**目 录**

[1. 软件设计文档摘要 3](#_Toc418972758)

[1.1. 本项目的意义 3](#_Toc418972759)

[1.2. 本文组织结构 3](#_Toc418972760)

[2. 系统需求分析 4](#_Toc418972761)

[2.1. 系统概述 4](#_Toc418972762)

[2.1.1. 系统应用目的 4](#_Toc418972763)

[2.1.2. 功能需求描述 4](#_Toc418972764)

[2.2. 系统用例分析 4](#_Toc418972765)

[2.2.1. 确定系统的角色 4](#_Toc418972766)

[2.2.2. 确定系统的用例 5](#_Toc418972767)

[2.2.3. 制定用例规约 5](#_Toc418972768)

[2.3. 系统性能需求 12](#_Toc418972769)

[2.4. 本章小结 13](#_Toc418972770)

[3. 系统设计 14](#_Toc418972771)

[3.1. 系统网络拓扑结构设计 14](#_Toc418972772)

[3.2. 系统功能模块总体设计 14](#_Toc418972773)

[3.3. 系统功能模块详细设计 15](#_Toc418972774)

[3.3.1. 欢迎界面 15](#_Toc418972775)

[3.3.2. 手机防盗 17](#_Toc418972776)

[3.3.3. 手机杀毒 20](#_Toc418972777)

[3.3.4. 骚扰拦截 21](#_Toc418972778)

[3.3.5. APP管理 23](#_Toc418972779)

[3.3.6. 来电查询 24](#_Toc418972780)

[3.3.7. 短信备份 26](#_Toc418972781)

[3.3.8. 清理优化 29](#_Toc418972782)

[3.3.9. 流量统计 30](#_Toc418972783)

[3.4. 数据库表结构设计 32](#_Toc418972784)

[3.4.1. 病毒数据库 32](#_Toc418972785)

[3.4.2. 来电查询数据库 32](#_Toc418972786)

[3.4.3. 骚扰拦截数据库 33](#_Toc418972787)

[3.5. 本章小结 33](#_Toc418972788)

[4. 系统实现 34](#_Toc418972789)

[4.1. 系统开发平台 34](#_Toc418972790)

[4.2. 系统功能实现 34](#_Toc418972791)

[4.2.1. 欢迎界面 34](#_Toc418972792)

[4.2.2. 手机防盗 35](#_Toc418972793)

[4.2.3. 手机杀毒 39](#_Toc418972794)

[4.2.4. 骚扰拦截 41](#_Toc418972795)

[4.2.5. APP管理 42](#_Toc418972796)

[4.2.6. 来电查询 42](#_Toc418972797)

[4.2.7. 短信备份 43](#_Toc418972798)

[4.2.8. 清理优化 44](#_Toc418972799)

[4.2.9. 流量统计 46](#_Toc418972800)

[4.3. 本章小结 47](#_Toc418972801)

[5. 系统测试 48](#_Toc418972802)

[5.1. 测试环境 48](#_Toc418972803)

[5.2. 测试内容 49](#_Toc418972804)

[5.2.1. 功能测试 49](#_Toc418972805)

[5.2.2. 综合测试 51](#_Toc418972806)

[5.2.3. 测试结果 52](#_Toc418972807)

[5.3. 本章小结 54](#_Toc418972808)

# 软件设计文档摘要

## 本项目的意义

本项目通过对Android系统架构、权限管理、内核源码、设计范例的研究，在充分调研国内外相关智能手机管家应用程序产品的基础上，设计并实现了基于Android平台的智能手机管家系统，目的在于解决人们平时生活中对于手机的日常管理问题。基本具备目前国内市场上同类管家APP的核心功能，垃圾清理、病毒查杀、信息备份、应用程序管理、来电查询、骚扰拦截、手机防盗。本项目也是对个人本科学习阶段所学到的调研、分析、建模、设计、编程和调试的知识进行一次综合实践。

## 本文组织结构

第一章是需求分析，通过对用户需求和市场上现有产品的分析调研，使用用例说明用户对于管家系统的各种功能需求。

第二章是系统功能设计，将用户需求分析确定的用例转换为具体功能和业务逻辑流程，并用统一建模语言（UML）的活动图描述。

第三章是系统功能实现，主要说明实现的技术逻辑和关键实现代码。

第四章是系统测试，通过对功能模块的测试和系统整体测试，检验软件的质量。

# 系统需求分析

## 系统概述

### 系统应用目的

本系统的应用目的是把日常手机管理的常用功能结合起来，充分考虑了手机管理、手机防盗、手机杀毒、来电显示、短信备份等实际应用场景，为广大用户提供方便快捷、实用可靠的手机管理服务。

### 功能需求描述

通过开题前期的市场调研，分析国内外市面上常见的智能手机管家系统，分析他们的功能设计、实现原理，根据用户需求对软件的总体架构进行规划，提炼出其共通的实用性较强的基础功能，主要功能如下：

* 1. 手机防盗功能：通过设置向导设置登录密码、防盗安全号码、绑定SIM卡、锁屏密码、开启相关远程操控手机权限，让手机在丢失后可通过安全号码的短信操控手机，利用GPS定位丢失手机的位置。
  2. 手机杀毒功能：通过可定期升级的内置病毒特征码数据库，扫描查杀手机病毒APP。
  3. 骚扰拦截功能：通过向数据库添加需要拦截的电话号码，选择拦截模式，可拦截短信和自动挂断骚扰来电，不影响手机系统正常使用。
  4. 来电查询功能：通过内置的归属地数据库识别来电的归属地，用户也可以开启自动查询来电属地服务。
  5. APP管理功能：通过查询用户安装的APP，提供一个统一的界面启动和卸载APP。
  6. 短信备份功能：通过数据的序列化和反序列化，将系统短信库的内存备份在SD卡以便以后用户还原短信。
  7. 清理优化功能：通过对手机缓存的清理，释放手机空间，提高手机运行速度。
  8. 流量统计功能：通过检测APP使用流量，让用户了解流量杀手应用。

## 系统用例分析

用例（use case）分析，常见于软件工程中，主要用来分析系统是如何对外界进行请求的一种方法。通过分析用户的使用场景以获取明确的业务需求。每个用例提供一个或多个用于说明系统和最终用户或其他系统的交互场景。本节通过三个步骤完成用例分析：确定系统中的行为者（Actor），确定用例（Use Case）并绘制系统的用例图，用例规约的制定。

### 确定系统的角色

角色（Actor）是指系统或其他系统的交互对象，它代表一个外部实体的作用。确定角色是为了明确与系统的交互的人或实体，方便软件开发者确定系统的交互行为。角色作为交互的对象可触发系统的输入事件或请求系统的输出事件，以此驱动系统的相应的运行逻辑。

本系统基于Android平台，为用户提供智能手机的日常管理服务。主要使用者为用户本人，而管家系统的手机防盗功能则需要他人手机作为安全号码进行远程安全操作。因此该系统中的角色为安装该系统的日常用户和绑定为安全号码的安全用户。

### 确定系统的用例

用例（Use Case）主要是用来说明如何与最终用户或其他系统的相互作用。本系统为用户提供智能手机的日常管理需求，分为几大功能，每一个功能是一个用例场景。系统的总用例图如下所示。



图 3 系统总用例图

### 制定用例规约

对于系统中的每一个用例，需要用一个用例规约来描述它。用例规约能够驱动用户的需求，对系统设计模型和实现模型的建立有重要的指导作用。下面对该系统中的每个用例规约进行说明。

#### 欢迎界面：



图 4 欢迎界面用例图

欢迎界面包括系统初始化、联网更新系统版本、联网更新数据库，其用例规约如表所示。

表 1 欢迎界面用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入欢迎界面 * 检查服务器是否有新版本 * 用户确认更新，下载新版本Apk安装包 * 弹出安装界面请用户安装 |
| 备选事件流 | * 网路异常则直接进入主页面 |
| 特殊要求 | 需要联网 |
| 前置条件 | 用户未关闭自动更新功能 |
| 后置条件 | 系统成功安装新版本 |

