**基于Android系统的文件管理器**

**方案概要设计**

**（v1.0）**

**IT FARMERS**

2016年4月

目录

目录

[1 引言 3](#_Toc448079071)

**[1.1 目的](#_Toc448079072)** [3](#_Toc448079072)

**[1.2 背景](#_Toc448079073)** [3](#_Toc448079073)

**[1.3 范围](#_Toc448079074)** [3](#_Toc448079074)

**[1.4系统概述](#_Toc448079075)** [4](#_Toc448079075)

[2系统环境 4](#_Toc448079076)

**[2.1运行环境](#_Toc448079077)** [4](#_Toc448079077)

**[2.2开发环境](#_Toc448079078)** [4](#_Toc448079078)

[3 系统总体设计 5](#_Toc448079079)

**[3.1 系统架构层次设计](#_Toc448079080)** [5](#_Toc448079080)

**[3.2 文件管理器的功能](#_Toc448079083)** [6](#_Toc448079083)

[4 系统功能的设计 7](#_Toc448079084)

**[4.1 文件管理器Activity的设计](#_Toc448079085)** [7](#_Toc448079085)

**[4.2](#_Toc448079086)** [各界面之间的转换 8](#_Toc448079086)

[5 文件管理器基本功能的实现 11](#_Toc448079087)

**[5.1 文件管理器主界面活动的实现](#_Toc448079088)** [11](#_Toc448079088)

**[5.2 文件操作功能的实现](#_Toc448079089)** [11](#_Toc448079089)

**[5.3 创建文件/文件夹功能的实现](#_Toc448079090)** [12](#_Toc448079090)

**[5.4 文件分享功能的实现](#_Toc448079091)** [12](#_Toc448079091)

[6 接口设计 13](#_Toc448079092)

**[6.1 界面接口](#_Toc448079093)** [13](#_Toc448079093)

**[6.2 文件操作接口](#_Toc448079094)** [13](#_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

本文档定义了与桌面系统体系结构、接口设计、服务器架构、网络管理的具体实现方法。

## 1.4系统概述

项目名称：基于Android的文件Explorer

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

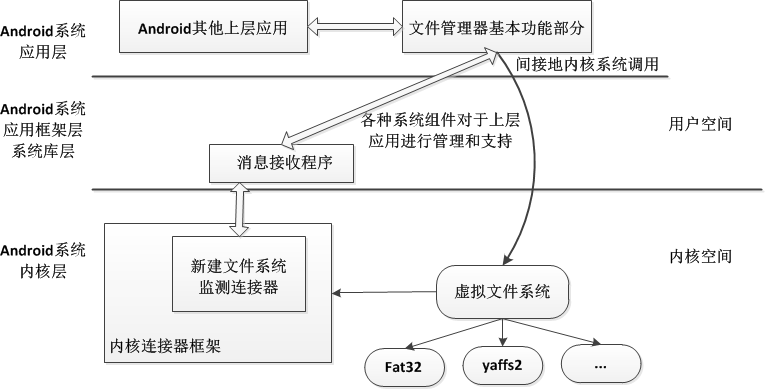
本系统的的开发环境如下：

* 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。根据文件管理器的功能需求及开发和编译过程中涉及Android系统层次的不同，文件管理器的实现可以分为两部分：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：

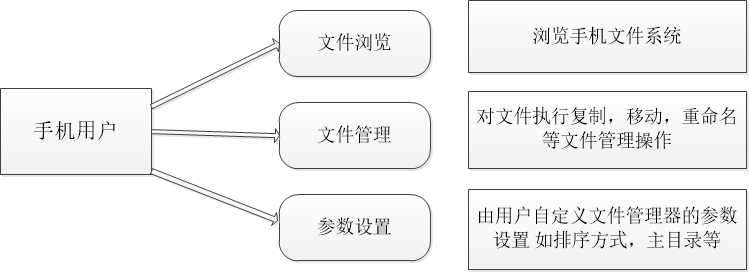


图3-2 手机用户用例图

# 4系统功能的设计

## 4.1文件管理器Activity的设计

根据系统的功能需求，文件管理器主要由3个界面组成，它们分别是作为应用主界面的FilerActivity、进行参数设置的FilerPreferences-Activity以及进行文件搜索的SearchActivity。

