|  |
| --- |
| 哈尔滨工业大学计算机学院 |
| 基于android的文件管理器结题报告 |
| 软件设计与开发实践III结题 |
|  |
| **李洪祥 1100300220** |
| **赵 岩 1100300214** |
| **2013/6/13 Thursday** |

目录

[一、 主要内容 2](#_Toc359084414)

[二、 技术难点 3](#_Toc359084415)

[1. 对于android开发架构的理解 3](#_Toc359084416)

[2. 程序调用文件管理的主要操作 5](#_Toc359084417)

[3. 复制（剪切）-粘贴功能的实现 6](#_Toc359084418)

[4. 多线程的实现 7](#_Toc359084419)

[5. 调用微博分享功能的实现 8](#_Toc359084420)

[三、 进展情况 8](#_Toc359084421)

[四、 代码和界面编写 9](#_Toc359084422)

[五、 实验结果和分析 9](#_Toc359084423)

[1、 断点调试 10](#_Toc359084424)

[2、 添加日志 10](#_Toc359084425)

[3、 真机测试 10](#_Toc359084426)

[六、 对课程的建议 10](#_Toc359084427)

[1. 增加助教数量 11](#_Toc359084428)

[2. 增强监管力度 11](#_Toc359084429)

[3． 关于项目容器 11](#_Toc359084430)

[七、 参考文献 11](#_Toc359084431)

[八、 附录： 12](#_Toc359084432)

[1. MainActivity.java 12](#_Toc359084433)

[2. MIME.java 46](#_Toc359084434)

[3. AndroidMain.xml 48](#_Toc359084435)

[4. string.xml 49](#_Toc359084436)

**基于android的文件管理器结题报告**

# 主要内容

在移动终端技术与互联网不断发展与融合的今天，手机已从简单的通话工具发展为集PDA、互动游戏、高分辨率摄像、移动视听于一体的全功能通信、数据处理工具，手机上的信息存储量也越来越多。但Android平台自身并不提供资源管理器功能，提供的应用程序接口（API）很简单。因此，需要开发实用的Android手机文件管理器，以满足广大用户对资源管理的需求。

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、移动等文件管理操作，调用其他应用对特定文件执行相关操作，文件排序，查看文件属性等辅助功能。

本应用基于Android SDK 1.8 平台，以Eclipse-adt为集成开发环境，选择Android上层应用开发通用的Java和xml语言，采用Activity显示界面，主要布局为ListView和GridView，通过File和IO的操作来是实现基本功能，通过，用户通过安装.apk文件进行使用，操作手机上的文件。通过Android SDK自带的adb工具和真机上的android系统实现程序的调试和运行。

实现的功能如下：



# 技术难点

## 对于android开发架构的理解

* 1. 应用程序

Android会同一系列核心应用程序包一起发布，该应用程序包包括email客户端，SMS短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用JAVA语言编写的。

1.2应用程序框架

开发人员也可以完全访问核心应用程序所使用的API框架。该应用程序的架构设计简化了组件的重用；任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块（不过得遵循框架的安全性限制）。同样，该应用程序重用机制也使用户可以方便的替换程序组件。[2]

1.3 程序库

Android 包含一些C/C++库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。

1.4 Android 运行库

Android 包括了一个核心库，该核心库提供了JAVA 编程语言核心库的大多数功能。

1.5 Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 Linux 内核也同时作为硬件和软件栈之间的抽象层。

各模块实现的方法如下：



## 程序调用文件管理的主要操作

本程序实现文件管理器的各种操作如复制、剪切、粘贴等工作，都是通过调用和封装文件管理的系统函数来实现的，所以在写程序之前了解有哪些函数可以调用是极为重要的，在程序中我们主要参考了以下函数列表

|  |  |
| --- | --- |
| 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信息 |
| 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 |
| getParentFile() | 返回由当前文件父文件名生成的新文件对象 |
| mkdir() | 用当前文件名创建一个文件夹 |
| delete() | 删除指定文件。如果为文件夹，必须为空 |

## 复制（剪切）-粘贴功能的实现

在本程序中对于文件（夹）的复制为通过以字节为单位读取源文件，再以字节为单位写向目标文件。读写操作通过FileInputStream/FileOutputStream和DataInputStream/DataOutputStream来实现。

复制的流程图如下：



图 1

## 多线程的实现

在粘贴文件的时候一方面要执行粘贴操作，另一方面要传递信息给进度条，便于进度条显示进度，此处要新开启一个进程，具体代码如下：

|  |
| --- |
| **new** Thread()  {  **public** **void** run()  {  copy(startPath, desPath);  // 通过Message对象向原线程传递信息  Message message = Message  .*obtain*();  messageHandler  .sendMessage(message);  progressDialog.dismiss();  // 如果为剪切则删除对应文件/文件夹  **if** ("cut".equals(flag))  {  **if** (copyPath.isFile())  {  copyPath.delete();  } **else**  {  delete(copyPath);  }  }  }  }.start();  } |

## 调用微博分享功能的实现

通过ResolveInfo得到所有具有SEND\_TO功能的应用名称，若包含“微博”的应用，则通过intent方法调用微博进行分享功能，否则提示错误。

# 进展情况

本次软件设计进度完全按照开题时的计划进行，基本完成了所有功能，发送功能因为没有查到相关文献，所以没有实现，但是我们另实现了微博分享的功能，抵消了这个不足

|  |  |  |
| --- | --- | --- |
| 周次 | 设计任务及要求 | 完成情况 |
| 2 | 确定题目，明确设计任务（具体到功能），搭建编程环境 | **完成** |
| 3-4 | 搜集Android相关资料了解与课题相关的知识, | **完成** |
| 5-7 | 进一步熟悉Android接口，以及相关编程环境，并开始着手编写程序，完成文件、文件夹的浏览功能 | **完成** |
| 8-10 | 完成文件、文件夹的复制、剪切、粘贴、新建、删除、重命名、根据文件名搜索文件等功能 | **完成** |
| 11-12 | 完成文件的发送、排序等功能 | **基本完成** |
| 13-15 | 尽量完成额外的功能如根据内容搜索文件等等 | **完成** |
| 16 | 写结题报告 | **完成** |