#### 手机防盗：



图 5 手机防盗用例图

手机防盗包括设置密码、绑定SIM卡、设置安全号码、开启防盗等用例，其用例规约如表所示。

表 2 手机防盗用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入设置向导界面 * 阅读相关功能描述 * 绑定SIM卡，并设置安全号码 * 开启系统相关权限和开启防盗信息 * 进入防盗状态界面 |
| 备选事件流 | * 用户成功完成设置向导，直接进入防盗状态界面 |
| 特殊要求 | 无 |
| 前置条件 | 该用户第一次使用或再次想重新设置信息 |
| 后置条件 | 用户成功登陆 |

#### 手机杀毒



图 6 手机杀毒用例图

手机杀毒包括杀毒和卸载病毒APP，其用例规约如下。

表 3 手机杀毒用例规约

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入杀毒界面 * 点击查杀病毒 * 扫描手机中安装的所有APP * 列出病毒APP并提供卸载按钮 |
| 备选事件流 | 无 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 查杀病毒完成，显示杀毒结果 |

#### 骚扰拦截



图 7 拦截骚扰用例图

骚扰拦截包括增删改查想要拦截的电话号码，选择拦截模式，开启拦截服务，其用例规约如下。

表 4 拦截骚扰用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入拦截界面 * 点击添加号码，填入号码和选择拦截模式 * 查看或修改或删除现有拦截电话 * 开启服务 |
| 备选事件流 | * 若用户未开启服务则提示开启 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 系统后台启动监听骚扰电话的服务 |

#### APP管理



图 8 APP管理用例图

APP管理主要包括查看手机剩余空间、APP的启动和卸载，其用例规约如下。

表 5 APP管理用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入APP管理界面 * 等待系统扫描手机所安装的APP * 选择指定应用程序卸载或启动 * 刷新列表，显示新的手机APP状态 |
| 备选事件流 | 无 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 系统重新检索相应的APP |

#### 来电查询



图 9 来电查询用例图

来电查询主要包括自助查询手机号归属地、开启来电自动查询服务，其用例规约如下。

表 6 来电查询用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入来电查询界面 * 输入所需查询电话号码 * 获得归属地地址 |
| 备选事件流 | * 开启来电自动查询服务 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 系统启动监听来电的服务 |

#### 短信备份



图 10 短信备份用例图

短信备份主要包括短信备份和短信还原，其用例规约如下。

表 7 短信备份用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入百宝箱界面 * 选择短信备份 |
| 备选事件流 | * 短信还原 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 系统序列化短信并存储在SD卡或系统反序列化将短信还原 |

#### 流量统计



图 11 流量统计用例图

流量统计主要包括手机流量查询，其用例规约如下。

表 8 流量统计用例图

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入流量查询界面 * 等待系统扫描手机所安装的APP * 查看按流量排序的APP |
| 备选事件流 | 无 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 无 |

#### 清理优化



图 12 清理优化用例图

清理优化主要包括手机缓存扫描、一键清除，其用例规约如下。

表 9 清理优化用例规约图

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 进入清理优化界面 * 等待系统扫描手机所安装的APP * 查看缓存大小 * 根据情况自助清理或一键清理 |
| 备选事件流 | 无 |
| 特殊要求 | 无 |
| 前置条件 | 无 |
| 后置条件 | 系统重新扫描手机APP缓存 |

#### 远程控制



图 13 远程控制用例图

远程主要包括绑定的安全用户通过手机短信命令查找丢失手机的位置、播放警报确认手机具体携带者、锁死屏幕和数据销毁防止隐私泄露，其用例规约如下。

表 10 远程控制用例规约表

|  |  |
| --- | --- |
| 事项 | 内容 |
| 事件流 | * 编辑短信命令 * 向丢失手机发送命令 * 系统根据相应命令执行 |
| 备选事件流 | 无 |
| 特殊要求 | 无 |
| 前置条件 | 系统成功接受短信 |
| 后置条件 | 系统完成相应命令 |

## 系统性能需求

本系统需要为用户提供各种手机管理功能，需要处理不同的逻辑和不同的任务处理的复杂性。比如面对查询数据库进行杀毒的业务逻辑，为了不影响用户交互体验，需要使用多线程的技术解决卡顿问题；若开启了骚扰拦截则需要持续对手机电话和短信进行监听并将监听到的内容查询数据库进行比对，由于Android平台限制Broadcast Receiver只能执行10秒，因此需要分工合作，Broadcast Receiver只负责监听并转发处理逻辑去在系统后台执行的Service服务。因此，该系统在设计时需要考虑以下性能指标，保证系统的效率和稳定运行。

稳定性要求：在涉及复杂逻辑时使用多线程技术防止阻塞用户的Activity交互界面引发ANR运行时异常。

运行要求：在系统发生异常或退出时保证启动垃圾回收机制，清理垃圾内存和服务从而不会造成手机运行卡顿。

兼容性要求：搭载Android的手机种类以及其上运行的Android版本众多，需要达到多屏幕适配要求防止交互逻辑失真，同时为了能正常运行，需要注意编程使用的API和编译器版本以保持良好的向下兼容。

可维护性要求：系统应保持良好的代码风格和注释，严格按照MVC设计模式，定义良好的接口从而达到组件的高内聚低耦合

## 本章小结

本章节主要对开发管家APP的需求进行了分析。首先描述系统的应用目的和功能需求。然后对涉及的每个功能从用户使用的角度进行了详细的用例分析，并说明了每个功能的用例规约。最后，描述了系统的性能需求。

# 系统设计

系统的总体设计是建立在需求分析基础上，通过分析用户用例，对软件总体结构进行规划，确定系统网路拓扑结构，设计数据库表结构，功能组件之间的关系结构。

## 系统网络拓扑结构设计

该系统基于C/S架构，系统主要功能运行在Android客户端APP上，服务器端主要包括用于接受用户反馈意见，更新APP版本，更新归属地查询的数据库和病毒查杀的数据库。系统的网路拓扑结构如图所示。



图 14 系统网络拓扑结构图

用户手机是APP的主要使用者，通过互联网可获取APP的更新和APP所需数据库的更新，若用户对APP功能有其他需求或反映系统Bug可通过评论功能提交意见。通过用户绑定的安全手机，可以发送短信命令用户手机执行安全操作。

## 系统功能模块总体设计

系统功能模块图，主要直观的说明系统具备的功能，该系统具备系统更新、手机防盗、手机杀毒、骚扰拦截、APP管理、来电查询、短信备份、流量统计、清理优化等就打模块，每个模块又包含子模块，系统整体模块结构如图所示。



图 15 系统整体功能模块结构图

## 系统功能模块详细设计

活动图是统一建模语言（UML）中用以描述对象活动的顺序关系所遵循的规则，侧重系统活动的行为顺序[17]。同时活动图是面向对象的，对于使用Java面向对象语言进行开发具有指导意义。本章是对模块进行具体的业务逻辑设计，分模块说明其功能的设计思路，采用活动图展示每个模块的具体业务逻辑。

### 欢迎界面

欢迎界面主要用于展现产品Logo、负责系统的初始化、联网检查系统的更新、联网更新系统内置的数据库。

欢迎界面的页面显示产品Logo和版权等相关信息。初始化功能主要在用户第一次进入系统进行系统数据库的初始化，将数据库文件复制到方便系统调用的data主目录下，若系统进行过更新或单独更新过数据库则再次初始化。自动更新功能通过联网检查服务器的JSON文本文件，解析出版本信息与系统版本信息对比，若有新版本则通过后台线程联网下载APK安装包，并弹出安装界面引导用户安装，若无版本更新或联网异常则进入系统主公你页面。欢迎界面的自动更新功能默认是启动的，用户可以到主功能界面设置是否自动更新。



图 16 欢迎界面活动图

### 手机防盗

手机防盗主要用于找回丢失的手机，提供一个设置向导引导用户将功能所需的基本信息录入系统，引导用户开启该模块的功能需要获取的系统权限，注册系统的监听服务来监听安全号码来信是否含有远程控制命令。

