**基于Android系统的文件管理器**

**概要设计**

**（v1.0）**

**IT FARMERS**

2016年4月

目录

目录

[1 引言 3](#_Toc448079071)

[**1.1 目的** 3](#_Toc448079072)

[**1.2 背景** 3](#_Toc448079073)

[**1.3 范围** 3](#_Toc448079074)

[**1.4系统概述** 4](#_Toc448079075)

[2系统环境 4](#_Toc448079076)

[**2.1运行环境** 4](#_Toc448079077)

[**2.2开发环境** 4](#_Toc448079078)

[3 系统总体设计 5](#_Toc448079079)

[**3.1 系统架构层次设计** 5](#_Toc448079080)

[**3.2 文件管理器的功能** 6](#_Toc448079083)

[4系统模块设计 7](#_Toc448079084)

[**4.1 文件管理器界面的设计** 7](#_Toc448079085)

[**4.2** 功能模块设计 8](#_Toc448079086)

[**4.3** 数据流模型 9](#_Toc448079086)

[5 文件管理器基本功能的实现 12](#_Toc448079087)

[**5.1 文件管理器主界面活动的实现** 12](#_Toc448079088)

[**5.2 文件操作功能的实现** 13](#_Toc448079089)

[**5.3 创建文件/文件夹功能的实现** 14](#_Toc448079090)

[**5.4 文件分享功能的实现** 14](#_Toc448079091)

[6 接口设计 15](#_Toc448079092)

[**6.1 界面接口** 15](#_Toc448079093)

[**6.2 文件操作接口** 15](#_Toc448079094)

# 1 引言

## 1.1 目的

此文档是从软件系统即程序系统设计的总体考虑，包括系统的基本处理流程、系统分析、系统体系结构设计、界面设计、运行、出错处理和系统维护等方面的设计，为详细设计提供基础。

## 1.2 背景

本文档基于需求分析而进行系统的概要设计。对需求分析解析为系统的架构，组件，接口和其他特性。

本文档面向的读者主要是:

* 开发人员
* 设计人员
* 项目管理人员
* 测试人员

## 1.3 范围

本文档定义了与文件管理器系统体系结构、界面设计、文件管理功能、接口设计具体实现方法。

## 1.4系统概述

项目名称：基于Android的文件管理器

本项目从实际需求出发，设计并实现Android手机文件管理器，能够完成文件系统浏览、文件操作管理、文件共享等功能，可以有效的帮助Android手机用户处理日常文件和管理存储在本地手机上的文件操作。文件管理器由位于系统应用层的基本功能部分和位于内核层的安全管理部分组成。基本功能部分负责响应用户界面的指令并控制后台文件系统的访问，采用MVC模式进行开发。

# 2系统环境

## 2.1运行环境

本系统的硬件环境如下：

* 支持Android系统的智能手机
  + 内存：1G以上

## 2.2开发环境

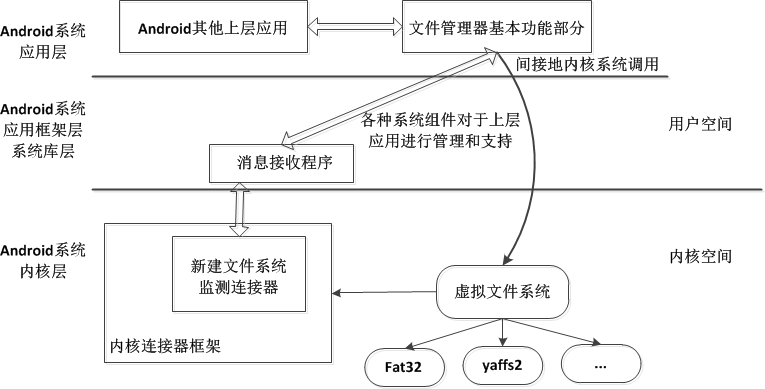
本系统的的开发环境如下：

* 操作系统：64bit的win7或以上版本
* 开发工具包：JDK Version jdk1.7.0\_72\Android SDK 4.0以上
* 开发环境：eclipse，Android Developer Tools v22.0.5-757759

