应时间超过 5 秒

二、 HandleMessage 回调函数执行超过 10 秒，BroadcasterReciver 里的 onRecive（）方法处理超过 10 秒

➢ 测试 Bug 重现

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

➢ 详细日志分析

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。

首先我们需要查看 Monkey 测试中是否出现了 ANR 或者异常，具体方法如上述。

然后我们要分析 log 中的具体信息，方法如下：

查看 log 中第一个 **Switch**，主要是查看 Monkey 执行的是那一个 **Activity**，譬如下面的 log 中，执行的是 com.tencent.smtt.SplashActivity，在下一个 swtich 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。

```
:Switch:#Intent;action=android.intent.action.MAIN;category=android.intent.category.
LAUNCHER;launchFlags=0x10000000;component=com.tencent.smtt/.SplashActivity;end
  // Allowing start of Intent {act=android.intent.action.MAIN
cat=[android.intent.category.LAUNCHER]cmp=com.tencent.smtt/.SplashActivity } in
package com.tencent.smtt
```

在下面的 log 中，Sending Pointer ACTION_DOWN 和 Sending Pointer ACTION_UP 代表当前执行了一个单击的操作;

Sleeping for 500 milliseconds 这句 log 是执行 Monkey 测试时，throttle 设定的间隔时间，每出现一次，就代表一个事件。

SendKey(ACTION_DOWN) //KEYCODE_DPAD_DOWN 代表当前执行了一个点击下导航键的操作;

Sending Pointer ACTION_MOVE 代表当前执行了一个滑动界面的操作。

:Sending Pointer ACTION_DOWN x=47.0 y=438.0

:Sending Pointer ACTION_MOVE x=-2.0 y=-4.0

ANR

如果 Monkey 测试顺利执行完成，在 log 的最后，会打印出当前执行事件的次数和所花费的时间；// Monkey finished 代表执行完成。Monkey 执行中断，在 log 的最后也能查看到当前已执行的次数。Monkey 执行完成的 log 具体如下：

```
Events injected: 6000

:Dropped: keys=0 pointers=9 trackballs=0 flips=0

## Network stats: elapsed time=808384ms (0ms mobile, 808384ms wifi, 0msnot connected)
// Monkey finished
```

5 App 测试常见问题

1. 什么是 Activity

什么是 activity，这个前两年出去面试 APP 测试岗位，估计问的最多了，特别是一些大厂，先问你是不是做过 APP 测试，那好，你说说什么是 activity?

如果没看过 android 的开发原理，估计这个很难回答，要是第一个问题就被难住了，面试的信心也会失去一半了，士气大减。

Activity 是 Android 的四大组件之一，也是平时我们用到最多的一个组件，可以用来显示 View。

官方的说法是 Activity 一个应用程序的组件，它提供一个屏幕来与用户交互，以便做一些诸如打电话、发邮件和看地图之类的事情，原话如下：

An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map.

Activity 是一个 Android 的应用组件，它提供屏幕进行交互。每个 Activity 都会获得一个用于绘制其用户界面的窗口，窗口可以充满屏幕也可以小于屏幕并浮动在其他窗口之上。

一个应用通常是由多个彼此松散联系的 Activity 组成，一般会指定应用中的某个 Activity 为主活动，也就是说首次启动应用时给用户呈现的 Activity。将 Activity 设为主活动的方法

当然 Activity 之间可以进行互相跳转，以便执行不同的操作。每当新 Activity 启动时，旧的 Activity 便会停止，但是系统会在堆栈也就是返回栈中保留该 Activity。

当新 Activity 启动时，系统也会将其推送到返回栈上，并取得用户的操作焦点。当用户完成当前 Activity 并按返回按钮是，系统就会从堆栈将其弹出销毁，然后回复前一 Activity 当一个 Activity 因某个新 Activity 启动而停止时，系统会通过该 Activity 的生命周期回调方法通知其这一状态的变化。