# 代码和界面编写

见附录

# 实验结果和分析

本次实验结果令人满意，实现了除了文件发送外的全部功能，虽然界面不够很美观，但是也可以接受，下面是实验结果的截图。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 主界面 | **可以进入系统文件夹** | **功能键** |
|  |  |  |
| 对文件（夹）的操作 | **进度条** | **微博分享** |

在实现功能后我们曾经对我们的程序进行充分的测试，保证没有bug，测试的方法包括：

## 断点调试

在代码容易出错的地方，进行断点调试，对变量进行监测，检查是否有错误。

## 添加日志

在函数和有分支的地方加入日志，运行的时候通过分析日志，可以判断程序流向是否正确

## 真机测试

在真机上测试程序的各个功能，并测试一些极限情况，例如不复制就粘贴，将出bug的可能性降低到最小。

# 对课程的建议

此课程锻炼了我们的程序设计能力，使我们熟练地掌握了一门程序设计语言及其开发环境。是我们更加深入地了解了一些数据结构的设计能力，并进一步加强了我们对操作系统的认知和应用能力。

对本课程我们有以下建议：

## 增加助教数量

在本课程中，我明显地感觉到了助教的数量不够，有一些问题找不到老师来解答，而且因为每个老师都有自己的研究方向，所以可能对我们所做的东西不是理解的非常透彻。所以我建议根据小组所做的方向，有针对性的增派更多的助教

## 增强监管力度

在本课程中，有很多同学缺席日常的软件设计课，尤其是中期检查后，更是如此，而在寝室做软件设计的人数也不是很客观，所以希望在以后的课程中加强监管力度，是我们的学弟学妹们更有动力去学习。

### 3． 关于项目容器

在课程主页中我看见了正在开发的项目容器，我感觉这个想法很好，希望这个模块尽快开发完成，相信这个模块会使得此课程变得更加高效！

# 参考文献

[1] Lauren Darcey Shane Conder. Android Wireless Application Development[M]．United States：Pearson Education，2001：174-217．