# 3系统总体设计

## 3.1系统架构层次介绍

Android的系统架构主要分为4层，从高到低分别是：(1)由运行在虚拟机上的应用程序组成的应用层；(2)由开发人员可以直接调用的组件所组成的应用框架层；(3)对应用框架提供支撑的系统运行库层；(4)包括驱动模型、内存管理、进程管理、网络协议栈等组件的Linux内核层，它依赖于Linux内核，也同时作为硬件和软件栈之间的抽象层。文件管理器的基本功能部分，主要包括文件系统的显示，访问操作及其它一些辅助功能，通过界面与用户进行直接的交互。与Android自带的应用类似，该部分功能位于Android系统架构最上层应用层，同时依赖于系统应用程序框架层和系统库的支持。从Android系统架构不同层次来看，文件管理器基本功能部分运行在用户空间，文件管理器在整个Android系统架构中的层次如图3-1所示：



3-1 文件管理器在整个系统中的架构层次

## 3.2文件管理器的功能

类似于PC中的资源管理器，为实现用户对于手机文件资源管理的需求，手机文件管理器需要实现的功能主要有文件系统浏览，目录跳转，对文件执行复制、移动、删除等文件管理操作，文件搜索，文件共享辅助功能。文件管理器的手机用户用例图如图3-2所示：

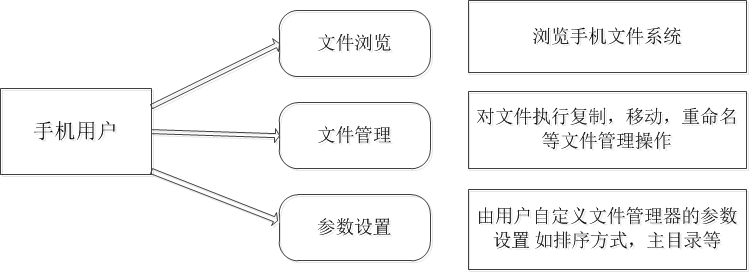


图3-2 手机用户用例图

# 4系统模块设计

## 4.1文件管理器界面的设计

点击运行文件管理器应用程序，首先进入的是Android文件管理器主界面，主要包含四个部分:第一部分为添加SD卡的部分，左边显示的添加的SD卡，最右边是点击添加SD卡的按钮。第二部分为显示当前Android用户浏览文件夹或

者文件的路径，当然用户也可以选择自己想要的路径进行切换到指定目录。第三部分为文件或者文件夹列表显示，默认显示路径为/mnt/sdcard目录，显示当前的文件。第四部分为最顶端的工具栏，主要包括以下几个功能，第一个按钮的功能是显示或者隐藏SD卡，其中，Android用户点击向上箭头为隐藏SD卡，点击向下箭头为展开SD卡。第二个按钮的功能是显示上一级目录，Android用户通过点击向上箭头来切换到自己指定的文件夹目录。第三个按钮的功能是显示历史记录，保存用户浏览过的文件夹目录地址，Android用户可以通过点击任意的目录路径来切换到当前用户指定的目录。第四个按钮的功能是选择复制多个或者单个文件或者文件夹。Android用户通过点击切换选择对多个文件的执行操作。第五个按钮的功能是刷新当前文件或者文件夹，以便及时更新用户当前文件内容。

第六个按钮的功能是菜单显示按钮，Android用户通过点击菜单选项按钮，来调出菜单选项，其中主要包括以下十大功能，分别是新建文件夹、新建文件、复制、粘贴、搜索、APK管理、设置、列表选项、文件库、退出，主界面如图4-1所示：

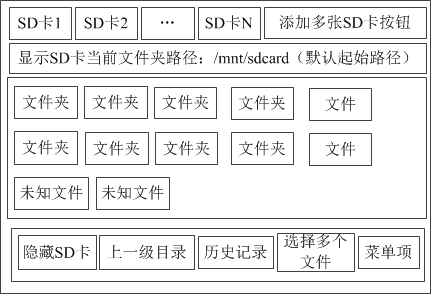


图4-1 文件管理器界面设计

当Android用户点击最底部工具栏中的菜单项时，其中包括了以下十个部分:新建文件夹、新建文件、复制对话框、粘贴、搜索、APK管理、设置、列表显示方式、文件库、退出，菜单选项界面设计如下图4-2所示：