Activity 因状态变化每个变化可能有若干种，每一种回调都会提供执行与该状态相应的特定操作的机会

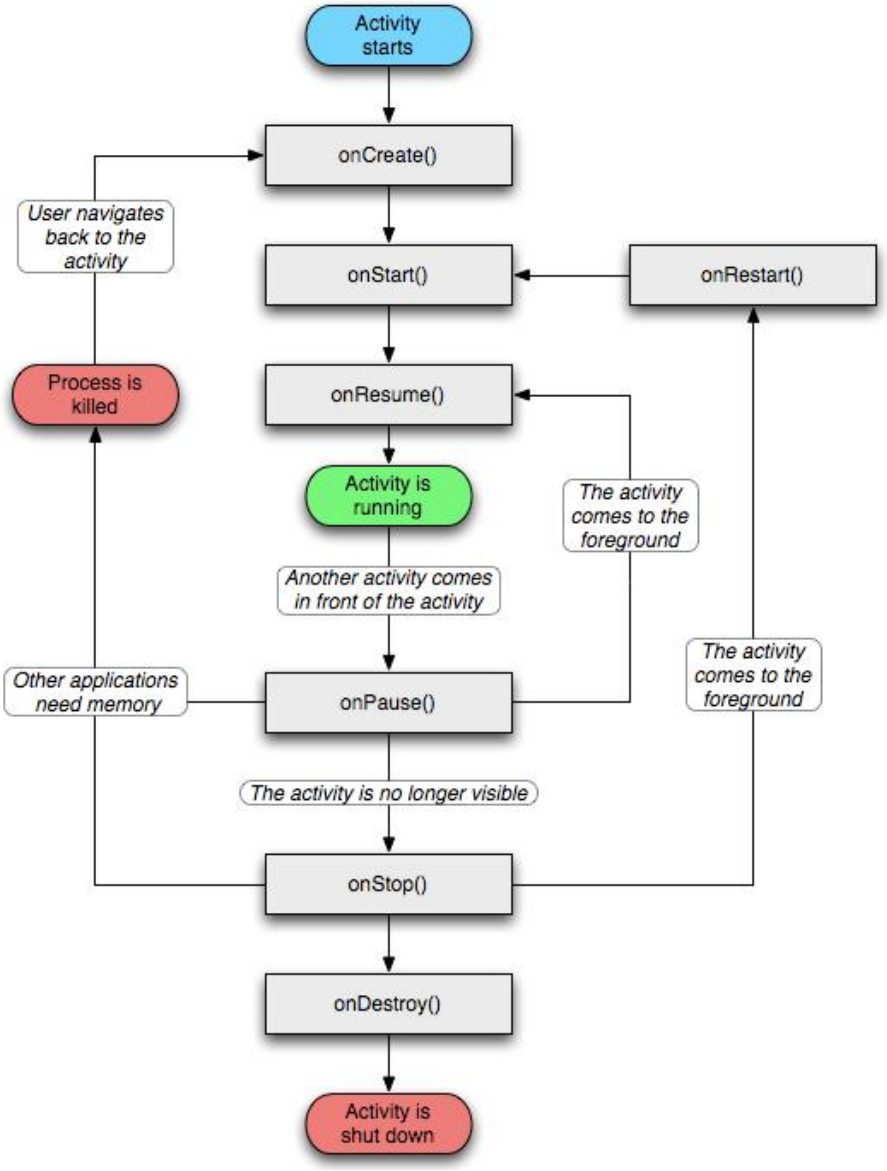
2. Activity 生命周期?