[2]王振丽．Android 底层开发技术实战详解——内核、移植和驱动[M]．北京: 电子工业出版社，2012．

[3]柯元旦. Android 内核剖析[M].北京：电子工业出版社.2011

[4]余志龙. Google SDK 开发范例大全[M]. 北京：电子工业出版社.2010

[5]陈煜，周荣辉． 基于Android系统的手机文件管理器的设计与实现[D]．铁路计算机应用，2012，21(9)：47-51

# 附录：

## MainActivity.java

|  |
| --- |
| **package** com.explorer;  **import** **static** com.mime.MIME.*MIME\_MapTable*;  **import** java.io.BufferedInputStream;  **import** java.io.BufferedOutputStream;  **import** java.io.DataInputStream;  **import** java.io.DataOutputStream;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.FileOutputStream;  **import** java.text.SimpleDateFormat;  **import** java.util.ArrayList;  **import** java.util.Arrays;  **import** java.util.Comparator;  **import** java.util.HashMap;  **import** java.util.List;  **import** java.util.Map;  **import** android.app.Activity;  **import** android.app.AlertDialog;  **import** android.app.Dialog;  **import** android.app.ProgressDialog;  **import** android.content.ComponentName;  **import** android.content.DialogInterface;  **import** android.content.Intent;  **import** android.content.pm.ApplicationInfo;  **import** android.content.pm.PackageManager;  **import** android.content.pm.ResolveInfo;  **import** android.net.Uri;  **import** android.os.Bundle;  **import** android.os.Environment;  **import** android.os.Handler;  **import** android.os.Looper;  **import** android.os.Message;  **import** android.view.KeyEvent;  **import** android.view.LayoutInflater;  **import** android.view.Menu;  **import** android.view.MenuInflater;  **import** android.view.MenuItem;  **import** android.view.View;  **import** android.view.View.OnClickListener;  **import** android.widget.AdapterView;  **import** android.widget.AdapterView.OnItemClickListener;  **import** android.widget.AdapterView.OnItemLongClickListener;  **import** android.widget.Button;  **import** android.widget.EditText;  **import** android.widget.ListAdapter;  **import** android.widget.ListView;  **import** android.widget.SimpleAdapter;  **import** android.widget.TextView;  **import** android.widget.Toast;  **import** android.text.style.SuperscriptSpan;  **import** android.util.Log;  **import** com.file.R;  **public** **class** MainActivity **extends** Activity **implements** OnClickListener,  OnItemClickListener, OnItemLongClickListener {  **private** TextView currentDir;// 用于显示文件夹的额目录，载入相应的文件夹，为新建的文件夹建立相应的目录  **private** Button btnC;// 手机内存 用于定位根目录  **private** Button btnE;// 存储卡 用于定位根目录  **private** ListView listView; // 显示相应按文件夹  **private** File rootDir; // 根目录文件夹  **private** File copyPath; // 当执行复制、粘贴等工作时，将原地址存储到copyPath中记录  **private** String flag;// 用于记录执行的操作，包括复制、剪切等  **private** String startFilePath;// 当执行粘贴等工作时，记录原地址  **private** String desFilePath;// 当执行粘贴等工作时记录目的地址  **private** ProgressDialog progressDialog;// 用于复制时的进度条  **private** **int** currentLen = 0;// 用于记录复制当前的文件（夹）数  **private** **long** totaLength = 0;// 用于记录总共要复制的文件（夹）数  **private** Handler messageHandler; // 主要接受子线程发送的数据, 并用此数据配合主线程更新UI.  **private** **static** String *Name* = "MainActivity";  **private** **int** sort = 0; // 0 = name and type; 1 = time; 2 = size  /\*\* Called when the activity is first created. \*/  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  Log.*v*(*Name*, "level1 onCreate");  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*main*);  currentDir = (TextView) findViewById(R.id.*currentDir*);  // fileName = (TextView) findViewById(R.id.name);  btnC = (Button) findViewById(R.id.*btnC*);// 手机内存按钮的声明  btnE = (Button) findViewById(R.id.*btnE*);// 存储卡按钮的声明  btnC.setOnClickListener(**this**);// 为手机内存按钮设置监听器  btnE.setOnClickListener(**this**);// 为存储卡按钮设置监听器  listView = (ListView) findViewById(R.id.*listView*);// 文件列表的晟敏  listView.setOnItemClickListener(**this**);// 为每个文件（夹）设置短按的监听器  listView.setOnItemLongClickListener(**this**);// 为每个文件（夹）设置长按的监听器  // 得到当前线程的Looper实例，由于当前线程是UI线程也可以通过Looper.getMainLooper()得到  messageHandler = **new** MessageHandler(Looper.*myLooper*());  // 设置根目录  **if** (Environment.*getExternalStorageState*().equals(// 判断是否挂载SD卡  Environment.*MEDIA\_MOUNTED*)) {  Log.*v*(*Name*,  "level2 Environment.getExternalStorageState().equals(Environment.MEDIA\_MOUNTED)");  rootDir = Environment.*getExternalStorageDirectory*();  } **else** {  Log.*v*(*Name*,  "level2 !Environment.getExternalStorageState().equals(Environment.MEDIA\_MOUNTED)");  rootDir = Environment.*getRootDirectory*();  }  loadFiles(rootDir);  }  // 自定义Handler  **class** MessageHandler **extends** Handler {  **public** MessageHandler(Looper looper) {  **super**(looper);  }  @Override  **public** **void** handleMessage(Message msg) {  Log.*v*(*Name*, "test initial rootdir"  + currentDir.getText().toString());  loadFiles(**new** File(currentDir.getText().toString()));  }  }  @Override  **public** **void** onBackPressed() {  // **TODO** Auto-generated method stub  Log.*v*(*Name*, "level1 onKeyDown");  // **TODO** Auto-generated method stub  Map<String, Object> map = (Map<String, Object>) **this**.listView  .getItemAtPosition(0);  **final** File file = (File) map.get("file");  **final** String str = (String) map.get("name");  // 如果不是根目录，则返回上一层目录  **if** (str.equals("上一级目录")) {  Log.*v*(*Name*, "level2 notrootDir");  loadFiles(file);  } **else**  // 如果为根目录，则退出  {  Log.*v*(*Name*, "level2 isrootDir");  MainActivity.**this**.finish();  }  }  @Override  // 初始化菜单，只会在第一次初始化菜单时调用  **public** **boolean** onCreateOptionsMenu(Menu menu) {  Log.*v*(*Name*, "level1 onCreateOptionsMenu");  MenuInflater inflater = getMenuInflater();  inflater.inflate(R.menu.*menu*, menu);  **return** **true**;  }  @Override  **public** **boolean** onOptionsItemSelected(MenuItem item) {  Log.