中山大学移动信息工程学院本科生实验报告

年级	1501	专业（方向）	移动（互联网）
学号	15352015	姓名	曹广杰
电话	13727022190	Email	1553118845@qq.com

Content

一、实验题目

SharedPreferences与File实现数据储存

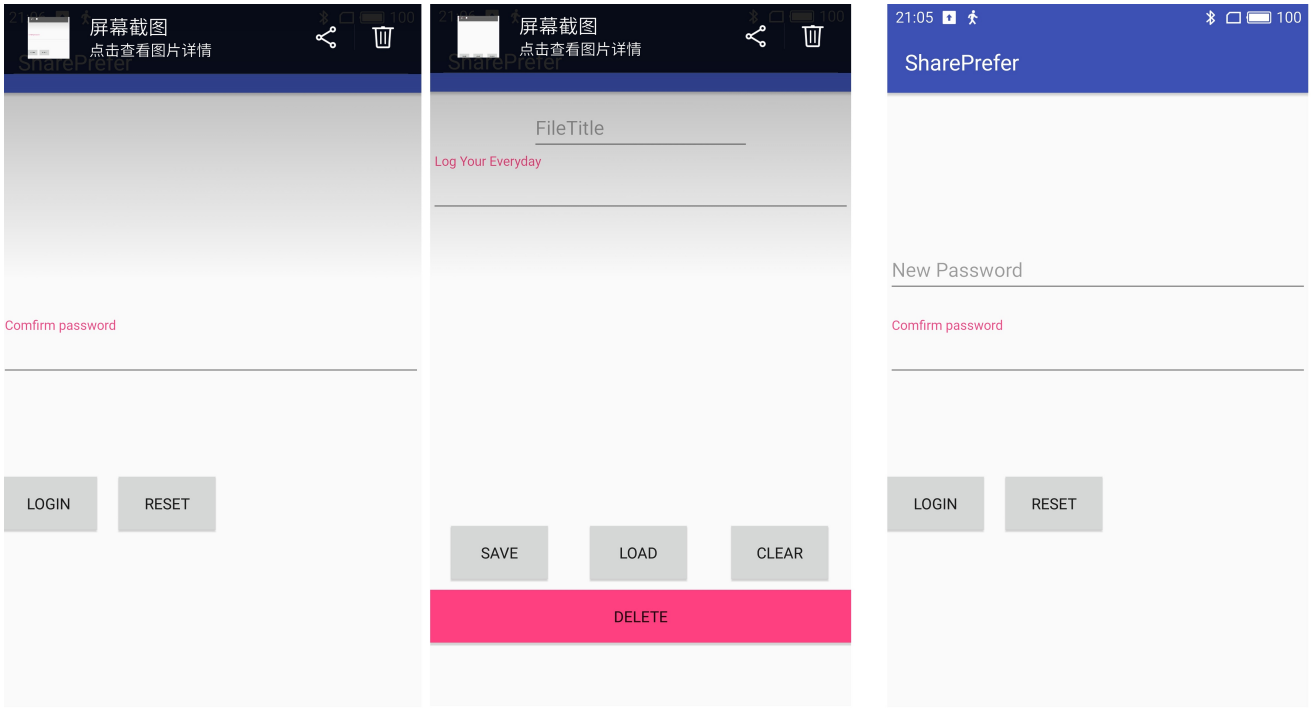
二、实现内容

本次实验模拟实现一个登录界面。

- 学习 SharedPreferences的基本使用
- 学习Android中常见的文件操作方法
- 复习 Android 界面编程

三、课堂实验结果

实验截图



实验步骤以及关键代码

实现登录界面的时候，需要完成对于密码界面的编程以及对于文件编写界面的编程。

设置密码界面

设置密码界面需要使用按钮，编辑框等组件，这里编辑框的使用笔者使用了文本输入布局，使得布局界面更加简洁：

```
<android.support.design.widget.TextInputLayout
    android:id="@+id/name">

    <EditText
        android:inputType="number" />
</android.support.design.widget.TextInputLayout>
```

