中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称：移动应用开发　　任课教师： 郑贵锋

年级	大三	专业（方向）	移动互联网
学号	15352408	姓名	张镓伟
电话	13531810182	Email	709075442@qq.com
开始日期	2017.12.8	完成日期	2017.12.9

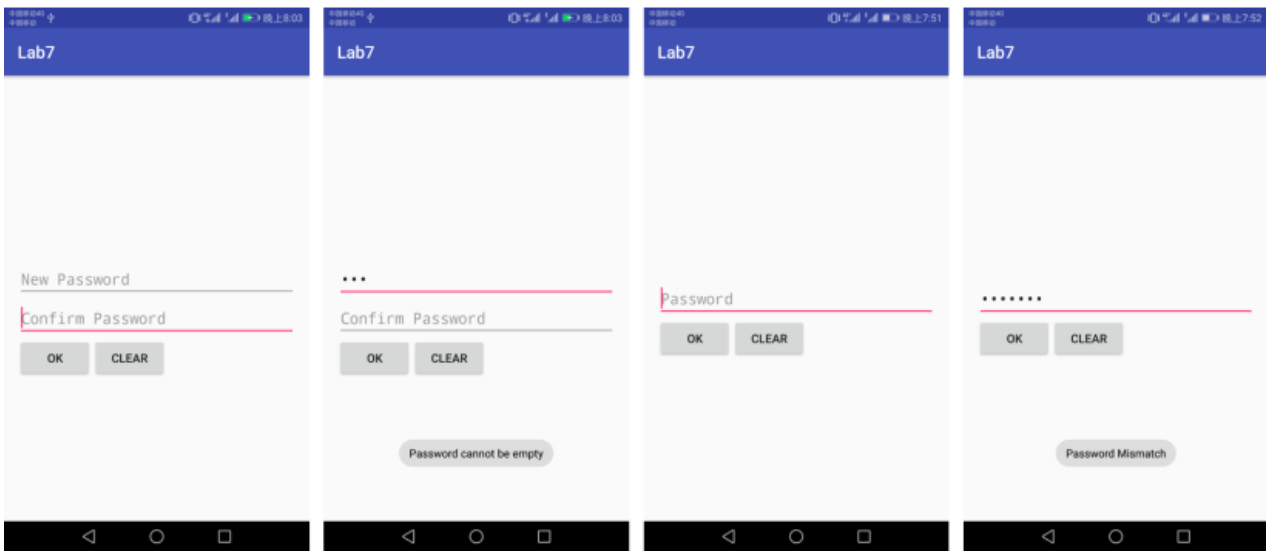
一、 实验题目

数据存储（一）

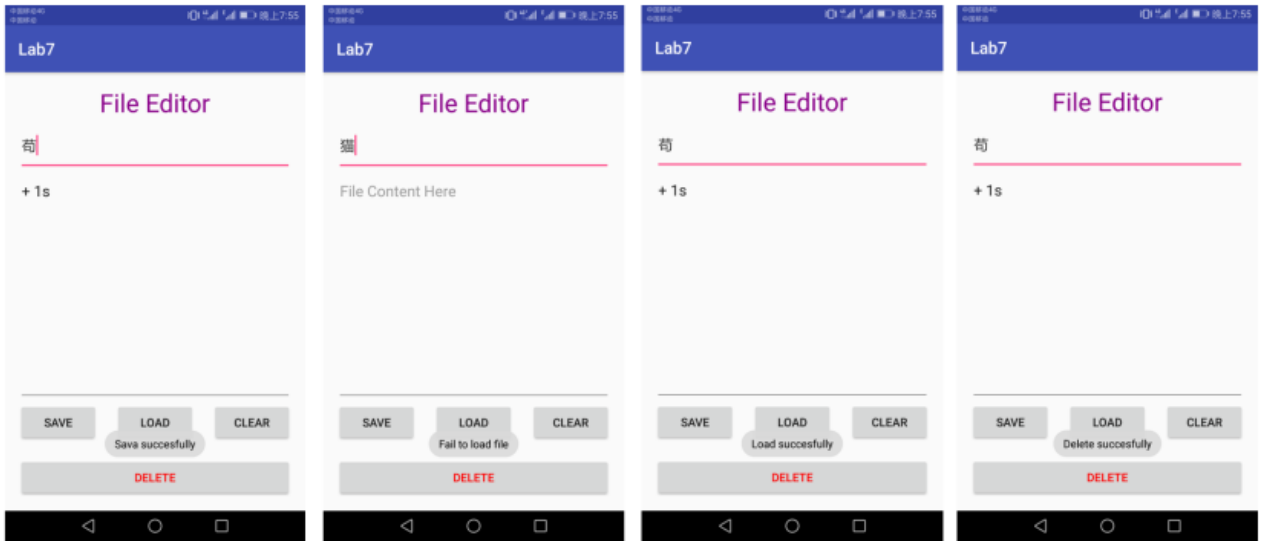
二、 实验目的

1. 学习 SharedPreferences 的基本使用。
2. 学习 Android 中常见的文件操作方法。
3. 复习 Android 界面编程。

三、 实验内容



从左至右， 依次为：初始密码界面、密码为空提示、 密码匹配后重新进入界面、 密码错误提示。



从左至右，依次为：保存成功提示、写入失败提示、 写入成功提示、删除成功提示。

1、 如图所示，本次实验需要实现两个 activity；

2、 首先，需要实现一个密码输入 activity：

a、 如果应用首次启动，则界面呈现出两个输入框，分别为新密码输入和确认密码输入框；

b、 输入框下方有两个按钮：

- OK 按钮，点击之后：
  - 若 new password 为空，则弹出密码为空的提示；
  - 若 new password 与 comfirm password 不匹配，则弹出不匹配的提示；
  - 若密码不为空且互相匹配，则保存密码，进入文件编辑界面。
- CLEAR 按钮，点击之后清除所有输入框的内容。

c、 完成创建密码后，退出应用再进入应用，则只呈现一个密码输入框；

- 点击 OK 按钮后，如果输入的密码与保存的密码不匹配，则弹出 Toast 提示；
- 点击 CLEAR 按钮后，清除密码输入框的内容。

d、 出于学习的目的，我们使用 SharedPreferences 来保存密码，但是在实际应用中我们会用更加安全的机制来保存这些隐私信息， 更多可以参考链接一和链接二

3、 然后，实现一个文件编辑 activity：

a、 界面底部有两行四个按钮， 第一行三个按钮高度一致，顶对齐，按钮水平均匀分布。按钮上方除了 ActionBar 和 StatusBar 之外的空间由标题和两个 EditText 占据， 文件内容编辑的 EditText 需要占据除去其他控件的全部屏幕空间，且内部文字竖直方向置顶，左对齐；