手机防盗由于功能较为强大，可远程控制手机锁屏甚至销毁手机全部数据，因此该功能需要用户设置密码防止他人随意串改。

手机防盗功能需要指导用户如何使用，在用户第一次使用该功能时要将说明详细写在开头，以便用户理解该功能的使用方法。

防盗功能主要是通过GPS定位来确定手机的大致位置，因为GPS定位不精确到几米的精度，因此需要结合手机报警声音识别特定的范围，同时要保证报警声音持续响起且防止被关闭，需要将手机屏幕锁屏并设置锁屏密码。如果因为各种各样的原因手机无法取回或者手机内信息的价值远远超过手机本身商品的价格，用户可以选择直接将手机的全部数据消除。手机丢失后如果被更换了SIM卡，失主无法得知手机新号码导致无法发送命令短信，因此需要设计一个手机启动检测SIM卡的功能，若发现SIM卡被更换则利用该SIM卡发送短信到绑定的安全号码，从而让失主得到手机新号码，可进行远程命令操作。经过以上分析，在设置向导中需要获得的信息有：本机SIM卡、安全号码、锁屏密码。

由于控制系统锁屏并设置锁屏密码是系统高级的权限，需要获取相应权限，因此需要提供一个方便的按钮让用户直接设置系统权限。最后，要给用户选择权是否真正开启手机防盗服务。

当用户选择启动手机防盗服务，将把监听安全号码短信的监听事件注册到Android手机底层的监听组件。由于监听组件的执行时间限制，需要将耗时的GPS定位注册成系统服务在后台获取GPS信息。

防盗功能活动图如下所示。



图 17 手机防盗活动图

当用户手机遗失或被窃，用户可用先前绑定的安全手机向用户手机发送指定的短信控制命令。此前用户已经开启了防盗服务，注册了监听事件，Android在接受到短信时会检查是否是安全号码，若是则进一步检查是否包含控制命令，若包含则执行该命令。

GPS定位命令：通过调用Android系统提供的定位功能获取经度、纬度信息并以短信回传给安全号码，由于GPS定位在室内效果极差，需要辅以网络定位，两者同时进行，将获取定位快的一方的信息返回。如果由于网路或GPS信号阻塞等原因获取不到GPS信息，则也返回获取位置中得知监听事件是否仍然有效获取命令；

报警声音命令：调用Android的媒体库循环播放报警声音，将手机音量调至最大并保持最大。

锁屏命令：调用Android的设备管理锁死屏幕，并更改解锁密码为之前设置的密码。

数据销毁：调用Android的设备管理直接回复出厂设置，彻底清除所有痕迹。

功能设计时充分考虑了隐秘性，手机防盗监听服务会在收到指令后立刻删除安全号码的短信命令，没有任何消息弹出，不留痕迹。而回传GPS地址的短信也不会在手机内保留。全过程悄然无息。



图 18 远程控制活动图

### 手机杀毒

手机杀毒主要功能是查杀手机安装的APP，通过系统内置的病毒数据库的特征值进行比对，扫描出病毒程序。若扫描有病毒，则调用APP卸载将病毒程序杀灭。

目前市面上主流的杀毒软件的核心原理都是相同的，即通过杀毒软件厂商对病毒的抓取和识别，确认并计算出病毒的特征值加入到杀毒厂商的数据库中，在扫描每个文件的同时计算文件的特征值并查询病毒特征值数据库比对是否和病毒的特征值相同，相同则为病毒。计算文件的特征值对于相同的文件必须唯一，这样才能准确的识别病毒。现在的主流的文档特征算法有MD5，SHA1算法。。

本功能的设计基于MD5的特征值算法，通过扫描系统内安装的每个APP，计算该APP的唯一MD5特征值，并查询系统内置的病毒数据库是否包含该特征值，判定该文件是否为病毒文件。病毒数据库的来源主要是抽取网上开源的病毒特征值数据，设计数据库的表结构并用SQL语句批量导入到SQLite数据库，详细数据库表结构参照下一章节的数据库表设计。



图 19 手机杀毒活动图

### 骚扰拦截

骚扰拦截功能主要有查看、添加、修改、删除需要拦截的电话号码，选择对应号码的拦截模式如电话拦截、短信拦截或全部拦截，将相应信息保存在数据库，并注册系统的监听事件监听来电号码，比对数据库是否需要相应的拦截。

骚扰拦截需要对拦截的电话号码进行管理，需要经常的进行增删改查的数据操作，因此使用数据库进行管理是较为理想的方案，方便的直接SQL语句进行数据库操作，减少通过代码直接操作的复杂程度。数据库表设计主要涉及对拦截的电话号码和拦截模式进行记录，详细数据库表结构参照下一章节的数据库表设计。

查询拦截号码主要用于在UI界面显示拦截电话号码和拦截清空；添加拦截号码通过弹出编辑对话框完成，用户编辑需要拦截的电话号码并选择相应的模式，当用户完成编辑操作并确定后，系统对数据库操作，插入新添加的电话号码和拦截模式，并通知UI界面更新。删除拦截号码通过点击删除按钮完成，同样会对数据库删除该行数据，并通知UI界面更新。改变拦截号码通过点击拦截号码的条目，弹出编辑对话框更改该行数据，并通知UI更新。

图 20 增删改查拦截号码活动图

开启拦截功能需要注册监听来电和短信的事件并开启拦截服务。核心的拦截功能的实现主要是对广播事件拦截。

短信拦截模式：当手机接收到短信时，Android平台会发送有序的通知广播，当拦截服务抓取到有序广播并查询数据库判定是否为需要拦截的电话号码，若不是则再次将广播传递出去，一切正常，若是需要拦截的对象，则将广播拦截，Android系统自带的短信应用则无法接受到短信，短信即被拦下。但是，由于不同版本的Android平台可能造成广播拦截的失效，因此除了广播拦截以外本功能还调用了Android系统自带短信应用的内容提供者，检测最新短信是否为拦截短信，是则删除。



图 21 短信拦截活动图

电话拦截：情况较为复杂，因为打电话是手机的根本核心，电话的广播是最高优先级，无法被拦截，因此需要调用Android系统自带的电话应用的挂断来电的功能。因为是直接调用其它的APP，属于跨进程调用，需要使用AIDL来调用电话应用的挂断功能从而达到拦截电话的功能。



图 22 电话拦截互动图

### APP管理

APP管理的功能主要包含查看手机剩余容量、获取每个APP信息和占用空间大小、启动APP和卸载APP。

查看手机剩余容量功能：是为了用户能较好掌握目前手机空间使用情况。Android底层是基于Linux的，Linux优秀的文件管理系统提供了对手机空间完善的记录文件，只需访问获取所需信息即可。

获取每个APP信息功能：是为了用户能了解手机安装APP的相关信息，APP的大部分信息可以从Android的包管理者中迭代抽取出来，并显示在可滑动上列表方便用户查看。而每个APP占用空间需要通过计算APP的安装路径的文件大小获得占用空间大小。

启动APP功能：通过包管理者求取的每个APP信息里包含了很多可用信息如作为APP在手机中唯一识别的标志——包名，通过包名作为索引可以向Android系统发送请求启动该包名代表的APP的指令。

卸载APP功能：同理，卸载APP也是通过前面获取的包名作为索引向Android系统发送请求启动该包名代表的APP的指令。



图 23 APP管理活动图

### 来电查询

来电查询功能主要包括自助查询电话归属地和来电自动查询电话归属地。

自助查询电话：将用户输入的电话号码作为索引查询内置的数据库进行比对，返回号码所属归属地。



图 24 来电自助查询活动图

来电自动查询：该功能默认是关闭的，若想使用需要用户自行开启来电自动查询服务。开启后会向手机注册监听来电的监听事件，平常状态不会启动服务消耗手机性能，但是监听到来电则启动自动查询服务并查询内置数据库，通过透明弹窗方式驻守在接听电话的界面提示来电归属地，电话接听后或拒听后弹窗自动消失，查询服务自动消亡。



图 25 来电自动查询活动图

### 短信备份

短信备份功能包括短信备份和短信还原。

短信备份：通过手机自带短信应用里的内容提供者将所有短信数据查询出来，将数据序列化成XML文件格式储存在SD卡中，用户可以将该XML文件拷贝保存。



图 26 短信备份活动图

短信还原：还原前会先清理目前手机存在的所有短信。再读取先前备份的XML文件，并进行数据反序列化，将数据还原成手机自带短信应用要求的数据格式，并通过其内容提供者插入，将短信还原。为了数据安全，本功能还设计保险措施，先检查是否存在备份文件，否则可能误删全部短信；在还原前弹出对话框让用户确认，防止用户错误触发还原。