*v*(*Name*, "level1 onOptionsItemSelected");  // 如果选择的是新建文件夹选项  **if** (item.getItemId() == R.id.*newFile*) {  Log.*v*(*Name*, "level2 item.getItemId() == R.id.newFile");  LayoutInflater factory = LayoutInflater.*from*(MainActivity.**this**);  **final** View view = factory.inflate(R.layout.*rename*, **null**);  AlertDialog d = **new** AlertDialog.Builder(MainActivity.**this**)  .setCancelable(**true**)  .setMessage("文件夹名")  .setView(view)  .setPositiveButton("确定",  **new** DialogInterface.OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog,  **int** which) {  Log.*v*(*Name*, "level3 onClick");  String dirName = ((EditText) view  .findViewById(R.id.*rename*))  .getText().toString();  String newFile = currentDir.getText()  .toString() + "/" + dirName;  **if** (**new** File(newFile).exists()) {  Log.*v*(*Name*,  "level 4 File(newFile).exits");  Toast.*makeText*(MainActivity.**this**,  "文件夹已存在", Toast.*LENGTH\_LONG*)  .show();  **return**;  }  File f = **new** File(currentDir.getText()  .toString(), dirName);  f.mkdir();  loadFiles(**new** File(currentDir.getText()  .toString()));  }  }).create();  d.show();  } **else** **if** (item.getItemId() == R.id.*open*) {  Log.*v*(*Name*, "level2 item.getItemId() == R.id.newFile");  LayoutInflater factory = LayoutInflater.*from*(MainActivity.**this**);  **final** View view = factory.inflate(R.layout.*rename*, **null**);  AlertDialog d = **new** AlertDialog.Builder(MainActivity.**this**)  .setCancelable(**true**)  .setMessage("需要打开的文件（夹）名")  .setView(view)  .setPositiveButton("确定",  **new** DialogInterface.OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog,  **int** which) {  Log.*v*(*Name*, "level3 onClick");  String dirName = ((EditText) view  .findViewById(R.id.*rename*))  .getText().toString();  String newFile = currentDir.getText()  .toString() + "/" + dirName;  **if** (**new** File(newFile).exists()) {  **final** File file = **new** File(newFile);  **if** (file.isDirectory()) {  Log.*v*(*Name*,  "level2 file.isDirectory");  **try** {  loadFiles(file);  } **catch** (Exception e) {// 若遇到权限不足的情况，则弹出警告  }  } **else** {// 如过是文件，则选择相应应用打开此文件  openFile(file);  }  } **else** {  Toast.*makeText*(MainActivity.**this**,  "此文件（夹)不存在", Toast.*LENGTH\_LONG*)  .show();  }  File f = **new** File(currentDir.getText()  .toString(), dirName);  f.mkdir();  loadFiles(**new** File(currentDir.getText()  .toString()));  }  }).create();  d.show();  } **else** **if** (item.getItemId() == R.id.*exit*) {  Log.*v*(*Name*, "level2 item.getItemId() == R.id.exit");  MainActivity.**this**.finish();  } **else** **if** (item.getItemId() == R.id.*paste*) {  paste();  } **else** **if** (item.getItemId() == R.id.*sort*) {  // sort and refresh  selectSortFiles();  } **else** **if** (item.getItemId() == R.id.*share*) {  share();  }  **return** **true**;  }  **public** **void** selectSortFiles() {  AlertDialog dialog = **new** AlertDialog.Builder(MainActivity.**this**)  .setTitle("排序")  .setItems(**new** String[] { "按类型", "按修改时间", "按大小", "按名称" },  **new** DialogInterface.OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog,  **int** which) {  sort = which;  // Toast.makeText(MainActivity.this, "" +  // which, Toast.LENGTH\_LONG).show();  loadFiles(**new** File(currentDir.getText().toString()));  }    }).create();  dialog.show();    }  /\*\*  \* 加载当前文件夹列表  \*  \*/  **public** **void** loadFiles(File dir) {  Log.*v*(*Name*, "level1 loadFiles");  List<Map<String, Object>> list = **new** ArrayList<Map<String, Object>>();// 声明一个Map数组，用来存储文件（夹）的显示信息  // 如果目录不为空的化，则显示相应的文件（夹）信息  **if** (dir != **null**) {  Log.*v*(*Name*, "level2 dir!= NULL");  // 如果不是根目录的话，则为 上级目录在ListView的最上方留一个接口  **if** (!dir.getAbsolutePath().equals(rootDir.getAbsolutePath())) {  Log.*v*(*Name*,  "level3 !dir.getAbsolutePath().equals(rootDir.getAbsolutePath())");  Map<String, Object> map = **new** HashMap<String, Object>();  map.put("file", dir.getParentFile());  map.put("name", "上一级目录");  map.put("img", R.drawable.*folder*);  list.add(map);  }  // 设置显示目录  currentDir.setText(dir.getAbsolutePath());  File[] files = dir.listFiles();  sortFiles(files);  // 为每跟文件（夹）的显示做准备，先将信息存储起来  **if** (files != **null**) {  Log.*v*(*Name*, "level2 files!=NULL");  **for** (File f : files) {  Map<String, Object> map = **new** HashMap<String, Object>();  map.put("file", f);  map.put("name", f.getName());  map.put("img",  f.isDirectory() ? R.drawable.*folder*  : (f.getName().toLowerCase()  .endsWith(".zip") ? R.drawable.*zip*  : R.drawable.*text*));  list.add(map);  }  }  } **else** {// 如果目录不存在则提示错误  Log.*v*(*Name*, "level2 Files == NULL");  Toast.*makeText*(**this**, "目录不正确，请输入正确的目录!", Toast.*LENGTH\_LONG*).show();  }  // 显示文件（夹）信息  ListAdapter adapter = **new** SimpleAdapter(**this**, list, R.layout.*item*,  **new** String[] { "name", "img" }, **new** **int**[] { R.id.*name*,  R.id.*icon* });  // listView.setChoiceMode(ListView.CHOICE\_MODE\_MULTIPLE);  listView.setAdapter(adapter);  }  /\*\*  \* 排序文件列表  \*  \*/  **private** **void** sortFiles(File[] files) {  Log.*v*(*Name*, "level1 sortFiles");  **if** (sort == 1) { // time  Arrays.*sort*(files, **new** Comparator<File>() {  **public** **int** compare(File file1, File file2) {  **if** (file1.lastModified() > file2.lastModified()) {  **return** 1;  } **else** {  **return** -1;  }  }  });  } **else** **if** (sort == 2) { // size  Arrays.*sort*(files, **new** Comparator<File>() {  **public** **int** compare(File file1, File file2) {  **if** (file1.length() > file2.length()) {  **return** 1;  } **else** {  **return** -1;  }  }  });  } **else** **if** (sort == 3) { // name  Arrays.*sort*(files, **new** Comparator<File>() {  **public** **int** compare(File file1, File file2) {  **if** (file1.getName().length() > file2.getName().length()) {  **return** 1;  } **else** {  **return** -1;  }  }  });;  } **else** { // type  Arrays.