SharedPreferences涉及到文件储存的部分，使用方法如下：

1. 初始化

   对该组件进行初始化的时候，需要为这个组件起一个名字，以便后来的查找与读取。同时还需要对这个储存单位的权限进行限定，以保证不会被其他的程序访问到。

   所以，就有两个概念：名字与权限。

   将其作为参数添加到初始化函数中即可：

   ```
   public static final String PREFRENCE_NAME = "PwdSave";
   public static int MODE = MODE_PRIVATE;
   SharedPreferences sharedPreferences =
               getSharedPreferences(PREFRENCE_NAME, MODE);
   ```

2. 数据写入

完成初始化操作之后，需要将编辑框中用户输入的密码嘻信息读取到该储存单元中，以便后续的使用。这里需要使用editor作为实现SharedPreferences的具有编辑功能的代理，而editor也需要初始化，该初始化操作是建立在prefer组件初始化的前提上的——毕竟editor是其代理：

```
EditText pass = NameBox.getEditText();
SharedPreferences.Editor editor = sharedPreferences.edit();
```

editor的赋值。在初始化之后，该组件可以正式接受赋值的操作，以实现信息的储存：

```
editor.putString("password", pass.getText().toString());
editor.commit();
```

函数`putString`就是用于传入字符串的函数，第二个参数就是将要储存的字符串，而第一个参数是该字符串的ID，用于查询和修改该字符串。完成了赋值操作之后，就可以提交该储存单元了，系统会将其储存到一个类似于.ini文件的轻量级文件里，以备后续的调用。

3. 数据读取

   前文说过，对于ShredPreferences的编辑代理来说，传入数据使用的函数是`putString`，那么对应的读取该参数使用的函数就应该是`getString`，读取参数的Demo如下：

   ```
   String key = sharedPreferences.getString("password", "NoKey");
   ```

   笔者希望将文件中的数据读取到名为`key`的String中，因此使用函数`getString`，其第一个参数就是希望读取的字符串的ID，程序就是根据其锁定希望查找的字符串。第二个参数是缺省值，以备查询失败的情况下，为当前的左值赋值。

4. 再次登录的检验密码界面

   再次登录的时候，笔者希望只要一个输出密码的输入框，而不是多一个输入框用于检验。爱中情况下需要隐藏一个组件，隐藏组件的函数很简单：`setVisibility`；但是困难的是知道何时应该隐藏多余的输入框——当然是在密码已经设置的时候，所以只要程序检查到当前是有密码的，就会隐藏一个多余的输入框：

   ```
   String key = sharedPreferences.getString("password", "NoKey");
   if (!key.equals("NoKey")){
       NameBox.setVisibility(View.INVISIBLE);
   }
   ```

编辑文件界面

在登录时候密码与之前设置的密码符合的时候，就应该允许用户进入当前的编辑界面。这里笔者使用Intent实现界面Activity之间的跳转：

```
Intent intent = new Intent(MainActivity.this, EditActivity.class);
startActivity(intent);
```

进入另一个界面之后，对于输入框的设置依然遵循之前对于密码框的设置操作，比较新颖的技术是对于文件储存方式的编写。

笔者希望在编辑界面编辑的数据可以传递到文件中，以便后续的读取，并允许长期保存。这就需要一个文件读写系统，这里笔者使用FileOutputStream作为实现的文件读写系统：

1. 初始化

```
String fname = "*.txt";
FileOutputStream fos =
        openFileOutput(fname, Context.MODE_PRIVATE);
```

这里的output表示从系统内存向文件中输出，第一个参数是输出文件的名字——这个名字对于之后的查询以及删除至关重要。如果没有这个文件，则程序就会创建一个名为fname中字符串的文件，第二个参数表示这个文件的读写权限。这里private表示的就是只有创建文件的程序才可以读写文件。

