**文件相关操作**

文件相关操作

重命名，打开文件 删除文件 复制文件 (FileItemClickListener.java)

1. 操作打开文件的详细代码如下 ，传入string类型的文件名 用功能函数doOpenfile实现操作。

public void doOpenFile(String file){

/\*\*

Log.d(tag, "zip: " + file);

ZipUtils.zip(file, file + "2");

/\*\*/

/\*\*\*/

Intent intent = new Intent(Intent.ACTION\_VIEW);

Uri uri = Uri.parse("file://" + file);

String type = null;

type = MimeTypeMap.getSingleton().getMimeTypeFromExtension(

MimeTypeMap.getFileExtensionFromUrl(file));

if (type == null) {

String tmp = file.toLowerCase();

if (tmp.endsWith("mp3") || tmp.endsWith("wav") || tmp.endsWith("wma")) {

type = "audio/\*";

} else if (tmp.endsWith("apk")) {

type = "application/vnd.android.package-archive";

}

}

if(type != null){

//intent.setType(type);

if (FileManager.D) Log.d(tag, type);

intent.setDataAndType(uri, type);

try {

fileManager.startActivityForResult(intent, 1);

} catch (ActivityNotFoundException e){

Toast.makeText(fileManager,

fileManager.getString(R.string.can\_not\_open\_file),

Toast.LENGTH\_SHORT).show();

}

} else {

Toast.makeText(fileManager,

fileManager.getString(R.string.can\_not\_find\_a\_suitable\_program\_to\_open\_this\_file),

Toast.LENGTH\_SHORT).show();

}

/\*\* \*/

}

2删除文件操作代码如下：

private void doDelete(boolean select){

/\*\*/

BufferedReader br = null;

String fl = " ";

int ret = -1;

ArrayList<FileInfo> fis = fileManager.currentFileInfo();

ArrayList<String> selString = new ArrayList<String>();

ArrayList<Integer> fTmp = fileManager.selectedItem();

int size = fTmp.size();

for (int i = 0; i < size; i++){

selString.add(fis.get(fTmp.get(i)).path());

}

try {

if (fileManager.isRoot()){

if (fileManager.multFile && select) {

for (int i = 0; i < size; i++){

fl = fl + "\"" + selString.get(i) + "\" ";

}

} else {

fl = "\"" + absPath + "\"";

}

deleProgress = fileManager.linux.shell.exec("su");

DataOutputStream shell = new DataOutputStream(deleProgress.getOutputStream());

String cmd = "rm -r " + fl + "\n" + "exit\n";

//String cmd = "cp /sdcard/ji /data/xjf\nexit\n";

shell.write(cmd.getBytes());

shell.flush();

shell.close();

// end root

} else {

if (fileManager.multFile && select) {

selString.add(0, "rm");

selString.add(1, "-r");

deleProgress = fileManager.linux.shell.exec(

selString.toArray(new String[size + 2]));

} else {

deleProgress = fileManager.linux.delete(absPath);

}

}

br = new BufferedReader(

new InputStreamReader(deleProgress.getErrorStream()));

ret = deleProgress.waitFor();

if (ret != 0) {

//Log.d(tag, "Error(code = " + ret + "): " + br.readLine());

if (ret != 9)

xHandler.sendEmptyMessage(HANDLER\_SHOW\_DELETE\_ERROR);

else

Log.d(tag, "Error(code = " + ret + "): " + br.readLine());

}else;

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

// Ë¢ÐÂÎÄ¼þÁÐ±í

if (ret == 0) {

Message msg = fileManager.listViewHandler.obtainMessage();

msg.what = FileManager.HANDLER\_REFRESH\_LISTVIEW;

/\*\*

msg.arg1 = FileDialog.HANDLER\_SET\_LISTVIEW\_SELECTED;

msg.arg2 = position - 1;

/\*\*/

fileManager.listViewHandler.sendMessage(msg);

}

doDeleDialog.dismiss();

doDeleDialog = null;

if (deleProgress != null)

deleProgress.destroy();

deleProgress = null;

}

/\*\*/

}

3.重命名文件操作代码如下：

private void doRename(){

BufferedReader br = null;

java.lang.Process p = null;

try {

if (fileManager.isRoot() == false){

p = fileManager.linux.moveFile(absPath, renamePath);

} else {

// root

p = fileManager.linux.shell.exec("su");