*sort*(files, **new** Comparator<File>() {  **public** **int** compare(File file1, File file2) {  **if** (file1.isDirectory() && file2.isDirectory()) {  Log.*v*(*Name*,  "level2 file1.isDirectory() && file2.isDirectory()");  **return** 1;  }  **if** (file2.isDirectory()) {  Log.*v*(*Name*,  "level2 !file1.isDirectory() && file2.isDirectory()");  **return** 1;  }  **return** -1;  }  });  }  }  /\*\*  \* 打开文件  \*  \* **@param** file  \*/  **private** **void** openFile(File file) {  Log.*v*(*Name*, "level1 openFile");  Intent intent = **new** Intent();  intent.addFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  // 设置intent的Action属性  intent.setAction(Intent.*ACTION\_VIEW*);  // 获取文件file的MIME类型  String type = getMIMEType(file);  // 设置intent的data和Type属性。  intent.setDataAndType(Uri.*fromFile*(file), type);  // 跳转  startActivity(intent);  }  /\*\*  \* 根据文件后缀名获得对应的MIME类型。  \*  \* **@param** file  \*/  **private** String getMIMEType(File file) {  Log.*v*(*Name*, "level1 getMIMEType");  String type = "\*/\*";  String fName = file.getName();  // 获取后缀名前的分隔符"."在fName中的位置。  **int** dotIndex = fName.lastIndexOf(".");  **if** (dotIndex < 0) {  Log.*v*(*Name*, "level2 dotIndex<0");  **return** type;  }  /\* 获取文件的后缀名 \*/  String end = fName.substring(dotIndex, fName.length()).toLowerCase();  **if** (end == "") {  Log.*v*(*Name*, "level2 end == ");  **return** type;  }  // 在MIME和文件类型的匹配表中找到对应的MIME类型。  **for** (**int** i = 0; i < *MIME\_MapTable*.length; i++) {  **if** (end.equals(*MIME\_MapTable*[i][0]))  type = *MIME\_MapTable*[i][1];  }  **return** type;  }  @Override  **public** **void** onClick(View v) {  Log.*v*(*Name*, "level1 onClick");  **if** (v.getId() == R.id.*btnC*) {  rootDir = Environment.*getRootDirectory*();  loadFiles(rootDir);  } **else** **if** (v.getId() == R.id.*btnE*) {  **if** (Environment.*getExternalStorageState*().equals(  Environment.*MEDIA\_MOUNTED*)) {  rootDir = Environment.*getExternalStorageDirectory*();  loadFiles(rootDir);  }  }  }  @Override  **public** **void** onItemClick(AdapterView<?> parent, View view, **int** position,  **long** id) {  Log.*v*(*Name*, "level1 onItemClick");  // 获取文件（夹）的引用  Map<String, Object> map = (Map<String, Object>) parent  .getItemAtPosition(position);  **final** File file = (File) map.get("file");  // 如果是文件夹的话，则进入此文件夹，显示此文件夹的内容  **if** (file.isDirectory()) {  Log.*v*(*Name*, "level2 file.isDirectory");  **try** {  loadFiles(file);  } **catch** (Exception e) {// 若遇到权限不足的情况，则弹出警告  Toast.*makeText*(**this**, "权限不足", Toast.*LENGTH\_SHORT*).show();  }  } **else** {// 如过是文件，则选择相应应用打开此文件  openFile(file);  }  }  @Override  **public** **boolean** onItemLongClick(AdapterView<?> parent, View view,  **int** position, **long** id) {  Log.*v*(*Name*, "level1 onItemLongClick");  Map<String, Object> map = (Map<String, Object>) parent  .getItemAtPosition(position);  **final** File file = (File) map.get("file");  AlertDialog dialog = **new** AlertDialog.Builder(MainActivity.**this**)  .setTitle("操作")  .setItems(  **new** String[] { "复制", "剪切", "粘贴", "分享", "重命名", "删除",  "属性" }, **new** DialogInterface.OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog,  **int** which) {  **switch** (which) {  **case** 0: // 复制  Log.*v*(*Name*, "level2 copy");  copyPath = **new** File(file.getAbsolutePath());  flag = "copy";  **break**;  **case** 1: // 剪切  Log.*v*(*Name*, "level2 cut");  copyPath = **new** File(file.getAbsolutePath());  flag = "cut";  **break**;  **case** 2: // 粘贴  Log.*v*(*Name*, "level2 paste");  paste();  **break**;  **case** 3: // 分享  Log.*v*(*Name*, "level2 send");  share();  **break**;  **case** 4: // 重命名  Log.*v*(*Name*, "level2 newName");  LayoutInflater factory = LayoutInflater  .*from*(MainActivity.**this**);  **final** View view = factory.inflate(  R.layout.*rename*, **null**);  ((EditText) view.findViewById(R.id.*rename*))  .setText(file.getName());  AlertDialog d = **new** AlertDialog.Builder(  MainActivity.**this**)  .setCancelable(**true**)  .setMessage("新文件名")  .setView(view)  .setPositiveButton(  "确定",  **new** DialogInterface.OnClickListener() {  @Override  **public** **void** onClick(  DialogInterface dialog,  **int** which) {  String newName = ((EditText) view  .findViewById(R.id.*rename*))  .getText()  .toString();  String newFile = currentDir  .getText()  .toString()  + "/"  + newName;  **if** (**new** File(  newFile)  .exists()) {  Toast.*makeText*(  MainActivity.**this**,  "文件名重复",  Toast.*LENGTH\_LONG*)  .show();  **return**;  }  file.renameTo(**new** File(  newFile));  loadFiles(**new** File(  currentDir  .getText()  .toString()));  }  }).create();  d.show();  **break**;  **case** 5: // 删除  Log.*v*(*Name*, "level2 delete");  AlertDialog ad = **new** AlertDialog.Builder(  MainActivity.**this**)  .setMessage(  "确实要删除" + file.getName()  + "吗?")  .setCancelable(**true**)  .setPositiveButton(  "确定",  **new** DialogInterface.OnClickListener() {  @Override  **public** **void** onClick(  DialogInterface dialog,  **int** which) {  **if** (file.isFile()) {  Log.*v*(*Name*,  "level3 isFile");  file.delete();  } **else** {  Log.*v*(*Name*,  "level3 !isFile");  delete(file);  }  loadFiles(**new** File(  currentDir  .getText()  .toString()));  };  }).create();  ad.show();  **break**;  **case** 6: // 显示属性  String[] attr = **new** String[] { "1", "2",  "3", "4" };  attr[0] = "文件名：\n" + file.getName();  attr[1] = "大小："  + String.*valueOf*(getLength(file))  + "Byte";  attr[2] = "修改时间：\n"  + (**new** SimpleDateFormat(  "yyyy-MM-dd HH:mm:ss"))  .format(file.lastModified());  attr[3] = "绝对路径：\n"  + file.getAbsolutePath();  **new** AlertDialog.Builder(MainActivity.