图 27 短信还原互动图

### 清理优化

清理优化主要扫描手机安装的每个APP的缓存，并将缓存删除释放手机内存空间，缓存的减少有助于提高手机运行性能。

缓存垃圾是应用临时存储的资源，是硬盘资源，缓存不及时清理的话，就会占用越来越多的手机存储空间。通常缓存帮助使硬件运行的更快，是一种时间换空间的策略。当一个应用程序来读取数据时，会优先从缓存中找到所需要的数据。如果发现直接执行，找不到的话再从网路下载或者重新执行程序生成所需数据，但是由于用户安装越来越多的APP，每个APP都会生成缓存，积累造成大量缓存数据，会影响手机运行性能，因此需要对缓存进行清理。

扫描APP缓存：通过Android的包管理者获取每个安装的APP的信息，通过APP的唯一包名，计算该路径包含的文件大小，将获得的缓存大小进行排序，用户可以根据需求选择删除对应APP的缓存。

清理指定APP缓存：由于Android系统架构原则上是无法清理其他APP的缓存，只能由Android自带的应用包管理程序才能清除，所以清理单个APP的缓存需要去调用应用包管理程序，用户手动点击清除缓存。



图 28 清理指定缓存活动图

一键清理：一键清除是将所有APP的缓存均清除，早期手机很容易消耗完手机存储空间，因此Android系统架构设计了一个在手机存储空间严重不足的情况下清理所有缓存的机制，本一键清理功能就是调用了这个机制。



图 29 一键清理活动图

### 流量统计

流量统计功能主要帮助用户查看手机移动网络的总流量、全部的通信方式总流量以及每个APP消耗的流量。

移动网络流量和全部通信流量统计：Android底层的Linux系统具有完善的网路统计功能，每时每刻对网路通信的数据交换进行记录。若想统计某个流量，只需去查找相应的流量记录。对于手机用户关心的主要是移动网络流量，全部流量仅作为参考提供给用户。

单APP流量统计：通过Android的包管理者抽取出APP在内核注册的唯一id信息，通过id去查找对应APP的流量大小，并将查出的流量做一个排序，方便用户查看和掌握APP的流量消耗情况。



图 30 流量统计活动图

## 数据库表结构设计

数据库是一个系统中重要的组成部分。Android系统已经内置SQLite数据库，无需考虑数据库系统的选择，因此本章节主要描述数据库表结构的设计。通过对系统功能需求和业务逻辑的分析抽取出相应的实体、实习属性和实体关系。

本系统涉及数据库增删改查的功能是手机杀毒、来电查询和骚扰拦截。主要涉及三张个数据库。病毒数据库包含病毒特征值表、来电查询数据库包含电话号码表和归属地表、拦截数据库包含骚扰拦截表。

### 病毒数据库

主要用于查询扫描的文件的特征值是否存在。

病毒特征值表：病毒特征值是通过MD5算法计算出的每个病毒文件唯一的特征值，符合作为主键的约束，因此设置为主键。病毒名称表明了一个病毒的类型和行为模式，因此设置名称熟悉。病毒需要相应的描述信息，因此设置描述信息属性。

表 11 病毒特征值表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 约束条件 | 说明 |
| md5 | INTEGER |  | 主键 | 主键 |
| name | VARCHAR | 45 | 非空 | 病毒名称 |
| desc | VARCHAR | 100 |  | 病毒描述 |

创建表的SQL命令：

CREATE TABLE virus(md5 INTEGER PRIMARY KEY, name VARCHAR(45) NOT NULL, desc VARCHAR(100));

### 来电查询数据库

电话归属地的判断主要由手机前7位和座机的区号识别，而归属地千差万别，两种数据的目的不同，需要分开两张表。前者记录不同号段的电话号码，后者记录归属地，两者的关系通过外键连接，不会造成数据冗余。

电话号段表：每个号段都是唯一的，因此作为主键。而归属地通过数字作为代理外键。

表 12 电话号段表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 约束条件 | 说明 |
| phone | INTEGER |  | 主键 | 电话号段 |
| addressid | INTEGER |  | 外键，非空 | 归属地代理键 |

创建表的SQL命令：

CREATE TABLE phone(phone INTEGER PRIMARY KEY, addressid INTEGER NOT NULL, FOREIGN KEY(addressid) REFERENCES address(addressid));

归属地表：虽然归属地也是唯一，但由于是字符串不适合作为外键，因此使用数字编号作为代理主键，同时该属性被电话号码表的代理外键引用。归属地名称则用来说明归属地。

表 13 归属地表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 约束条件 | 说明 |
| addressid | INTEGER |  | 主键 | 代理主键 |
| address | VARCHAR | 50 | 非空 | 归属地 |

创建表的SQL命令：

CREATE TABLE address(addressid INTEGER PRIMARY KEY AUTOINCREMENT, address VARCHAR(50) NOT NULL);

### 骚扰拦截数据库

骚扰拦截记录拦截的电话号码和拦截模式（电话或短信或全部拦截）

拦截电话表：拦截电话则必须存储电话，虽然拦截的电话是唯一的，但由于用户需要经常增删改查拦截电话，若将电话作为主键，主键是无法复用的，所以应该设置代理主键。一个电话拦截的要么是电话要么是短信，所以需要选择需要拦截的模式。

表 14 拦截电话表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 约束条件 | 说明 |
| \_id | INTEGER |  | 主键 | 代理主键 |
| phone | VARCHAR | 20 | 非空 | 拦截电话 |
| mode | VARCHAR | 2 | 非空 | 拦截模式 |

创建表的SQL命令：

CREATE TABLE blockcall(\_id INTEGER PRIMARY KEY AUTOINCREMENT, phone VARCHAR(20) NOT NULL, mode VARCHAR(2) NOT NULL);

## 本章小结

本章节主要介绍管家APP的系统功能设计。首先说明了系统采用的C/S网路拓扑结构。然后通过UML的活动图，对根据需求分析所划分的功能模块进行具体的业务逻辑设计。最后设计了系统功能模块所需使用到的数据库表结构。

# 系统实现

## 系统开发平台

系统的实现和用户交互主要在Android平台，开发工作主要面向Android平台，服务器平台仅保存JSON格式的系统更新信息，较少开发工作。系统使用轻量级应用服务器Tomcat8搭建，开发工具使用谷歌提供的官方开发者工具包，使用Java语言开发系统各个功能模块的业务逻辑，数据库采用Android内置的SQLite3，详细开发平台说明如表所示。

表 15 开发平台说明表

|  |  |
| --- | --- |
| 开发项目 | 说明 |
| 开发语言 | Java语言 |
| 开发工具包 | JDK 1.8.40，Android SDK 21 |
| 集成开发工具（IDE） | Android Studio 1.1.0，Intellij Idea 14 |
| 操作系统 | Windows 8.1，Ubuntu 14.04.2 |
| 数据库 | SQLite3 |
| 应用服务器 | Tomcat8 |

## 系统功能实现

本章描述的是系统各个功能模块的具体实现方法。功能的实现遵循MVC编程模式的软件设计，显示界面，业务逻辑和数据分离到各自的去耦元件，实现良好的解耦，提高软件的可重用性和可维护性，易于修改，随时根据需要。以下分功能模块详细叙述其功能实现的业务逻辑和关键实现代码。

### 欢迎界面

欢迎界面的具体实现分三个部分：更新数据库，初始化系统，更新系统

更新数据库：首先通过Http协议连接网络服务器，读取网路服务器的JSON格式的说明文件，解析JSON得知数据库版本是否有更新，如果更新下载并放置在应用程序安装后的路径。

初始化系统：主要是初始化后面其他模块需要的数据库。由于要执行对数据库的查询需要连接数据库。数据库最初是打包在APK安装包的Asset原生目录下的，安装后并不会主动释放到文件安装目录，因此需要主动进行复制。

更新系统：与更新数据库同理，通过读取网路服务器的JSON说明文件，解析并得到版本信息，若有变化则根据JSON文件提供的下载地址，获取文件流并下载下来，最后调用手机的安装管理应用程序进行更新安装。