DataOutputStream shell = new DataOutputStream(p.getOutputStream());

String cmd = "mv \"" + absPath + "\" \"" +

renamePath + "\"\nexit\n";

shell.write(cmd.getBytes());

shell.flush();

shell.close();

}

br = new BufferedReader(

new InputStreamReader(p.getErrorStream()));

if (p.waitFor() != 0) {

Toast.makeText(fileManager, br.readLine(),

Toast.LENGTH\_LONG).show();

//return;

}else{

Message msg = fileManager.listViewHandler.obtainMessage();

msg.what = FileManager.HANDLER\_REFRESH\_LISTVIEW;

/\*\*

msg.arg1 = FileDialog.HANDLER\_SET\_LISTVIEW\_SELECTED;

msg.arg2 = position;

/\*\*/

fileManager.listViewHandler.sendMessage(msg);

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

p.destroy();

}

}

4.复制文件操作（后有粘贴）

public void doMultCopy(){

multPaste.clear();

int size = fileManager.selectedItem().size();

for (int i = 0; i < size; i++){

multPaste.add(fileManager.currentFileInfo()

.get(fileManager.selectedItem().get(i)).path());

}

}

借助服务 copyservice实现复制功能（copyservice.java）

启动copy service 服务

void startCopyService(ArrayList<String> from,

String to, boolean cut){

pastePath = fileManager.currentPath();

if ( (pastePath.equals(FileManager.RW\_ROOT)

&& !fileManager.canWriteRoot() )

|| (pastePath.equals(FileManager.RW\_SYSTEM)

&& !fileManager.canWriteSystem())) {

Toast.makeText(fileManager, "µ±Ç°Ä¿Â¼²»ÄÜ½øÐÐÐ´²Ù×÷£¬ÕæÒªÐ´µÄ»°£¬µãÓÒÉÏ½ÇµÄro±äÎªrwºóÔÙÐ´",

Toast.LENGTH\_LONG).show();

return;

}

allDoLikeThis = false;

copyWarningSelection = -1;

copyFileService.isCut = cut;

copyFileService.root = fileManager.isRoot();

copyFileService.setFrom(from);

copyFileService.setToParentPath(to);

fileManager.startService(new Intent(fileManager, CopyFileService.class));

/\*\*/

}

在复制文件的service中 通过服务各周期来实现复制的一连串操作：

一：启动复制：

public void startCopy(ArrayList<String> from,

String toPath){

if (isCopying)

return;

File dFile = new File(toPath);

if (!root && !dFile.canWrite()) {

doFailure();

return;

}

String[] cmds = null;

if (!dFile.isDirectory())

return;

deleAfterCopy = isCut;

int ret = -1;

checkFile = true;

int count = from.size();

DataOutputStream shell = null;

BufferedReader err = null;

/\*\*

\* ÏÈÊÔÊÔÄÜ²»ÄÜÓÃmvÃüÁîÔËÐÐ

\* \*/

if (isCut) {

handler.sendEmptyMessage(FileItemClickListener.HANDLER\_SHOW\_CUT\_DIALOG);

tmpCutFiles = (ArrayList<String>) from.clone();

}

if (isCut && (!root || dFile.canWrite())) {

String fPath = null;

try {

File dst;

for (int i = 0; i < count; i++) {

fPath = from.get(i);

toFile = toPath + "/" + Common.getPathName(fPath);

dst = new File(toFile);

if (dst.exists()) {

ret = multFile(dst.getAbsolutePath());

if (ret == FileItemClickListener.COPY\_CANCEL) {

doCancel();

return;

}

if (ret == FileItemClickListener.COPY\_SKIP) {

tmpCutFiles.remove(fPath);

continue;

}

dst = new File(toFile);

}

if (toFile.contains(fPath))

continue;

cmds = new String[] { "mv", fPath, toFile };

moveProcess = linux.exec(cmds);

ret = moveProcess.waitFor();

if (ret != 0) {

if (!root && !dFile.canWrite()) {

doFailure();

return;

}

startDoCopy(from, toPath);

return;

}

}

// success

if (isCut) {

deleAfterCopy = false;

doSuccess();

return;

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

if (moveProcess != null)

moveProcess.destroy();

}

} else {

startDoCopy(from, toPath);

}

} private boolean checkFile = false;