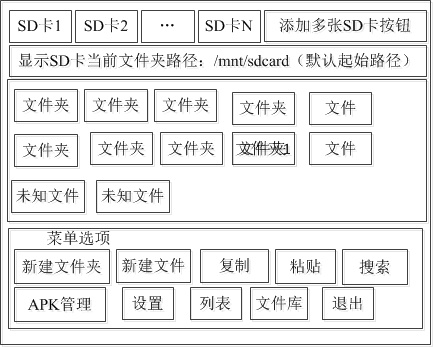


图4-2 文件管理器菜单选项界面设计

Android用户通过点击添加多张SD卡按钮来添加多张SD卡，方便用户操作，在显示SD卡路径的地址栏中，用户也可以通过选择地址栏上的路径来切换到用户指定的目录下，并对文件执行操作命令。在显示文件列表的视图中，Android用户可以看到当前/mnt/sdcard目录下的所有文件和文件夹，其中包括有txt文档，move影音模式，MP3音乐播放器，htm文件，photo照片，zip解压包，Apk应用程序，未知文件的类型。最底端的文件工具栏主要包括隐藏SD卡，显示上一级目录，显示当前用户浏览过的历史记录，选择多个文件或者当个文件，菜单项。Android用户通过点击经典菜单项来对文件夹进行操作功能。

4.2功能模块设计

Android文件管理器是一款操作简单，界面华丽，占用内存资源小的一款手机软件，该软件包含了对文件或者文件夹的基本功能操作，例如我们经常会对文件或者文件夹进行新建一个文件夹或者文件，还可以复制或者剪切到另外一个文件夹(也称为移动文件)。Android文件管理器除了这些基本的常用功能操作之外，还提供了丰富且具有特色功能的文件操作，为Android用户提供更为快捷简便的文件操作。Android文件管理器支持多标签页的显示功能，Android用户可以根据自己的需求来新建多张SD卡，来存放用户文件。

文件搜索功能： Android用户可以通过点击菜单选项，选择文件搜索来对文件进行查找功能。首先用户输入关键字进行查找，Android文件管理器提供两种搜索模式:一种是大小写敏感搜索。另一种是完全匹配搜索。通过这两种搜索Android用户可以精确的查找到自己想要查找的文件。

文件设置功能：Android用户通过点击菜单选项，选择设置，来对文件夹或者文件有一个详细的说明和解释，并且提供关于/帮助用户手册，来协作Android用户的操作。文件设置功能主要包括隐藏以.开头的文件、修改带root权限、显示文件大小(以列表的形式显示)、显示文件时间(以列表的形式显示)、查看SD卡容量、查看手机内部存储空间、显示Android文件管理器关于/帮助信息。

文件列表显示功能：提供两种列表显示功能，一种是以图表的形式来显示当前文件或者文件夹，另一种是以列表的形式来显示当前文件或者文件夹的详细信息，包括创建文件或者文件夹的大小和时间。

APK应用程序软件管理功能：提供查看软件的详细信息，列出软件的版本号，是否强制停止当前的应用程序和卸载该应用程序，不仅如此，还提供清楚数据缓存的功能，Android用户可以通过点击Clear data按钮清楚数据缓存。最后还提供了对权限的说明包括当前SD卡的存储信息、个人信息、手机状态信息。Android用户也可以通过点击Show all来显示system的权限信息。

文件拖拽功能：Android用户通过长按文件夹或者文件来弹出拖拽文件菜单项。通过点击拖拽来执行拖曳文件操作，根据拖拽文件位置的不同分为以下三种:第一种，如果拖曳文件拖拽到自己本身的位置，则系统提示是否删除当前文件。第二种，如果拖拽文件至文件地址显示栏以上的话，则不执行任何操作。第三种，如果拖拽文件至最底端的工具栏上，则提示用户是否复制到当前文件夹上。