相关关键代码：

连接网络服务器：

HttpURLConnection conn = (HttpURLConnection) apkUrl.openConnection();

conn.setRequestMethod("GET");

int responseCode = conn.getResponseCode();

解析JSON文件：

JSONObject json = new JSONObject(jsonStr);

String version = (String) json.get("version");

String apkurl = (String) json.get("apkurl");

调用安装APK：

Intent intentInstall = new Intent();

intentInstall.setAction("android.intent.action.VIEW");

intentInstall.setDataAndType( uri, "APPlication/vnd.android.package-archive");

startActivity(intent);

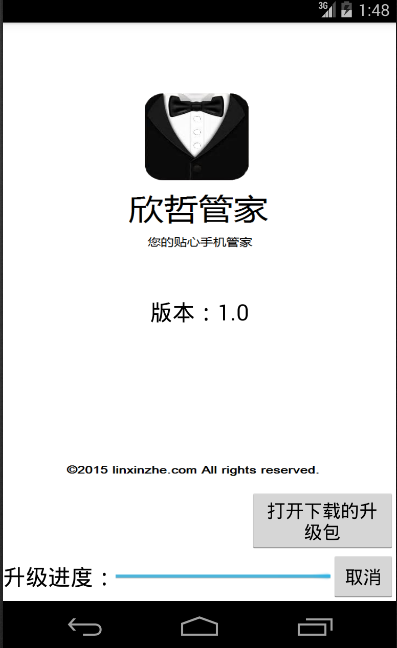


图 31 欢迎界面截图

### 手机防盗

手机防盗的具体实现分为三个部分，登陆密码、设置向导收集基本信息和防盗状态、安全号码短信命令监听。

登陆密码：在用户第一次使用时需要设置密码。功能实现通过弹出对话框的形式。只有当用户输入密码且两次密码一致的情况下，密码经过MD5单向加密保存在SharedPreferences中。当用户以后再次使用手机防盗功能时需要输入密码，并通过MD5计算后和存储的密码进行比对，比对一致才能进入手机防盗功能。



图 32 手机防盗截图

设置向导和防盗状态：通过几个Activity界面收集防盗功能所需基本信息，设置向导之间的交互使用手势识别，向左滑动翻页。首先是阅读如何使用功能的信息，主要放置说明的文本。接着绑定本机的SIM卡，通过TelephonyManager获取手机的Sim卡序列号；输入安全号码手机号或者直接从手机联系人中选择。最后，设置锁屏密码并选择是否开启防盗保护，以及选择授予该功能权限。SIM卡序列号、安全号码和锁屏密码均保存在SharedPreferences。设置完毕后，用户再次进入则查看目前设置好的防盗状态。防盗状态主要读取SharedPreferences并显示参数信息。

相关关键代码如下：

获取Sim卡序列号：

TelephonyManager tm = (TelephonyManager) getSystemService(TELEPHONY\_SERVICE);

String sim = tm.getSimSerialNumber();

保存相关的设置信息：

SharedPreferences sp = getSharedPreferences("config", MODE\_PRIVATE);

SharedPreferences.Editor editor = sp.edit();

editor.putString("xxx", xxx);//保存信息

editor.commit();

授予权限代码：

Intent intentPermis = new Intent(DevicePolicyManager.ACTION\_ADD\_DEVICE\_ADMIN);

ComponentName mDeviceAdminSample = new ComponentName(Setup3LostFoundActivity.this, MyReceiver.class);

intentPermis.putExtra(DevicePolicyManager.EXTRA\_DEVICE\_ADMIN, mDeviceAdminSample);

intentPermis.putExtra(DevicePolicyManager.EXTRA\_ADD\_EXPLANATION, "开启后可以远程锁屏和远程销毁数据");

startActivity(intentPermis);

读取相关的配置信息：sp.getString("xxx", "")//读取xxx，若不存在则为空

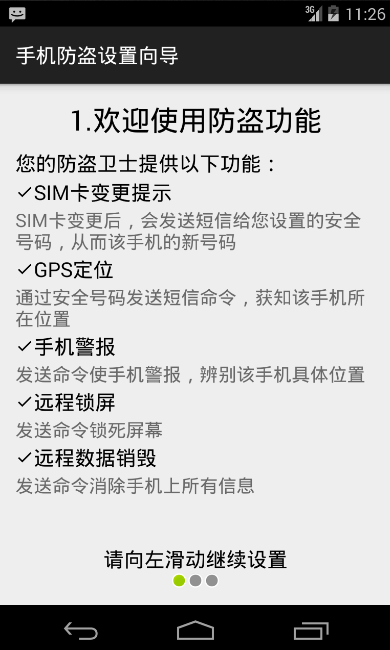


图 33 手机防盗截图1

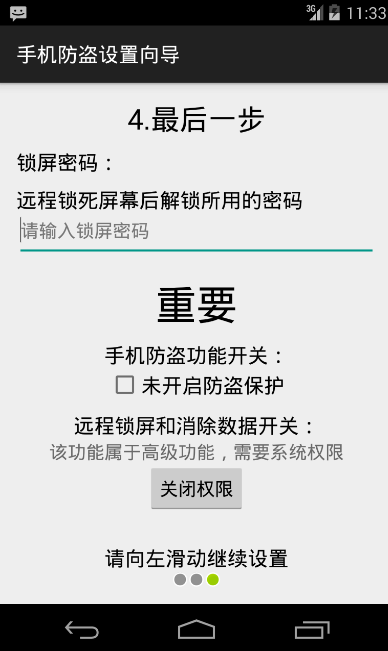


图 34 手机防盗截图2

安全号码短信命令监听：若开启了防盗保护选项则向系统注册监听短信的广播接受者。当系统获得短信后，判断是否是安全号码再判断是否是命令以及是哪种命令。如果是GPS定位命令，则开启GPS定位服务，调用手机定位并返回经度、纬度信息以短信形式回发给安全号码，为了获取GPS的可靠性，网路定位和GPS定位两个线程同时进行，并返回快的一方的信息。如果是播放报警声音命令，则启动手机的媒体播放功能并音量为最大和循环播放。如果是锁屏命令，则调用手机的设备管理应用，锁死屏幕并设置解锁密码为预先设置的密码。如果是数据销毁命令，则调用手机的设备管理应用，清除所有数据并恢复出厂设置。为了保证隐秘性，所有的命令短信在收到后即刻被删除。

相关关键代码：

注册监听短信的广播接收者：

<receiver android:name=".receiver.SMSReceiver">

<intent-filter android:priority="1000" >

<action android:name="android.provider.Telephony.SMS\_RECEIVED" />

</intent-filter>

</receiver>

监控信息和获取信息的发送者和内容：

Object[] objs = (Object[]) intent.getExtras().get("pdus");

for (Object o : objs) {

SmsMessage sms = SmsMessage.createFromPdu((byte[]) o);

String sender = sms.getOriginatingAddress();

String body = sms.getMessageBody();

…

}

网路定位：

LocationManager lmNET= (LocationManager) getSystemService(Context.LOCATION\_SERVICE);

listenerNET = new MyNETLocationListener();

lmNET.requestLocationUpdates(LocationManager.NETWORK\_PROVIDER, 60 \* 1000, 50, listenerNET);

…

private class MyNETLocationListener implements LocationListener {

@Override

public void onLocationChanged(Location location) {

String longitude =location.getLongitude() + "\n";//经度

String latitude =location.getLatitude() + "\n";//纬度

String accuracy = location.getAccuracy() + "\n";//精度

…

}

GPS定位：

lmGPS.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 60 \* 1000, 50, listenerGPS);

播放报警音乐：

MediaPlayer soundPlayer = MediaPlayer.create(context, R.raw.alert);

soundPlayer.setLooping(true);

soundPlayer.setVolume(1.0, 1.0);

player.start();

锁死屏幕：

DevicePolicyManager dpm =

(DevicePolicyManager) context.getSystemService(Context.DEVICE\_POLICY\_SERVICE);

ComponentName who = new ComponentName(context, MyAdminReceiver.class);

if (dpm.isAdminActive(who)) {

dpm.lockNow();

dpm.resetPassword(sp.getString("lockpsw", "9999"), 0);

}

数据销毁：

devicePolicyManager.wipeData(DevicePolicyManager.WIPE\_EXTERNAL\_STORAGE);

devicePolicyManager.wipeData(0);