周期即活动从开始到结束所经历的各种状态。生命周期即活动从开始到结束所经历的各个状态。从一个状态到另一个状态的转变，从无到有再到无，这样一个过程中所经历的状态就叫做生命周期。

Activity 本质上有四种状态：

1) 运行（Active/Running）:Activity 处于活动状态，此时 Activity 处于栈顶，是可见状态，可以与用户进行交互；
2) 暂停（Paused）:当 Activity 失去焦点时，或被一个新的非全面屏的 Activity，或被一个透明的 Activity 放置在栈顶时，Activity 就转化为 Paused 状态。此刻并不会被销毁，只是失去了与用户交互的能力，其所有的状态信息及其成员变量都还在，只有在系统内存紧张的情况下，才有可能被系统回收掉
3) 停止(Stopped):当 Activity 被系统完全覆盖时，被覆盖的 Activity 就会进入 Stopped 状态，此时已不在可见，但是资源还是没有被收回
4) 系统回收（Killed）:当 Activity 被系统回收掉，Activity 就处于 Killed 状态，如果一个活动在处于停止或者暂停的状态下，系统内存缺乏时会将其结束（finish）或者杀死（kill）。这种非正常情况下，系统在杀死或者结束之前会调用 onSaveInstance()方法来保存信息，同时，当 Activity 被移动到前台时，重新启动该 Activity 并调用 onRestoreInstance()方法加载保留的信息，以保持原有的状态。

在上面的四中常有的状态之间，还有着其他的生命周期来作为不同状态之间的过度，用于在不同的状态之间进行转换，生命周期的具体说明见下。



3. Android 四大组件

Android 四大基本组件：Activity、BroadcastReceiver 广播接收器、ContentProvider 内容提供者、Service 服务。

Activity:

应用程序中，一个 Activity 就相当于手机屏幕，它是一种可以包含用户界面的组件，主要用于和用户进行交互。一个应用程序可以包含许多活动，比如事件的点击，一般都会触发一个新的 Activity。

BroadcastReceiver 广播接收器：

应用可以使用它对外部事件进行过滤只对感兴趣的外部事件(如当电话呼入时，或者数据网络可用时)进行接收并做出响应。广播接收器没有用户界面。然而，它们可以启动一个 activity 或 serice 来响应它们收到的信息，或者用 NotificationManager 来通知用户。通知可以用很多种方式来吸引用户的注意力——闪动背灯、震动、播放声音等。一般来说是在状态栏上放一个持久的图标，用户可以打开它并获取消息。

ContentProvider 内容提供者:

内容提供者主要用于在不同应用程序之间实现数据共享的功能，它提供了一套完整的机制，允许一个程序访问另一个程序中的数据，同时还能保证被访问数据的安全性。只有需要在多个应用程序间共享数据时才需要内容提供者。例如：通讯录数据被多个应用程序使用，且必须存储在一个内容提供者中。它的好处：统一数据访问方式。

Service 服务：

是 Android 中实现程序后台运行的解决方案,它非常适合去执行那些不需要和用户交互而且还要长期运行的任务（一边打电话，后台挂着 QQ）。服务的运行不依赖于任何用户界面，即使程序被切换到后台，或者用户打开了另一个应用程序，服务扔然能够保持正常运行，不过服务并不是运行在一个独立的进程当中，而是依赖于创建服务时所在的应用程序进程。当某个应用程序进程被杀掉后，所有依赖于该进程的服务也会停止运行（正在听音乐，然后把音乐程序退出）。

4. Android 和 IOS 测试区别？

App 测试中 ios 和 Android 有哪些区别呢？

1) Android 长按 home 键呼出应用列表和切换应用，然后右滑则终止应用；
2) 多分辨率测试，Android 端 20 多种，ios 较少；
3) 手机操作系统，Android 较多，ios 较少且不能降级，只能单向升级；新的 ios 系统中的资源库不能完全兼容低版本中的 ios 系统中的应用,低版本 ios 系统中的应用调用了新的资源库，会直接导致闪退（Crash）；
4) 操作习惯：Android，Back 键是否被重写，测试点击 Back 键后的反馈是否正确；应用数据从内存移动到 SD 卡后能否正常运行等；
5) push 测试：Android：点击 home 键，程序后台运行时，此时接收到 push，点击后唤醒应用，此时是否可以正确跳转；ios，点击 home 键关闭程序和屏幕锁屏的情况（红点的显示）；
6) 安装卸载测试：Android 的下载和安装的平台和工具和渠道比较多，ios 主要有 app store，iTunes 和 testflight 下载；
7) 升级测试：可以被升级的必要条件：新旧版本具有相同的签名；新旧版本具有相同的包名；有一个标示符区分新旧版本（如版本号），对于 Android 若有内置的应用需检查升级之后内置文件是否匹配（如内置的输入法）