**this**)  .setTitle("属性信息")  .setItems(attr, **null**)  .setNegativeButton("确定", **null**)  .show();  **break**;  **default**:  **break**;  }  // Toast.makeText(MainActivity.this, "" +  // which, Toast.LENGTH\_LONG).show();  }  }).create();  dialog.show();  /\*  \* Intent data=new Intent(); data.putExtra("path",  \* file.getAbsolutePath()); this.setResult(RESULT\_OK,data);  \* this.finish();  \*/  **return** **true**;  }  **public** **void** paste() {  **final** String startPath = copyPath.getAbsolutePath();  **final** String desPath = currentDir.getText().toString() + "/"  + copyPath.getName();  File[] files = **new** File(currentDir.getText().toString()).listFiles();  **for** (File file : files) {  **if** (copyPath.getName().equals(file.getName())) {  Toast.*makeText*(MainActivity.**this**, "此文件/文件夹已存在",  Toast.*LENGTH\_SHORT*).show();  **return**;  }  }  **int** length = (**int**) (getLength(copyPath) / (1024));  progressDialog = **new** ProgressDialog(MainActivity.**this**);  progressDialog.setProgressStyle(ProgressDialog.*STYLE\_HORIZONTAL*);  progressDialog.setMessage("正在复制");  progressDialog.setMax(length);  progressDialog.setProgress(0);  progressDialog.setCancelable(**false**);  progressDialog.show();  **new** Thread() {  **public** **void** run() {  copy(startPath, desPath);  // 通过Message对象向原线程传递信息  Message message = Message.*obtain*();  messageHandler.sendMessage(message);  progressDialog.dismiss();  // 如果为剪切则删除对应文件/文件夹  **if** ("cut".equals(flag)) {  **if** (copyPath.isFile()) {  copyPath.delete();  } **else** {  delete(copyPath);  }  }  }  }.start();  }  **public** **void** share() {  **try** {  ShareSample sample = **new** ShareSample();  sample.getShareList("分享标题", "要分享的内容哦。呵呵", **this**);  } **catch** (Exception e) {  e.printStackTrace();  }  }  /\*\*  \* 返回删除文件后的空文件夹列表  \* \*/  **public** **void** delete(File delFile) {  Log.*v*(*Name*, "level1 delete");  **if** (delFile.isFile()) {  Log.*v*(*Name*, "level2 delFile.isFile()");  delFile.delete();  } **else** **if** (delFile.isDirectory()) {  Log.*v*(*Name*, "level2 delFile.isDirectory()");  File[] files = delFile.listFiles();  **for** (File file : files) {  **if** (file.isDirectory())  delete(file);  **else** {  file.delete();  }  }  delFile.delete();  }  **return**;  }  /\*\*  \* 获取文件/文件夹大小  \* \*/  **private** **long** getLength(File file) {  Log.*v*(*Name*, "level1 getLength");  totaLength = 0;  **if** (file.isDirectory()) {  Log.*v*(*Name*, "level2 file.isDirectory");  File[] files = file.listFiles();  **for** (File file2 : files) {  totaLength = totaLength + getLength(file2);  }  } **else** {  Log.*v*(*Name*, "level2 !file.isDirectory");  totaLength = totaLength + file.length();  }  **return** totaLength;  }  /\*\*  \* 复制功能，startFilePath为文件源路径，desFilePath为目的路径  \* \*/  **public** **boolean** copy(String startFilePath, String desFilePath) {  Log.*v*(*Name*, "level1 copy");  currentLen = 0;  totaLength = 0;  **this**.startFilePath = startFilePath;  **this**.desFilePath = desFilePath;  // 判断是否返回成功的变量  **boolean** copyFinished = **false**;  File startFile = **new** File(startFilePath);  File desFile = **new** File(desFilePath);  // 如果源文件是个文件  **if** (startFile.isFile()) {  Log.*v*(*Name*, "level2 startFIle.isFile");  copyFinished = **this**.copySingleFile(startFile, desFile);  // 如果源文件是个文件夹，就需要递归复制  } **else** {  Log.*v*(*Name*, "level2 !delFile.isFile()");  // 如果目标文件夹是源文件夹的一个子目录的情况，拒绝复制，因为会造成无限循环  **if** (desFilePath.startsWith(startFilePath)) {  Log.*v*(*Name*, "level3 startWith");  System.*out*.println("error copy");  **return** **false**;  } **else** {  Log.*v*(*Name*, "level3 !startWith");  copyFinished = **this**.copyFolder(startFile, desFile);  }  }  **return** copyFinished;  }  /\*\*  \* 复制单个文件，如果复制多个文件可以递归调用  \*/  **private** **boolean** copySingleFile(File startSingleFile, File desSingleFile) {  Log.*v*(*Name*, "level1 copySingleFile");  **boolean** rightCopy = **false**;  // -------从源文件中输入内存入byte中，在将byte写入目标文件--------------------  FileInputStream singleFileInputStream = **null**;  DataInputStream singleDataInputStream = **null**;  FileOutputStream singleFileOutputStream = **null**;  DataOutputStream singleDataOutputStream = **null**;  **try** {  singleFileInputStream = **new** FileInputStream(startSingleFile);  singleDataInputStream = **new** DataInputStream(  **new** BufferedInputStream(singleFileInputStream));  singleFileOutputStream = **new** FileOutputStream(desSingleFile);  singleDataOutputStream = **new** DataOutputStream(  **new** BufferedOutputStream(singleFileOutputStream));  **byte**[] b = **new** **byte**[1024];  **int** len = -1;  **while** ((len = singleDataInputStream.read(b)) != -1) {  currentLen = currentLen + len;  singleDataOutputStream.write(b, 0, len);  progressDialog.setProgress(currentLen / (1024));  }  // 刷新缓冲区  singleDataOutputStream.flush();  } **catch** (Exception e) {  e.printStackTrace();  } **finally** {  **try** {  **if** (singleDataInputStream != **null**)  singleDataInputStream.close();  **if** (singleDataOutputStream != **null**)  singleDataOutputStream.close();  } **catch** (Exception e) {  e.printStackTrace();  }  }  // 判断源文件和目标文件大小是否相同，如果相同证明复制成功  **if** (startSingleFile.length() == desSingleFile.length()) {  Log.*v*(*Name*, "level2 copy\_success");  rightCopy = **true**;  } **else** {  Log.*v*(*Name*, "level2 copy\_failed");  rightCopy = **false**;  }  **return** rightCopy;  }  /\*\*  \* 递归复制文件夹，因为文件夹下不止一个文件，里面可能有文件或文件夹， 因此需要调用递归方法  \*  \* **@param** startFolder  \* = 需要复制的文件夹  \* **@param** desFolder  \* = 复制目的地的文件夹  \* **@return** = true 表示成功，false 表示失败  \*/  **public** **boolean** copyFolder(File startFolder, File desFolder) {  Log.