删除命令短信：

long id = getThreadId(context);

Uri smsUri = Uri.parse("content://sms/conversations/" + id);

context.getContentResolver().delete(smsUri, null, null);

### 手机杀毒

手机杀毒的具体实现分为三个部分：扫描APP，查询病毒数据库，杀毒。

扫描APP：由于扫描APP是一个运行时间较长的动作，不能直接在主线程操作，否则会阻塞主线程，造成应用程序无响应（ANR）异常。因此扫描的操作必须开启新线程，并将扫描结果实时显示在界面上。因为每个应用程序在手机内System目录下都保存着每个应用的安装包，扫描的原理主要通过包管理者获取每个安装的应用程序信息，获取每个应用程序的安装包所在目录，并计算安装包的md5特征值。获得的MD5特征值将作为搜索病毒数据库的索引。若查到的不是病毒则用绿色字体的Textview显示，是病毒则用

查询病毒数据库：要实现查询数据库则首先建立数据库的连接，Android对于SQLite3数据库的连接通过帮助类SQLiteOpenHelper实现，实现该帮助类，并在每次查询数据库时通过它获取数据库的连接，再执行SQL语句，查询MD5特征值是否在数据库里。

杀毒：实现杀毒主要提取包名作为索引，去调用Android的安装管理应用程序卸载病毒APP。

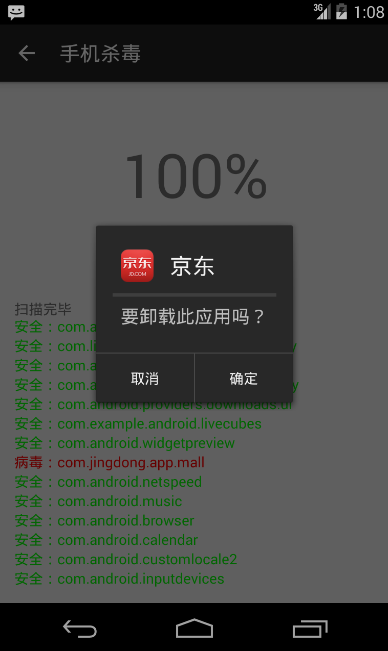


图 35 手机杀毒截图

相关关键代码：

扫描APP：

List<PackageInfo> packageInfos = pm.getInstalledPackages(PackageManager.COMPONENT\_ENABLED\_STATE\_DEFAULT);

for (PackageInfo info : packageInfos) {

File file = new File(info.APPlicationInfo.sourceDir);

String md5File = MD5Tools.encrypt(file);

if (Md5AntivirusDao.isVirus(AntivirusActivity.this, md5File)) {

…

}

…

}

查询数据库：

boolean isVirusExist = false;

File file = new File(context.getFilesDir(), "antivirus.db");

SQLiteDatabase db = SQLiteDatabase.openDatabase(file.toString(), null, SQLiteDatabase.OPEN\_READONLY);

cursor = db.rawQuery("SELECT \* FROM datable WHERE md5=?", new String[]{md5});

if (cursor.moveToNext()) {

isVirusExist = true;

}

杀毒：

for (APPInfo virus : viruses) {

Intent intentKill = new Intent();

intentKill.setAction("android.intent.action.VIEW");

intentKill.setAction("android.intent.action.DELETE");

intentKill.addCategory("android.intent.category.DEFAULT");

intentKill.setData(Uri.parse("package:" + virus.APPPackageName));

startActivityForResult(intentKill,RESULT\_OK);

}

### 骚扰拦截

骚扰拦截的具体实现分为：增删改查数据库，来电监听拦截。

增删改查数据库：通过实现SQLiteHelper帮助类，并在用户编辑拦截信息后对数据库执行相应的SQL语句进行增删改查。

来电监听拦截：在用户开启拦截后，要向手机注册监听来电和短信的事件，并将该监听者的优先级调至最高。在接受到短信后手机会发送一个收到短信的广播，此时监听者启动拦截服务，检查短信发送者是否存在数据库内，且是否符合拦截要求，若符合则将该广播直接拦截，广播无法向下传递则手机自带短信应用程序无法接受到短信，拦截成功。若接到的是电话，则检查来电号码和拦截要求是否符合，若符合则跨进程去调用手机自带通话应用程序的挂断电话方法，实现电话拦截。具体实现是通过反射获取到” android.os.ServiceManager”类。通过它提供的getService方法去获取通话服务的IBInder对象。IBinder对象才可以调用通话应用程序的挂断电话方法。

相关关键代码：

拦截短信：

abortBroadcast();

反射获取IBinder对象挂断电话：

Class smClass = BlockCallService.class.getClassLoader().loadClass("android.os.ServiceManager");

Method method = smClass.getDeclaredMethod("getService", String.class);

IBinder ibinder = (IBinder) method.invoke(null, TELEPHONY\_SERVICE);

ITelephony.Stub.asInterface(ibinder).endCall();



图 36 骚扰拦截截图

### APP管理

APP管理的实现分为：计算手机剩余空间，扫描APP信息并计算APP大小，启动APP，卸载APP

计算剩余空间：Android文件管理系统和Linux是相同的。利用StatFs可得到文件系统的相关信息，通过可用文件块和文件块大小计算出手机剩余空间。

扫描APP信息：与前面的杀毒同理，由于可能阻塞主线程因此开新线程获取每个安装的APP信息。在尚未全部读取完毕前，显示ProgressBar告知用户仍在读取信息中。读取完毕后将得到的信息显示在布局控件ListView上，并按大小排序。

启动APP：通过将获得的包名作为索引让Android的包管理者去打开相应APP。

卸载APP：通过获取的包名调用Android的安装管理应用程序来卸载APP。

相关关键代码：

计算剩余空间：

StatFs statFs = new StatFs(Environment.getDataDirectory().getAbsolutePath());

int localMemory = statFs.getAvailableBlocks() \* statFs.getBlockSize();

启动APP：

PackageManager pm = getPackageManager();

Intent intent = pm.getLaunchIntentForPackage(packname);

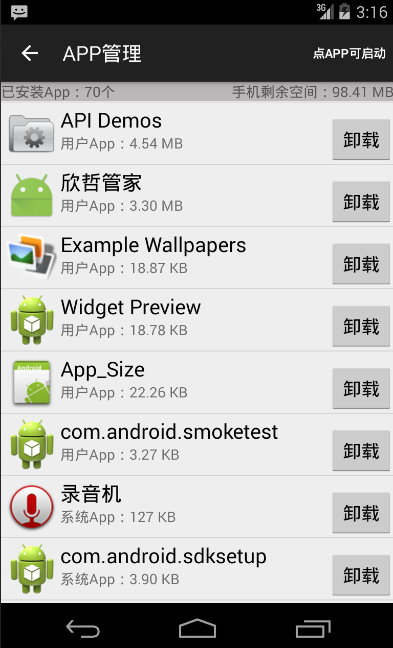


图 37 APP管理截图

### 来电查询

来电查询的实现分为：自助查询号码，来电自动查询

自助查询号码：通过实现SQLiteHelper帮助类，并在用户自助输入所需查询的来电号码后对归属地数据库执行相应的SQL查询语句。

来电自动查询：原理与骚扰拦截类似，通过向手机注册监听来电的事件，当手机检测到来电，则查询归属地数据库，并弹出浮窗告知用户来电属地，当来电被接听或拒听则系统会发送来电状态改变的广播事件，监听者收到后将浮窗销毁。

相关关键代码：

来电自动查询弹窗：

WindowManager wm = (WindowManager) getSystemService(WINDOW\_SERVICE);

WindowManager.LayoutParams params = new WindowManager.LayoutParams();

…

wm.addView(view, params);//添加窗体



图 38 来电查询截图

### 短信备份

短信备份的实现主要为：短信数据序列化和数据反序列化。

数据序列化：首先要通过内容提供者获取所有的短信数据，通过将短信数据打包成SmsInfo的JavaBean，再通过XML序列化器将短信数据序列化成按照一定标签格式排列的XML写入用户的SD卡，作为备份。

数据反序列化：主要是读取之前备份的XML文件，将XML用Pull解析方式反序列化，将读取到的标签对应写入SmsInfo的JavaBean属性，将打包好的SmsInfo再通过内容提供者插入回手机的短信应用，从而还原短信。