实现复制：

private void startDoCopy(ArrayList<String> from,

String toPath){

int fSize = from.size();

copyLength = 0;

copyedLength = 0;

isCopying = true;

isCancel = false;

allDoSame = false;

selection = -1;

Message msg;

boolean result = false;

try {

// ¼ÆËãÎÄ¼þ×Ü´óÐ¡. toPath.contains("/sdcard")ÊÇÒòÎªÔÚ±ðµÄÄ¿Â¼ÓÐ¿ÉÄÜÓÐÈíÁ´½Ó,,java´¦Àí²»ÁË

if (toPath.contains("/sdcard")) {

for (int i = 0; i < fSize; i++) {

copyLength += FileOperation.getDirectorySize(from.get(i));

}

} else {

copyLength = 100;

}

// !cut

if (!deleAfterCopy) {

msg = handler

.obtainMessage(FileItemClickListener.HANDLER\_SHOW\_COPY\_PROGRESS\_DIALOG);

msg.arg1 = (int) copyLength;

perSize = ((int) copyLength / 100);

handler.sendMessage(msg);

}

int ret = 0;

//rootÓÃ»§¸´ÖÆ !canWrite &&

if (root) {

String fsPerm = fileManager.getCurrentDirPerm();

if (fsPerm != null && fsPerm.equals(FileManager.Mounts.RO)) {

Log.d(tag, "file system read only");

return;

}

///

rootProcess = fileManager.linux.shell.exec("su");

rootOS = new DataOutputStream(rootProcess.getOutputStream());

rootEBR = new BufferedReader(new InputStreamReader(rootProcess.getErrorStream()));

} // root

// ¶àÎÄ¼þ´¦ÀíÐèÒª,,Í³Ò»´¦Àí

for (int i = 0; i < fSize; i++) {

String fr = from.get(i);

toFile = toPath + "/" + Common.getPathName(fr);

File fd = new File(toFile);

if (checkFile) {

if (fr.equals(toFile)) {

toFile = Common.pathNameAppend(toFile, "(2)");

while ((fd = new File(toFile)).exists()) {

toFile = Common.pathNameAppend(toFile, "(2)");

}

}

}

File fs = new File(fr);

// ¸´ÖÆ¶Ô»°¿òÏÔÊ¾ µ±Ç°²Ù×÷ÎÄ¼þ

if (!isCut) {

msg = handler.obtainMessage(FileItemClickListener.HANDLER\_PROCESS\_SET\_MESSAGE);

Bundle b = new Bundle();

b.putString(FileItemClickListener.BUNDLE\_FROM\_PATH, fr);

b.putString(FileItemClickListener.BUNDLE\_TO\_PATH, toFile);

msg.setData(b);

handler.sendMessage(msg);

}

//!canWrite && root

if (root) {

String rootCmd = "";

if (fd.exists()) {

ret = multFile(fd.getAbsolutePath());

if (ret == FileItemClickListener.COPY\_CANCEL) {

doCancel();

return;

}

if (ret == FileItemClickListener.COPY\_SKIP) {

if (isCut) {

tmpCutFiles.remove(fr);

}

continue;

}

fd = new File(toFile);

}

/\*\*/

if (!isCut) {

rootCmd = "cp -fpr \"" + fs.getAbsolutePath() + "\" \""

+ fd.getAbsolutePath()

+ "\"\n";

} else {

rootCmd = "mv -r \"" + fs.getAbsolutePath() + "\" \""

+ fd.getAbsolutePath()

+ "\"\n";

}

if (FileManager.D) Log.d(tag, rootCmd);

rootOS.write(rootCmd.getBytes());

if (rootEBR.ready()) {

Log.e(tag, "rootEBR: " + rootEBR.readLine());

// return;

}

/\*\*\*/

} else {

// no root

if (copyFile(fs, fd) == FileItemClickListener.COPY\_CANCEL) {

isCancel = true;

doCancel();

return;

}

handler.sendEmptyMessage(FileItemClickListener.HANDLER\_REFRESH\_LIST);

}

/\*\*/

} // for ()

result = true;

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

//Log.d(tag, "finally");

if (rootProcess != null) {

try {

rootOS.writeBytes("exit\n");

rootProcess.waitFor();

if (FileManager.D) {

Log.d(tag, " s: " + rootOS.size());

Log.d(tag, " " + rootProcess.waitFor());