*v*(*Name*, "level1 copyFolder");  **boolean** rightCopy = **false**;  rightCopy = **this**.recursionCopy(startFolder, desFolder);  **return** rightCopy;  }  /\*\*  \* 复制文件夹函数，此函数是个递归，会复制文件夹下的所有文件  \*  \* **@param** recFileFolder  \* = 需要拷贝的文件夹或子文件夹  \* **@param** recDesFolder  \* = 拷贝的目的文件夹或子文件夹，  \* **@return** = true表示成功， false表示失败  \*/  **private** **boolean** recursionCopy(File recFileFolder, File recDesFolder) {  Log.*v*(*Name*, "level1 recursionCopy");  File desFolder = recDesFolder;  // 因为目的文件夹或子文件夹不存在，需要创建  desFolder.mkdir();  // 此为需要拷贝的文件夹下的所有文件列表（其中有文件和文件夹）  File[] files = recFileFolder.listFiles();  // 如果是个空文件夹  **if** (files.length == 0)  **return** **true**;  /\*  \* 将文件夹下所有文件放入for循环，如果是文件，那么调用copySingleFile()拷贝文件， 如果是文件夹，那么递归调用此函数  \*/  **for** (File thisFile : files) {  // 如果此文件是个文件，那么直接调用单个文件复制命令复制文件  **if** (thisFile.isFile()) {  // 得到此文件的新位置地址  String desContentFilePath = **this**.getSubFilePath(startFilePath,  desFilePath, thisFile.getAbsolutePath());  **boolean** rightCopy = **this**.copySingleFile(thisFile, **new** File(  desContentFilePath));  // 如果复制失败，就跳出循环停止复制  **if** (!rightCopy)  **return** **false**;  // 如果是个文件夹  } **else** {  /\*  \* 此函数是为了得到目的文件夹的地址， 如：源文件夹为：D:/yingzi/text (其中text文件夹下有另一个文件夹  \* second : D:/yingzi/text/second) 目标位置为：E:/level1/text  \* 那么此second文件夹在目标地址的位置就是 E:/level1/text/second  \*/  String desContentFilePath = **this**.getSubFilePath(startFilePath,  desFilePath, thisFile.getAbsolutePath());  // 递归的调用此函数，确保函数都被复制完全  **boolean** rightCopy = recursionCopy(thisFile, **new** File(  desContentFilePath));  **if** (!rightCopy)  **return** **false**;  }  }  **return** **true**;  }  /\*\*  \* 此函数是为了得到目的文件夹的地址， 如：源文件夹为：D:/yingzi/text (其中text文件夹下有另一个文件夹 second :  \* D:/yingzi/text/second) 目标位置为：E:/level1/text 那么此second文件夹在目标地址的位置就是  \* E:/level1/text/second 此方法中 startFolderPath = D:/yingzi/text (源文件夹) ；  \* desFolderPath = E:/level1/text (目标位置)； currentFilePath =  \* D:/yingzi/text/second(需要复制的子文件夹) 返回值为： E:/level1/text/second  \*/  **private** String getSubFilePath(String startFolderPath, String desFolderPath,  String currentFilePath) {  Log.*v*(*Name*, "level1 getSubFilePath");  String currentDesFilePath = **null**;  **int** i = startFolderPath.length();  // int j = desFolderPath.lastIndexOf("/");  // String subDirPath = startFolderPath.substring(0, i);  // String subDesPath = desFolderPath.substring(0, j);  currentDesFilePath = desFolderPath + "/"  + currentFilePath.substring(i + 1);  **return** currentDesFilePath;  }  **class** ShareSample {  /\*\*  \* 获得分享列表（简单实现，只是把核心的东西写了写。不是太实用）  \* \*\*/  **public** **void** getShareList(**final** String title, **final** String content,  **final** Activity mActivity) {  **final** Map<String, ResolveInfo> appInfo = **new** HashMap<String, ResolveInfo>();  List<ResolveInfo> appList = getShareTargets(mActivity);  **final** String[] items;  **if** (appList.size() > 0) {  **for** (**int** i = 0; i < appList.size(); i++) {  ResolveInfo tmp\_ri = (ResolveInfo) appList.get(i);  // ActivityInfo tmp\_ai=tmp\_ri.activityInfo;  ApplicationInfo apinfo = tmp\_ri.activityInfo.applicationInfo;  String tmp\_appName = apinfo.loadLabel(  mActivity.getPackageManager()).toString();  // 这里就做了一个简单的得到应用名称判断， 实际情况根据 ResolveInfo 这东东  // 可以得到很多很多有用的东东哦。  **if** (!tmp\_appName.equals("微博")) {  **continue**;  }  appInfo.put(tmp\_appName, tmp\_ri);  }  items = **new** String[appInfo.size()];  **int** j = 0;  **for** (Map.Entry<String, ResolveInfo> myEntry : appInfo  .entrySet()) {  items[j] = myEntry.getKey();  j++;  }  } **else** {  Toast.*makeText*(mActivity.getApplicationContext(), "暂无分享应用",  Toast.*LENGTH\_SHORT*).show();  **return**;  }  **if** (appInfo.size() < 1) {  Toast.*makeText*(mActivity.getApplicationContext(), "暂无分享应用",  Toast.*LENGTH\_SHORT*).show();  **return**;  }  /\*\*  \* 这里用对话框简单的一个实现，实际开发中应该要单独抽取出来封装成一个自定义的列表。  \*  \* \*\*/  **new** AlertDialog.Builder(mActivity)  .setTitle("选择分享")  .setSingleChoiceItems(items, 0,  **new** DialogInterface.OnClickListener() {  **public** **void** onClick(DialogInterface dialog,  **int** item) {  CreateShare(title, content, mActivity,  appInfo.get(items[item]));  // CreateShare(title,content,mActivity);  dialog.cancel();  }  }).show();// 显示对话框  }  /\*\*  \* 实现自定义分享功能(主要就是启动对应的App)  \* \*\*/  **private** **void** CreateShare(String title, String content,  Activity activity, ResolveInfo appInfo) {  **try** {  Intent shareIntent = **new** Intent(Intent.*ACTION\_SEND*);  shareIntent.setComponent(**new** ComponentName(  appInfo.activityInfo.packageName,  appInfo.activityInfo.name));  // 这里就是组织内容了，  // shareIntent.setType("text/plain");  shareIntent.setType("image/\*");  shareIntent.putExtra(Intent.*EXTRA\_TEXT*, content);  shareIntent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  activity.startActivity(shareIntent);  } **catch** (Exception e) {  e.printStackTrace();  // **TODO**: handle exception  }  }  /\*\*  \* 默认分享列表（网上一大堆）  \* \*\*/  **private** **void** CreateShare(String title, String content, Activity activity) {  Intent intent = **new** Intent(Intent.*ACTION\_SEND*);  intent.setType("image/\*");  intent.putExtra(Intent.*EXTRA\_SUBJECT*, title);  intent.putExtra(Intent.*EXTRA\_TEXT*, content);  intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  activity.startActivity(Intent.*createChooser*(intent, title));  }  /\*\*  \* 获得所有带Intent.ACTION\_SEND的应用列表。 ResolveInfo 这个东东真不错。  \* \*\*/  **private** List<ResolveInfo> getShareTargets(Activity activity) {  Intent intent = **new** Intent(Intent.*ACTION\_SEND*, **null**);  intent.addCategory(Intent.*CATEGORY\_DEFAULT*);  intent.setType("text/plain");  PackageManager pm = activity.getPackageManager();  **return** pm.queryIntentActivities(intent,  PackageManager.*COMPONENT\_ENABLED\_STATE\_DEFAULT*);  }  }  } |