相关关键代码：

获取所有短信数据：

Uri uri = Uri.parse("content://sms/");

Cursor cursor = resolver.query(uri, new String[]{"body", "address",type", "date"}, null, null, null);

短信数据序列化：

XmlSerializer xmlSerializer = Xml.newSerializer();

…

xmlSerializer.startTag(null, "sms");

xmlSerializer.startTag(null, "body");

xmlSerializer.text(body);

xmlSerializer.endTag(null, "body");

…

xmlSerializer.endTag(null, "sms");

数据反序列化：

XmlPullParser parser = Xml.newPullParser();

int eventType = parser.getEventType();

while (eventType != XmlPullParser.END\_DOCUMENT) {

switch (eventType) {

case XmlPullParser.START\_TAG:

switch (parser.getName()) {

case "body":

smsContent.put("body", parser.nextText());

…

}

eventType = parser.next();

}



图 39 短信备份截图

### 清理优化

清理优化的具体实现分为：扫描每个APP信息，根据

扫描APP缓存：与APP扫描原理相同，同时由于扫描也会阻塞主线程，因此扫描在新线程中进行。扫描后可获得APP的唯一包名，通过他们获取缓存文件路径，计算该路径包含的文件大小，将获得的缓存大小进行排序，用户可以根据需求选择删除对应APP的缓存。

清理指定APP缓存：由于Android系统架构原则上是无法清理其他APP的缓存，只能由Android自带的应用包管理程序才能清除，所以清理单个APP的缓存需要去调用应用包管理程序，用户手动点击清除缓存。

一键清理：一键清除是将所有APP的缓存均清除，Android系统架构设计了一个在手机存储空间严重不足的情况下清理所有缓存的机制，本一键清理功能就是调用了这个机制，由于是跨进程调用，因此使用了AIDL接口作为获取方法的中介，使用了系统源码中的IPackageDataObserver.aidl，通过继承它的Stub重写onRemoveCompleted方法，同时因为谷歌隐藏了该方法，因此需要用反射的方法调用该方法。

相关关键代码：

获取缓存大小：

File cacheFIle=new File(‘/data/data/’+packageName+”/cache”);

long cacheSize=cacheFIle.Length();

清理指定APP缓存：

intentClean.setAction(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);

Uri uri = Uri.fromParts("package", packName, null);

intentClean.setData(uri);

startActivity(intent);

一键清理：

method = PackageManager.class.getMethod("freeStorageAndNotify", long.class, IPackageDataObserver.class);

method.invoke(pm, Integer.MAX\_VALUE, new IPackageDataObserver.Stub() {

@Override

public void onRemoveCompleted(String packName, boolean succeeded) throws RemoteException {}

}

…

});



图 40 清理优化截图

### 流量统计

流量统计的实现主要分为：获取每个APP信息，根据APP信息查询流量。

扫描APP信息：与APP管理的扫描原理相同，通过Android的包管理者抽取出APP在内核注册的唯一id信息。

获取流量：通过id去Android底层的Linux系统的网络统计记录文件中查找对应APP的流量大小，并将查出的流量做一个排序，方便用户查看和掌握APP的流量消耗情况。

相关关键代码：

获取移动数据流量：

long traffic2g = TrafficStats.getMobileRxBytes() + TrafficStats.getMobileTxBytes();//移动网络流量

long trafficTotal = TrafficStats.getTotalRxBytes() + TrafficStats.getTotalTxBytes();//总流量

获取单个APP流量数据：

int uid = APPlicationInfo.uid;

long rxBytes = TrafficStats.getUidRxBytes(uid);//单APP下载流量

long txBytes = TrafficStats.getUidTxBytes(uid);//单APP上传流量



图 41 流量统计截图

## 本章小结

本章主要介绍了管家APP的每个功能模块的具体实现方法。首先介绍开发平台的相关要求，然后分功能模块说明其具体实现的方法和涉及的相关关键代码。

# 系统测试

软件测试是一种通过使用测试工具，按照测试方案和流程对产品进行功能和性能测试的方法，软件测试的目的是为了有效的控制软件的质量。对于本系统的软件测试从两个方面着手进行。一是分模块测试，包含对于该功能模块的功能实现和交互逻辑正确性的检验测试。二是综合测试，以系统为整体单位进行测试，检验稳定性和性能。

## 测试环境

测试环境主要说明测试工具和运行测试的硬件平台以及系统运行的测试平台。

表 16 测试平台软件环境

|  |  |
| --- | --- |
| 参数 | 型号 |
| 操作系统 | Windows8.1, NT 6.3 |
| 测试工具 | Android Studio1.1.0，Android Device Monitor，Intellij Idea 14 |
| 数据库测试 | SQLite Expert Personal |
| 服务器 | Tomcat8 |
| 操作系统 | Windows8.1, NT 6.3 |

表 17 测试平台PC和服务器

|  |  |
| --- | --- |
| 参数 | 型号 |
| CPU | Intel Xeon E3-1230V3 4×3.30GHz |
| GPU | AMD Radeon HD 6790 |
| 内存 | 8G |
| 硬盘 | Samsung SSD 840 PRO 120GB |
| 操作系统 | Windows8.1, NT 6.3 |

表 18 测试平台Android虚拟机（虚拟Nexus4）

|  |  |
| --- | --- |
| 参数 | 型号 |
| CPU | Intel x86 Atom |
| GPU | PC的GPU |
| 分辨率 | 屏宽4.7英寸；1280×768 |
| 内存 | RAM容量：1GB |
| 操作系统 | Android4.4.2 |

表 19 测试平台真机红米1S

|  |  |
| --- | --- |
| 参数 | 型号 |
| CPU | 高通MSM8628四核1.6GHz |
| GPU | Adreno 305 |
| 分辨率 | 4.7英寸，1280×720 |
| 内存 | RAM容量：1GB |
| 操作系统 | MIUI V5系统，支持Android4.3 |

## 测试内容

### 功能测试

功能测试的目的是检查功能的实现情况，是否存在业务流程错误、交互逻辑错误、功能缺失。下面测试将分功能模块进行，对关键业务进行设计测试用例并测试。

表 20 功能测试目标

|  |  |
| --- | --- |
| 测试目的 | 确认功能模块是否正常有效 |
| 测试范围 | 系统全部功能模块 |
| 测试方法 | 1. 使用有效数据获得预期结果。 2. 使用无效数据获得错误提示 3. 业务逻辑流程的正确运用 |
| 测试标准 | 数据输入和输出预期一致 |

表 21 欢迎界面测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 网路连接断开是否会阻塞系统 | 手机断开网络连接 | 连接超时自动进入主页面 | 符合预期 |
| 2 | 下载更新流程 | 下载数据库并更新 | 数据库更新 | 符合预期 |
| 3 | 下载更新流程 | 下载安装包并更新 | 系统版本更新 | 符合预期 |
| 4 | 用户取消更新 | 取消更新 | 进入主界面 | 符合预期 |

表 22 手机防盗测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 设置初次登陆密码 | 密码不一致 | 无法进入系统 | 符合预期 |
| 2 | 再次登陆系统 | 错误密码 | 无法进入系统 | 符合预期 |
| 3 | 设置向导信息录入 | 录入信息 | 防盗状态页面正确显示录入的信息 | 符合预期 |
| 4 | 获取系统权限 | 点击获取权限 | 权限管理应用程序中包含本APP | 符合预期 |
| 5 | 手机GPS定位 | 发送定位短信命令 | 收到经度、纬度短信 | 真机不依靠网络定位的GPS定位存在超时现象 |
| 6 | 播放报警声音 | 发送定位短信 | 手机播放报警声音 | 符合预期 |
| 7 | 锁死屏幕 | 发送锁屏短信 | 手机锁屏并设置了密码 | 符合预期 |
| 8 | 数据销毁 | 发送销毁短信 | 虚拟机重置，数据清空 | 符合预期 |

表 23 手机杀毒测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 在病毒数据库中插入正常APP的特征值，验证是否查出病毒 | 执行杀毒 | 病毒提示 | 符合预期 |
| 2 | 杀毒 | 点击杀毒 | 病毒被卸载 | 符合预期 |