另外：对于测试还需要注意一下几点：

1) 并发（中断）测试：闹铃弹出框提示，另一个应用的启动、视频音频的播放，来电、用户正在输入等，语音、录音等的播放时强制其他正在播放的要暂停；
2) 数据来源的测试：输入，选择、复制、语音输入，安装不同输入法输入等；
3) push（推送）测试：在开关机、待机状态下执行推送，消息先死及其推送跳转的正确性；应用在开发、未打开状态、应用启动且在后台运行的情况下是 push 显示和跳转否正确；推送消息阅读前后数字的变化是否正确；多条推送的合集的显示和跳转是否正确；
4) 分享跳转：分享后的文案是否正确；分享后跳转是否正确，显示的消息来源是否正确；
5) 触屏测试：同时触摸不同的位置或者同时进行不同操作，查看客户端的处理情况，是否会 crash 等

5. App 出现 ANR，是什么原因导致的？

那么导致 ANR 的根本原因是什么呢？简单的总结有以下两点：

- 主线程执行了耗时操作，比如数据库操作或网络编程
- 其他进程（就是其他程序）占用 CPU 导致本进程得不到 CPU 时间片，比如其他进程的频繁读写操作可能会导致这个问题。

细分的话，导致 ANR 的原因有如下几点：

1) 耗时的网络访问
2) 大量的数据读写
3) 数据库操作
4) 硬件操作（比如 camera）
5) 调用 thread 的 join()方法、sleep()方法、wait()方法或者等待线程锁的时候
6) service binder 的数量达到上限
7) system server 中发生 WatchDog ANR
8) service 忙导致超时无响应
9) 其他线程持有锁，导致主线程等待超时
10) 其它线程终止或崩溃导致主线程一直等待。

6. App 出现 crash 原因有哪些？

为什么 App 会出现崩溃呢？百度了一下，查到和 App 崩溃相关的几个因素：内存管理错误，程序逻辑错误，设备兼容，网络因素等，如下：

1) 内存管理错误：可能是可用内存过低，app 所需的内存超过设备的限制，app 跑不起来导致 App crash。或是内存泄露，程序运行的时间越长，所占用的内存越大，最终用尽全部内存，导致整个系统崩溃。亦或非授权的内存位置的使用也可能会导致 App crash。
2) 程序逻辑错误：数组越界、堆栈溢出、并发操作、逻辑错误。e.g. app 新添加一个未经测试的新功能，调用了一个已释放的指针，运行的时候就会 crash。
3) 设备兼容：由于设备多样性，app 在不同的设备上可能会有不同的表现。
4) 网络因素：可能是网速欠佳，无法达到 app 所需的快速响应时间，导致 app crash。或者是不同网络的切换也可能会影响 app 的稳定性。

7. App 对于不稳定偶然出现 ANR 和 Crash 时怎么处理的？

app 偶然出现 ANR 和 Crash 是比较头疼的问题，由于偶然出现无法复现步骤，这也是一个测试人员必备的技能，需要抓日志。查看日志主要有 3 个方法：

- 方法一：app 开发保存错误日志到本地

一般 app 开发在 debug 版本，出现 ANR 和 Crash 的时候会自动把日志保存到本地实际的 sd 卡上，去对应的 app 目录取出来就可以了

➢ 方法二：实时抓取

当出现偶然的 crash 时候，这时候可以把手机拉到你们 app 开发那，手机连上他的开发代码的环境，有 ddms 会抓日志，这时候出现 crash 就会记录下来日志。

尽量重复操作让 bug 复现就可以了，也可以自己开着 logcat，保存日志到电脑本地,参考这篇：https://www.cnblogs.com/yoyoketang/p/9101365.html

```
adb logcat | find "com.sankuai.meituan" >d:\hello.txt
```