2. 写文件

```
String ftxt = "YoYoYo~";
fos.write(ftxt.getBytes());
fos.flush();
fos.close();
```

传入的参数不能是单纯的字符串，需要将其转换为Byte，便于文件的储存以及权限的设置。写入之后还需要调用`flush`函数，这是因为写数据的时候，都是先写到缓存中，待到缓存中的数据填满之后才会写入文件——那么当数据量太少的时候，写入操作就可能会失败。这种情况下，使用`flush`表明将后续的所有数据都一次写入文件中。写完之后就关闭文件，这是一个好习惯~

3. 加载文件

加载文件也是使用这个数据结构，只不过调用不同的接口。流语言的实现就是这样方便，具有系统性，这也是面向对象编程的优势。

调用的接口这里使用`openFileInput`，参数为文件的名字：

```
FileInputStream fis = openFileInput(fname);
byte[] tmp = new byte[fis.available()];
while ((len = fis.read(tmp)) > 0){
    ftxt.append(new String(tmp, 0, len));
}
fis.close();
```

因为之前储存到文件中的数据都是转化为Byte的——便于写入以及加密，这里读取的时候也需要将数据从Byte转化为字符串。

代码中的tmp是开辟的一个Byte的动态数组，恰好可以储存文件中的数据，后面的循环结构是遍历这里的Byte，将每一个byte转为字符串，并将字符串补充道ftxt的后面，这里的函数append是拓展填充的意思。

得到字符串之后，对EditTextView中的字符串使用setText函数就可以修改其文字了：

```
editmain.getEditText().setText(ftxt);
```