4.3数据流模型

通常，数据流图是分层绘制的，整个过程反映了自顶向下进行功能分解和细化的分析过程。顶层（也称第0层）DFD用于表示系统的开发范围，以及该系统与周围环境的数据交换关系；最底层DFD代表了那些不可进一步分解的"原子加工"；中间层DFD是对上一层父图的细化，其中的每一个加工可以继续细化，中间层次的多少由系统的复杂程度决定。

（1） 第0层DFD将整个系统表示成一个加工；

（2） 确定并标记主要的输入和输出；

（3） 分离出下一层中的加工、数据对象和存储，并对其进行细化，一次细化一个加工；

（4） 标记所有加工和箭头；

（5） 重复步骤（3）和（4），直到所有的加工只执行一个简单的操作，可以很容易地用程序实现。

第0层DFD图如图4-3所示

文件管理者(用户)维护文件信息和操作信息，并登陆文件相关操作与信息。 图4-3 文件管理器DFD顶层图

第1层DFD图如图4-4所示，对第0层DFD图中的一个加工"文件explorer"进行展开：



图4-3 文件管理器DFD第一层图

第2层DFD图如图4-5所示，对第1层DFD图中的一个加工”explorer操作”进行展开

图4-3 文件管理器DFD第一层图

绘制第0层DFD的时候，将整个系统看成一个加工，然后找出作用于该加工的外部实体，以及相应的数据输入和输出。对于"安卓文件exploerer"而言，整个系统就是一个加工"explorer"。从用户的需求描述可知，"用户"是数据的源点，"目标程序"是数据的终点。另外，用户需要录入提供文件操作命令，说明各个操作数据流；同样，"操作请求"和"操作结果"也是数据流。根据上述分析，得到如图所示的第0层DFD。绘制下一层数据流图时，细化第0层的加工"文件explorer"，从而描述系统的主要功能。第0层DFD得知，"操作命令"是用户需要提供的一个信息，同样得到对文件explorer这个中间系统有关文件操作的另一个加工。另外，同时增加这些数据流对应的数据存储，第1层DFD。为了继续进行分解，我们分析第1层DFD中的加工对目标程序有关文件explorer的操作。操作时需要提供合法性检查，因此，此次加工可以分解为"合法性检查"和"查询成绩"两个处理步骤，从而形成如图所示的第2层DFD。

# 5 系统基本功能的实现

## 5.1 文件管理器主界面活动的实现

文件管理器主界面活动的实现过程中，主要设计了以下几个类

(1) 继承自Activity的活动类，作为应用的入口在配置文件中中进行注册，主要负责应用主界面的显示处理及与用户的文件管理操作进行交互。

(2) 活动中继承自系统BaseAdapter的FileListAdapter类，作为Listview列表项的适配器，完成界面列表项的布局设计，资源绑定等功能，在其中覆盖了父类的getView()函数。

(3)FileManager类中封装了各种文件的操作方法，主要在界面文件操作菜单响应过程中被调用。通过调用java.io.File类库实现对Android文件系统的访问操作。

listview是Android开发中最常用的组件之一，它通过一个适配器(adapter)以列表项的形式显示指定的数据，通过使FilerActivity继承自ListActivity，可以满足用户方便直观地浏览Android文件系统内容的要求。适配器是用来把数据映射到listview上的中介，通过setListAdapter( )函数可以实现当前listview与指定适配器的联系。为了更灵活地实现本应用中界面的自定义效果，在FilerAetivity活动中定义一个继承自Android。ArrayAdapter的适配器FileListAdapter，在其中的构造函数中指定主界面listview中每一行布局的xml文件，同时新建getView()函数以覆盖父类中的同名函数，在其中实现布局id与指定资源数据的绑定，使主界面上显示达到的效果。

选项菜单(OptionsMenu)和上下文菜单(ContextMenu)是Android开发中最常见的元素，为某些操作提供了合适的入口。开发人员只需预先设计好相应的菜单选项，以xml文件的形式存放，在随后执行系统回训函数的过程进行有选择的显示菜单项即可。相比而言，ContextMenu更适合设置一些与单个view项操作相关的菜单项，OptionsMenu适合设置一些与整体相关的选项。可以把文件重命名、删除、查看属性放在ContextMenu中实现，而搜索文件、排序方式设置等操作由OptionsMenu完成。