➢ 方法三：第三方 sdk 统计工具

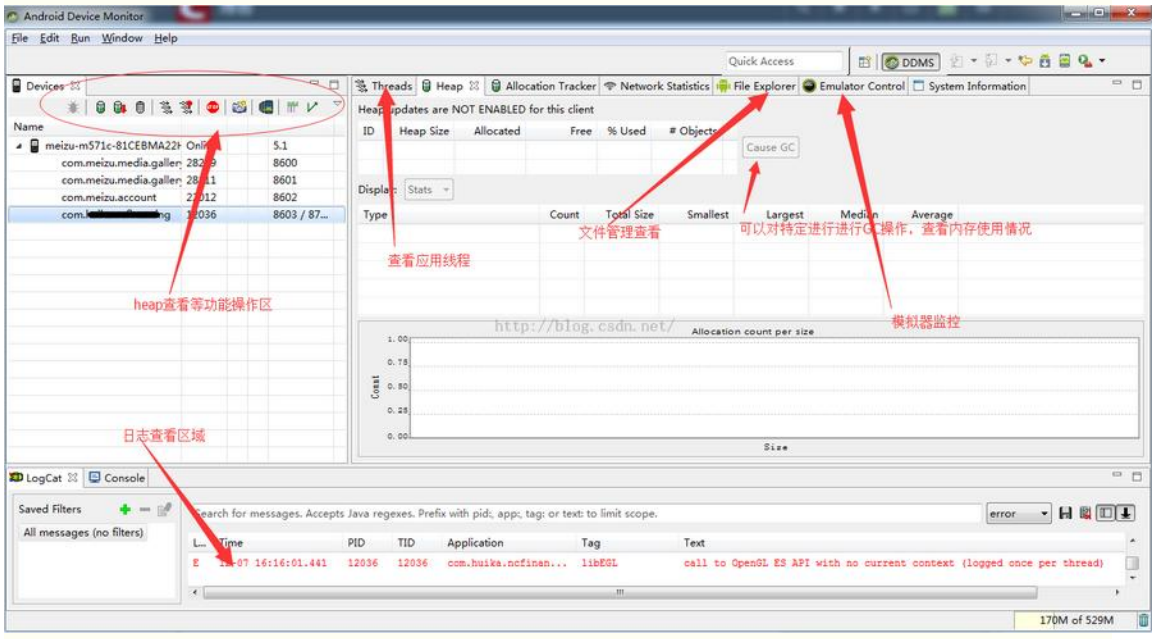
一般接入了第三方统计 sdk,比如友盟统计，在友盟的后台会抓到报错的日志

8. app 的日志如何抓取?

app 本身的日志，可以用 logcat 抓取，参考这篇：https://www.cnblogs.com/yoyoketang/p/9101365.html

```
adb logcat | find "com.sankuai.meituan" >d:\hello.txt
```

也可以用 ddms 抓取，手机连上电脑，打开 ddms 工具，或者在 Android Studio 开发工具中，打开 DDMS



关于 ddms 更多的功能，参考这篇：https://www.cnblogs.com/gaobig/p/5029381.html

9. 你平常会看日志吗，一般会出现哪些异常（Exception）？

这个主要是考察你会不会看日志，是不是看得懂 java 里面抛出的异常，Exception

一般面试中 java Exception（runtimeException ）是必会被问到的问题

app 崩溃的常见原因应该也是这些了。常见的异常列出四五种，是基本要求。

常见的几种如下：

- NullPointerException - 空指针引用异常
- ClassCastException - 类型强制转换异常。

- IllegalArgumentException - 传递非法参数异常。
- ArithmeticException - 算术运算异常
- ArrayStoreException - 向数组中存放与声明类型不兼容对象异常
- IndexOutOfBoundsException - 下标越界异常
- NegativeArraySizeException - 创建一个大小为负数的数组错误异常

NumberFormatException - 数字格式异常

SecurityException - 安全异常

UnsupportedOperationException - 不支持的操作异常