}

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

rootProcess.destroy();

}

rootProcess = null;

rootEBR = null;

if (result) {

doSuccess();

} else {

doFailure();

}

// Log.d(tag, "finish");

}

} // startDoCopy

复制文件：

private int copyFile(File src, File dst) throws InterruptedException {

doIsFile = false;

int ret = SUCCESS;

Message msg;

try {

if (!root && dst.exists() && checkFile) {

ret = multFile(dst.getAbsolutePath());

if (ret == FileItemClickListener.COPY\_CANCEL) {

return FileItemClickListener.COPY\_CANCEL;

}

if (ret == FileItemClickListener.COPY\_SKIP) {

return FileItemClickListener.COPY\_SKIP;

}

dst = new File(toFile);

}

if (src.isFile()) {

// ÎÄ¼þ

doIsFile = true;

fIn = new FileInputStream(src);

fOut = new FileOutputStream(dst);

in = new BufferedInputStream(fIn, BUFF\_SIZE);

out = new BufferedOutputStream(fOut, BUFF\_SIZE);

byte[] bytes = new byte[BUFF\_SIZE];

int length;

int tmpSize = 0;

while ((length = in.read(bytes)) != -1) {

out.write(bytes, 0, length);

copyedLength += length;

tmpSize += length;

/\*\*/

if (isCancel) {

dst.delete();

return (ret = FileItemClickListener.COPY\_CANCEL);

}

if (!isHidden && (tmpSize >= perSize) && !deleAfterCopy) {

msg = handler.obtainMessage(

FileItemClickListener

.HANDLER\_INCREMENT\_COPY\_PROGRESS);

msg.arg1 = tmpSize;

tmpSize = 0;

handler.sendMessage(msg);

}

/\*\*/

}

out.flush();

} else {

// ÎÄ¼þ¼Ð

//Log.d(tag, "folder");

if (toFile.contains(src.getAbsolutePath()))

return FileItemClickListener.COPY\_SKIP;

dst.mkdirs();

File[] fFiles = src.listFiles();

int count = fFiles.length;

for (int i = 0; i < count; i++) {

ret = copyFile(fFiles[i], new File(dst.getAbsoluteFile() + "/"

+ fFiles[i].getName()));

if ( ret == FileItemClickListener.COPY\_CANCEL) {

return FileItemClickListener.COPY\_CANCEL;

}

}

}

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

if (!doIsFile)

return ret;

try {

if (fIn != null)

fIn.close();

if (fOut != null)

fOut.close();

return ret;

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return ret;

}

二：周期内的其他操作：包括撤销复制 复制进程结束 杀死等

具体代码如下：

结束：

private void doFinish(boolean success) {

isCopying = false;

allDoSame = false;

if (destroyAfterCopy)

this.stopSelf();

from = null;

toParentPath = null;

moveProcess = null;

if (success && !isCut) {

handler.sendEmptyMessage(FileItemClickListener.HANDLER\_COPY\_FINISHED);

} else if (success){

handler.sendEmptyMessage(FileItemClickListener.HANDLER\_CUT\_FINISH);

}

}

失败：

private void doFailure(){

doFinish(false);

handler.sendEmptyMessage(

FileItemClickListener.HANDLER\_COPY\_FAILURE);

}

撤销：

private void doCancel(){

doFinish(false);

handler.sendEmptyMessage(

FileItemClickListener.HANDLER\_COPY\_CANCEL);

}

成功：

private void doSuccess() {

if (deleAfterCopy){

doDelete();

}

doFinish(true);

try {

if (out != null) {

out.close();

}

if (in != null)

in.close();

} catch (IOException e) {

e.printStackTrace();

}

}

public String pasteToPath = "";

public int selection = -1;

public boolean allDoSame = false;

public boolean isHidden = false;

public boolean successOrCancel() { return isCancel;}

public void setHidden(boolean b) {

isHidden = b;

}

粘贴：

public void doPaste() {

if (multPaste.isEmpty()) {

// Toast.makeText(FileDialog.this, "Clipboard is empty",

// Toast.LENGTH\_SHORT).show();

Message msg = fileManager.listViewHandler.obtainMessage();

msg.what = FileManager.HANDLER\_CLIP\_BOARD\_EMPTY;

fileManager.listViewHandler.sendMessage(msg);

isCopying = false;

return;

}