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。

另外，文件管理器器还提供了文件搜索的功能，即能够根据关键依次匹配指定目录下的文件/文件夹名称，并以列表形式显示出搜索的结果，该功能由SearchActivity来实现。Android 的搜索功能能够帮助开发者很好地管理搜索对话框，使开发人员小需要自己去开发一个搜索框，不需要考虑把搜索枢放在什么位置 ,搜索框是否会影响当前的界而等不重要的细节问题，所有这些工作都由搜索管理器（SearchManager）来辅助实现。当Android搜索栏中输入关键字执行搜索时，搜索管理器将使用一个专门的Intent”android.intent-action.SEARCH”，将它传给配置文件Android-Manifest.xml中声明的处理搜索结果的activity，即本应用中的SearchActivity。通过get-Intent().getStringExtra()函数即可获得用户输入的关键字，然后按关键字执行搜索操作，并以列表形式显示出得到的结果。用ArrayList<String>类型的变量mSearchedFiles记录找到每个目标的路径信息。整个搜索的大致流程如图4-2所示：

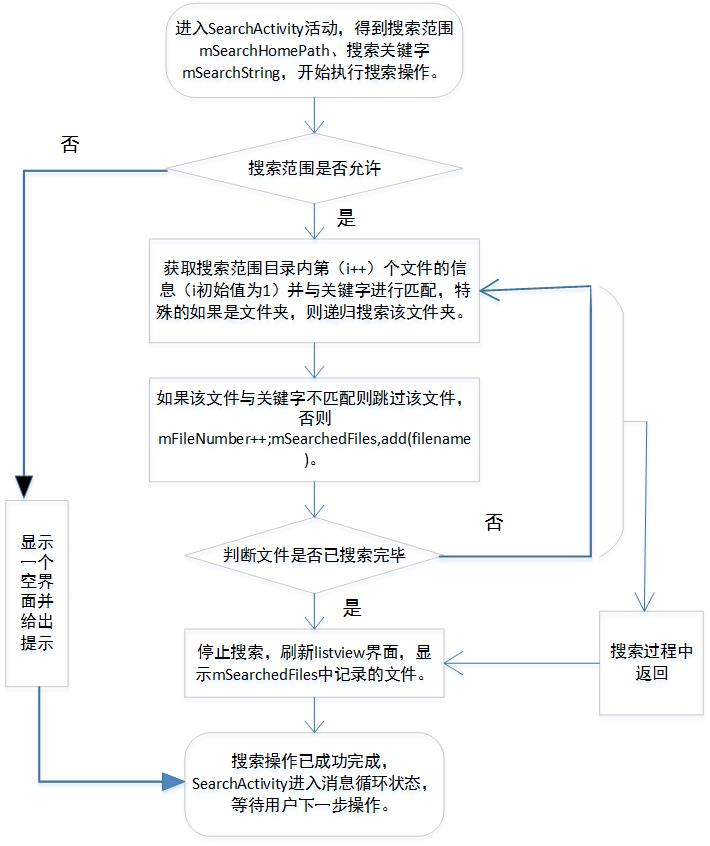


图4-2 文件搜索过程的流程图

## 4.2 各界面之间的转换

Android 应用开发中，每一个界面即对应着一个activity。用户在使用手机软件时，需要在不同的界面之间来回转换。相对应的程序中每一个activity都由开始和结束的过程。

在配置文件AndroidManifest.xml，应用包含的所有活动及每个话动所要响应的intent，然后在程序中需要的地方用startActivity()函数向系统发送指定的Intent，由系统启动合适的活动响应lntent。在程序中，用Activity.finish()函数结束当前活动，由活动管理器(Activity Manager)确定下一个进程的到来。在文件管理器的整个应用中，各个activity之间的转换如图4-3所示：

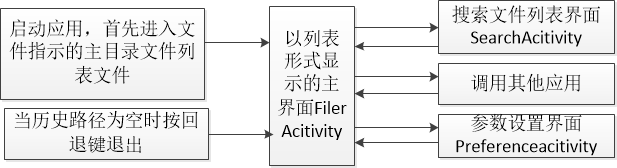


图4-3 文件管理器中各个activity之间的转换

# 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。

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。

MVC (Model-View-Controller)是当前比较流行的软件开发框架，它的思想与观察者模式类似。Android项目整体就是一种典型的MVC结构，其中广泛使用的xml布局文件即是view，activity和intent，起到了控制器的作用，提供者对数据层做了良好的封装，而且提供者把数据管理的范畴从数据库泛化到了数据的概念。通过使用MVC模式，可以使文件管理器的界面设计工作和文件系统相关处理操作相互独立，让界面的设计更加灵活，文件管理功能也可以进行模块化的开发。同时也能够使开发人员专心处理前端界面显示与后台文件系统数据之间的关系，设计出高效稳定的程序。

在视图层，根据不同的用户操作及应用的状态，由逻辑层控制界面布局的变化及显示，辅助完成各种文件操作的过程。应用中使用了大量的ContectMenu、OptionsMenu等系统菜单项，在合适的位置为用户提供文件操作的入口。另外针对应用的具体情况，在界面底部建立一个工具栏，通过控制文件选择、操作执行、界面跳转等过程，再配合ListView()函数所设置每一行的显示效果，使用户更方便的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 |