## MIME.java

|  |
| --- |
| **package** com.mime;  **public** **class** MIME {  **public** **static** **final** String[][] *MIME\_MapTable*={  //{��׺��MIME����}  {".3gp", "video/3gpp"},  {".apk", "application/vnd.android.package-archive"},  {".asf", "video/x-ms-asf"},  {".avi", "video/x-msvideo"},  {".bin", "application/octet-stream"},  {".bmp", "image/bmp"},  {".c", "text/plain"},  {".class", "application/octet-stream"},  {".conf", "text/plain"},  {".cpp", "text/plain"},  {".doc", "application/msword"},  {".docx", "application/vnd.openxmlformats-officedocument.wordprocessingml.document"},  {".xls", "application/vnd.ms-excel"},  {".xlsx", "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"},  {".exe", "application/octet-stream"},  {".gif", "image/gif"},  {".gtar", "application/x-gtar"},  {".gz", "application/x-gzip"},  {".h", "text/plain"},  {".htm", "text/html"},  {".html", "text/html"},  {".jar", "application/java-archive"},  {".java", "text/plain"},  {".jpeg", "image/jpeg"},  {".jpg", "image/jpeg"},  {".js", "application/x-javascript"},  {".log", "text/plain"},  {".m3u", "audio/x-mpegurl"},  {".m4a", "audio/mp4a-latm"},  {".m4b", "audio/mp4a-latm"},  {".m4p", "audio/mp4a-latm"},  {".m4u", "video/vnd.mpegurl"},  {".m4v", "video/x-m4v"},  {".mkv", "video/mpeg"},  {".mov", "video/quicktime"},  {".mp2", "audio/x-mpeg"},  {".mp3", "audio/x-mpeg"},  {".mp4", "video/mp4"},  {".mpc", "application/vnd.mpohun.certificate"},  {".mpe", "video/mpeg"},  {".mpeg", "video/mpeg"},  {".mpg", "video/mpeg"},  {".mpg4", "video/mp4"},  {".mpga", "audio/mpeg"},  {".msg", "application/vnd.ms-outlook"},  {".ogg", "audio/ogg"},  {".pdf", "application/pdf"},  {".png", "image/png"},  {".pps", "application/vnd.ms-powerpoint"},  {".ppt", "application/vnd.ms-powerpoint"},  {".pptx", "application/vnd.openxmlformats-officedocument.presentationml.presentation"},  {".prop", "text/plain"},  {".rc", "text/plain"},  {".rmvb", "audio/x-pn-realaudio"},  {".rtf", "application/rtf"},  {".sh", "text/plain"},  {".tar", "application/x-tar"},  {".tgz", "application/x-compressed"},  {".txt", "text/plain"},  {".wav", "audio/x-wav"},  {".wma", "audio/x-ms-wma"},  {".wmv", "audio/x-ms-wmv"},  {".wps", "application/vnd.ms-works"},  {".xml", "text/plain"},  {".z", "application/x-compress"},  {".zip", "application/x-zip-compressed"},  {"", "\*/\*"}  };  } |

## AndroidMain.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*  package=*"com.file"*  android:versionCode=*"1"*  android:versionName=*"1.0"* >  <application  android:icon=*"@drawable/icon1"*  android:label=*"@string/app\_name"* >  <activity  android:name=*"com.explorer.MainActivity"*  android:label=*"@string/app\_name"* >  <intent-filter>  <action android:name=*"android.intent.action.MAIN"* />  <category android:name=*"android.intent.category.LAUNCHER"* />  </intent-filter>  </activity>  <activity  android:name=*".SharePage"*  android:label=*"分享到微博"* >  <intent-filter>  <action android:name=*"android.intent.action.SEND"* />  <category android:name=*"android.intent.category.DEFAULT"* />  <data android:mimeType=*"image/\*"* />  </intent-filter>  </activity>  </application>  </manifest> |

## string.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <string name=*"hello"*>Hello World, MainActivity!</string>  <string name=*"app\_name"*>文件浏览器</string>  <string name=*"c"*>手机内存</string>  <string name=*"e"*>储存卡</string>  <string name=*"newFile"*>新建文件夹</string>  <string name=*"open"*>打开</string>  <string name=*"exit"*>退出</string>  <string name=*"paste"*>粘贴</string>  <string name=*"sort"*>排序</string>  <string name=*"share"*>分享</string>  </resources> |