4. 删除文件

删除文件的接口为：

```
deleteFile(fname);
```

系统会根据文件的名称，对储存目录下的文件进行删除操作。

跳转界面

按返回键的时候跳转到主界面而不是上一层，这里笔者使用的是终止上一次的Activity，即在进入编辑界面之后，直接对MainActivity进行finish。

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.edit_main);
    MainActivity.main.finish();
}
```

其实常规的操作应该是清空Activity栈，使用`setFlag`以及在`Manifest.xml`中对于当前布局使用`noHistory=";true";`，进入当前界面则清空之前的所有Activity，那么在按下返回键的时候，也就不会找到上一次的Activity的界面了。

实验遇到困难以及解决思路

笔者在第一版的App中，进入编辑界面会闪退。

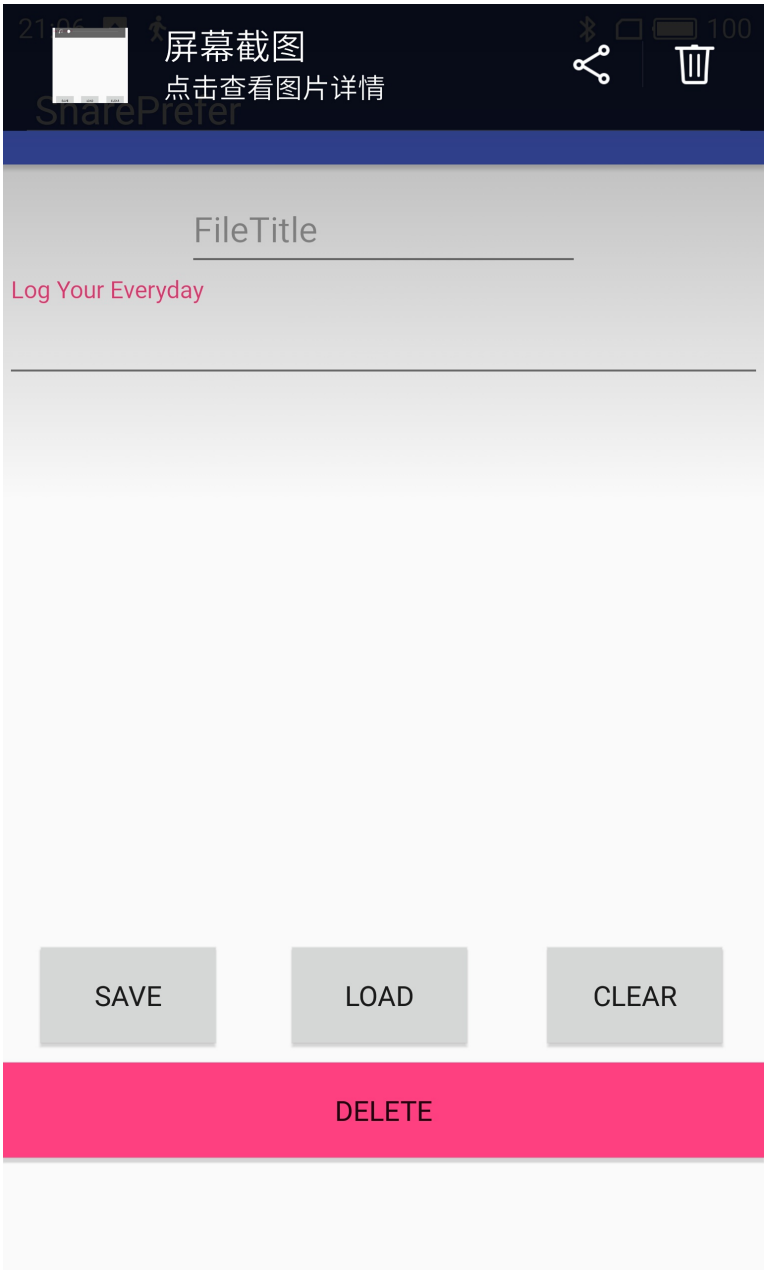
- 笔者在检查了所有代码之后（包括Intent以及跳转后界面的初始化代码），又使用debug工具实现逐行运行，但是总是在跳转后界面的OnCreate函数运行之前就会崩溃。
- 这就意味着不是代码实现的问题了，闪退往往是因为内存错误，因此猜测可能是关联文件的问题，笔者于是在`Manifest.xml`中添加了对于Activity的声明，问题解决。

在第二版的App中，添加初始化函数会闪退

- 先使用debug工具逐行运行，发现跳转目的界面的OnCreate函数可以运行，这就说明初始化函数没有问题。
- 继续设置断点，发现是赋值的错误——误将`TextInputLayout`赋值为`TextInoutEditText`；奇怪的是竟然没有报错，只是单纯地闪退，笔者猜测可能是使用了`findviewbyid`接口的原因。

四、课后实验结果

从编辑界面按返回键会直接回到主界面，而不是上一层：



在另一个界面调用另一个界面的finish函数可以强制终止。

五、实验思考及感想

本次实验的主要任务是完成文件储存系统的实现。

从具体实现上来讲，本次实现并不是很难，概念上也不是非常难理解，但是笔者有一些的思考。笔者发现Android的开发，到最后都是对于接口的使用，尽管如此，接口已经是千奇百怪数不胜数了。实话讲，笔者没有十足的把握可以掌握所有的接口用法，而笔者使用Android也不是为了陶冶情操，Android的具体意义就是作为一种工具使用的，如果现在说一个工具10年还不能完全掌握，那么笔者可以非常负责任地说，这款工具的开发可以说是非常失败的。但是笔者又毫无办法，心存不满却又没有建设性的意见。只能尽可能记录下每一个使用到的接口，以备自己与他人查询。

与其到各处blog上查询某种组件的用法，真的不如看实验文档，老师的文档好于助教的，助教的解释优于民间blog的。笔者发现老师的课件尽管只是使用很少量的代码，但是通过一个架构的使用，让使用顺序非常清晰——每一个组件、每一个布局、每一个接口、每一个接口使用的位置，解释地都非常清晰。这里真的是要感谢老师，老师的课件一定要珍藏一份，做的非常用心。