表 24 骚扰拦截测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 增删改查拦截电话 | 增删改查拦截电话 | 界面更新 | 符合预期 |
| 2 | 拦截短信 | 被拦截电话发送短信 | 无短信提示 | 符合预期 |
| 3 | 拦截电话 | 被拦截电话拨打电话 | 无来电提示 | 符合预期 |

表 25 APP管理测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 启动APP | 点击APP | 启动APP | 符合预期 |
| 2 | 卸载APP | 点击卸载 | 卸载APP | 符合预期 |
| 3 | 手机剩余空间是否正确 | 查看剩余空间 | 与系统剩余空间一致 | 符合预期 |

表 26 来电查询测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 查询电话号码 | 输入电话号码 | 获得归属地 | 符合预期 |
| 2 | 来电自动查询 | 拨打电话 | 弹出归属地弹窗 | 符合预期 |

表 27 短信备份测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 备份短信到XML | 备份短信 | 输出备份XML | 符合预期 |
| 2 | 从XML还原短信 | 还原短信 | 短信还原 | 符合预期 |

表 28 清理优化测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 扫描缓存 | 扫描APP | 缓存与手机应用管理程序数值一致 | 符合预期 |
| 2 | 清除缓存 | 点击清除 | 缓存清除 | 虚拟机符合预期。真机MIUI系统无效 |
| 3 | 一键清除 | 点击一键清除 | 清除所有缓存 | 虚拟机符合预期。真机MIUI系统无效 |

表 29 流量查询测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例序号 | 测试内容 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 移动网络流量 | 扫描 | 移动网络流量与运营商查询一致 | 符合预期 |

### 综合测试

综合测试以系统为整体单位进行各方面指标的测是，主要使用自动化测试的方法检查系统是否能稳定、可靠的运行。具体的自动化测试使用Testin云测平台进行。测试内容包括：

1. 安装卸载测试：测试APP在指定终端上是否可正常安装、正常卸载，给出不通过的错误原因。
2. 运行稳定性测试：主要测试APP的运行稳定性。
3. 性能测试：测试APP运行时的性能数据。
4. UI适配测试：测试APP的显示界面与目标的屏幕是否适配。

本次测试共选择了103款主流Android智能手机机型，覆盖三千万用户。

### 测试结果

#### 兼容性稳定性测试：

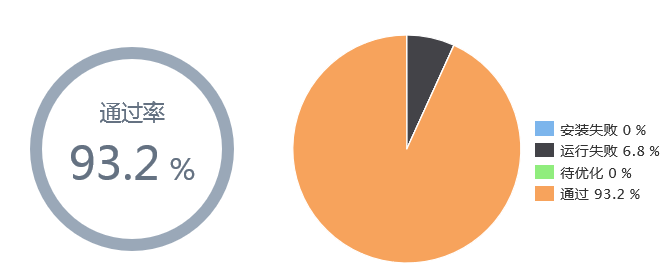


图 42 兼容性测试通过率



图 43 兼容性测试详情



图 44 兼容性测试结果

分析：系统在各项测试指标中均通过，安装、卸载、运行、稳定性和屏幕适配情况均较好，具有较高的通过率。说明本系统的软件质量较高、系统稳定性和可靠性较强，能在绝大多数主流机型上成功运行。



图 45 失败原因统计图

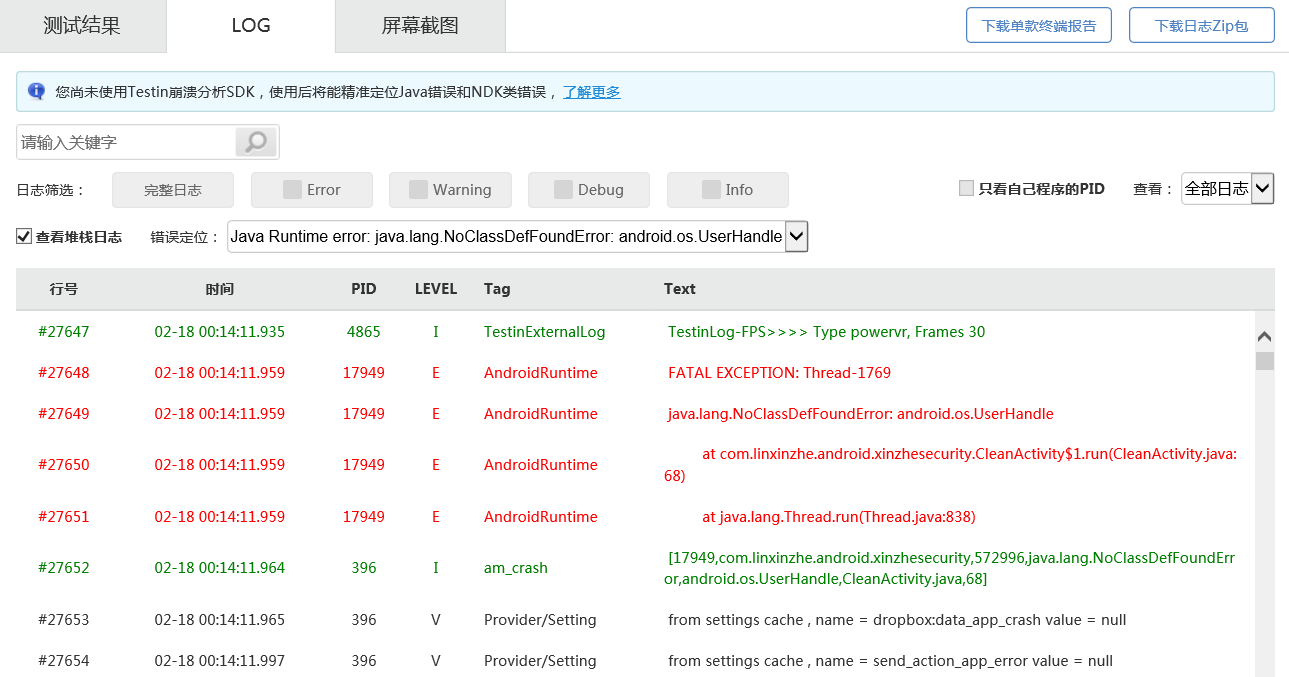


图 46 失败原因详情



图 47 失败原因定位

失败原因分析：所有失败终端的原因均是在运行测试中发现异常。原因为：Java Runtime error: java.lang.NoClassDefFoundError: android.os.UserHandle。主要出现在4.1.2和4.0.3系统上，各个品牌和屏幕尺寸均有，说明主要是在特定Android系统出现的问题。失败原因为找不到android.os.UserHandle类的定义，具体代码是在清理优化功能下的Method myUserId = UserHandle.class.getDeclaredMethod("myUserId");。说明对于这些特定机型的Android系统由于厂商对Android系统的修改导致android.os.UserHandle的缺失，无法正常使用清理优化功能。

#### 性能测试（96款成功运行的设备平均值）

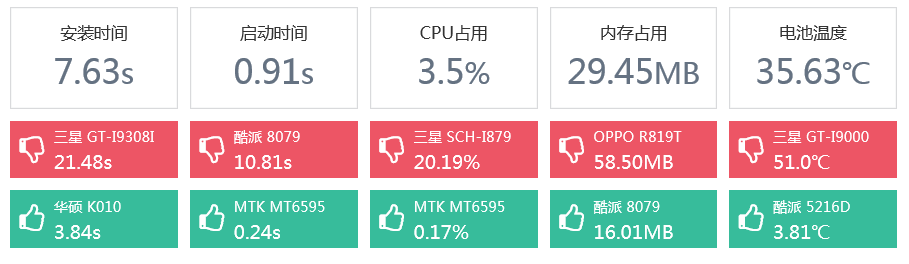


图 48 性能测试结果

分析：从总体来看，本系统的运行性能较好。从CPU占用和内存占用上看，本系统对手机性能的消耗较低。启动时间较快，软件响应性能较好。

## 本章小结

本章节主要介绍管家APP系统相关的测试内容和测试结果。对于每个模块的测试以功能测试为主，测试业务逻辑实现情况，测试结果良好。对于系统整体使用了第三方自动化测试平台，具备较高的通过率，说明APP的软件质量较高。