在Android本身的实现中，大量用到了Preference Activity对系统进行信息配置和管理。文件管理器的开发中，也同样使用FilerPreferences-Activity为用户提供参数设置的接口。它的布局文件放在固定的路径res/xml下。由FileActivity中OptionsMenu中的一个菜单项来启动。

用户的界面操作及对应的响应函数如下表5-1

表5-1 用户界面操作

|  |  |
| --- | --- |
| 用户操作 | 对应的响应函数 |
| 点击某view视图 | onListItemClick |
| 长按某view视图 | onCreateContextMenu |
| 选择ContextMenu菜单项 | onContextItemSelected |
| 选择view中的icon图标 | getView函数中设定的监听器 |
| 按menu键 | OnCreateOptiondMenu |
| 选择OptionsMenu菜单项 | onOptionsItemSelected |
| 按回退键 | onKeyDown |
| 按home键 | Activity Management管理进程 |

## 5.2 文件操作功能的实现

本应用要实现的文件管理功能上要有单个或批量文件的移动、复制、重命名等操作。针对应用的具体情况，在界面底部建立了一个工具栏(包括yankbar, copybar, movebar)，配合listview中各行的icon按钮，更方便地完成对文件管理的大部分操作。

在程序中，建立了File类型的全局变量mCur-Dir，存放将要进入或正在显示的目录信息；ArrayList<String>类型的全局变量mCurF les存放将要显示或正在显示的目录中的文件列表；ArrayList<String>类型的全局变量mYanked存放即将执行复制/移动等操作的文件路径集合；Stack<HistoryPathlnfo>类型的变量mPathHistory存放操作过程中的目录历史路径；Boolean类型变量incopymove作为应用。存放是否正处于即将复制/移动文件的状态标志。文件管理操作执行的过程中所调用的主要系统函数如表5-2所示：

表5-2 程序调用文件管理器的主要系统函数

|  |  |
| --- | --- |
| 函数名 | 函数说明 |
| isDirectory() | 判断文件是否为文件夹 |
| Exists() | 判断当前的文件系统中是否有该文件 |
| getAbsolutePath() | 获得文件的绝对路径 |
| getParentFile() | 返回当前文件父本文件名生成的新文件对象 |
| List() | 返回当前文件目录下所有文件的名称列表信息 |
| getName() | 获得文件的名称信息 |
| renameTo(File) | 重命名当前文件名为参数文件名称 |
| mkdir() | 用当前文件名创建一个文件夹 |
| delete() | 删除指定文件，如为文件夹，必须为空 |

## 5.3 创建文件/文件夹功能的实现

创建文件/文件夹功能入口在下方工具栏中，实现了HorizontalListView，继承了HorizontalSrcollView,在其中的设置选项中，以GridView的形式调用了initCreateFileDialog()函数为用户提供了创建文件与文件夹的选项。在filemenuItem中设置了设置的样式。

## 5.4 文件分享功能的实现

文件分享功能使用微信提供的接口，在本地环境中，将微信提供的接口函数存放至libs目录下,注册ShareSDK并得到AppKey，下载SDK的压缩包，得到libs目录，其中包含ShareSDK的类库，将Libs\Global-Dependences下的jar包复制到libs目录下，从Libs\platforms选择所需平台，复制相应压缩包到本地项目的libs文件夹中并解压。

# 6 接口设计

## 6.1 界面接口

6.1.1拖动文件图标

**public void** drag(**int** position)

|  |  |
| --- | --- |
| Key | Value |
| position | 图标位置 |

## 6.2 文件操作接口

6.2.1获取文件夹大小

**public static long** getDirectorySize(File f) **throws** IOException

|  |  |
| --- | --- |
| Key | Value |
| f | 得到该目录下所有文件的大小 |

6.2.2粘贴文件

**public static void** copyFile(String oldPath, String newPath)

|  |  |
| --- | --- |
| key | Value |
| oldPath | 文件的原目录 |
| newPath | 目标目录 |

6.2.3获取文件属性

**private void** doGetProperties(String f)

|  |  |
| --- | --- |
| key | Value |
| f | 文件路径 |