b、 在文件名输入框内输入文件名， 在文件内容编辑区域输入任意内容， 点击 SAVE 按钮后能够保存到指定文件，成功保存后弹出 Toast 提示；

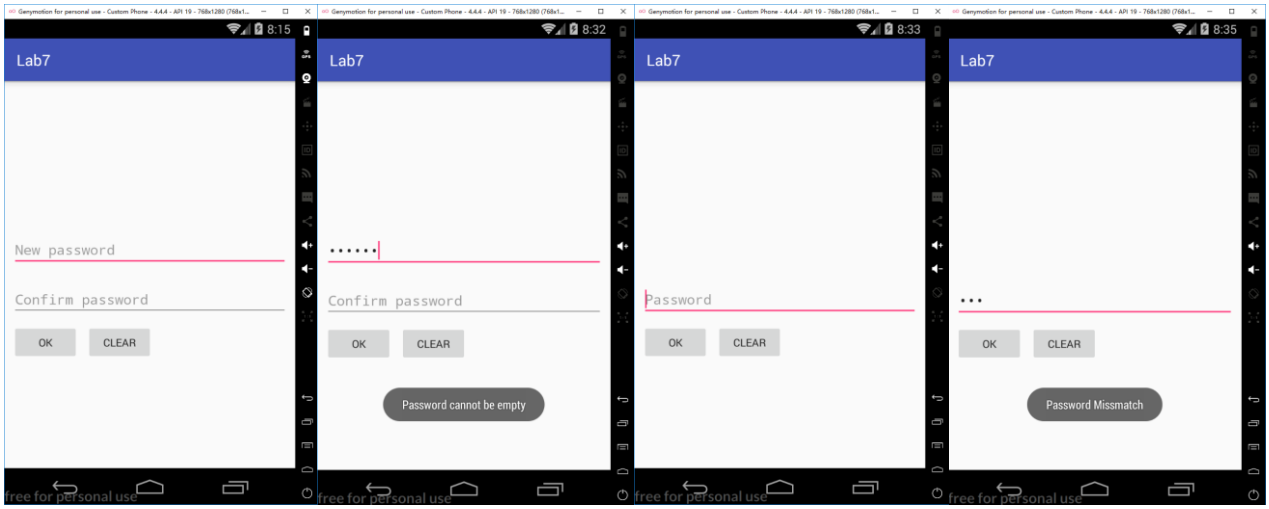
c、 点击 CLEAR 按钮，能够清空文件内容编辑区域内的内容；

d、 点击 LOAD 按钮，能够按照文件名从内存中读取文件内容，并将文件内容写入到编辑框中。如果成功导入，则弹出成功的 Toast 提示， 如果导入失败（例如：文件不在）， 则弹出读取失败的 Toast 提示。

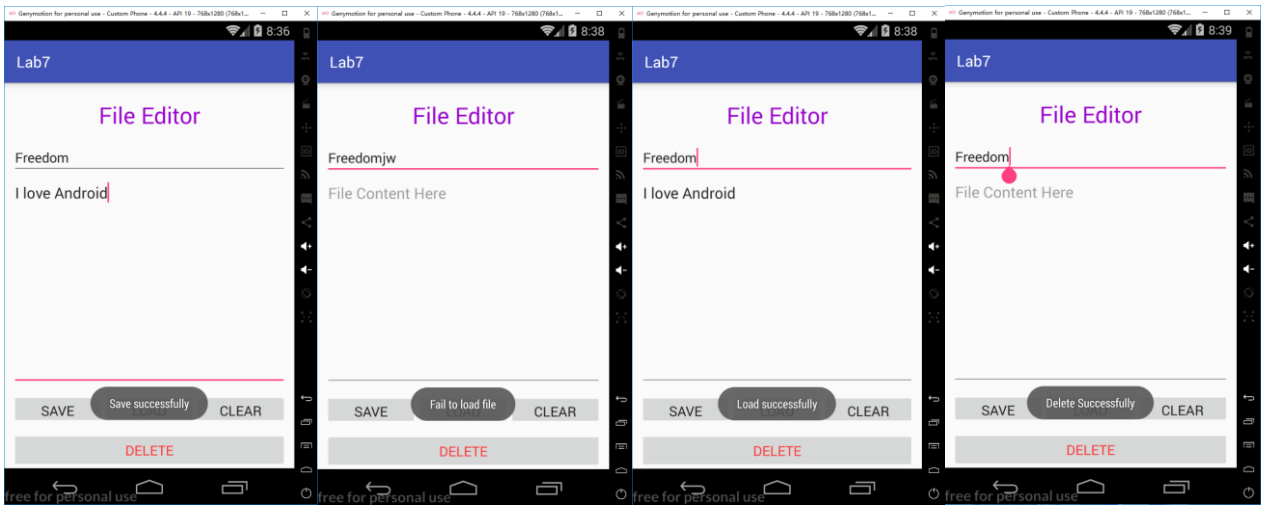
e、 点击 DELETE 按钮，能够按照文件名从内容中删除文件，删除文件后再载入文件，弹出导入失败的 Toast 提示。

四、 课堂实验结果

（1） 实验截图



从左至右， 依次为：初始密码界面、密码为空提示、 密码匹配后重新进入界面、 密码错误提示。



从左至右，依次为：保存成功提示、写入失败提示、 写入成功提示、删除成功提示。

（2） 实验步骤以及关键代码

1.XML 布局代码

起始界面布局：

这次界面的布局比较简单，用两个 EditText 实现密码输入框和密码确认框。两个按钮使用了 LinerLayout 确保在同一水平线上。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.example.freedom.lab7.MainActivity">

    <EditText
        android:id="@+id/text1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="200sp"
        android:layout_marginLeft="10sp"
        android:layout_marginRight="10sp"
        android:textSize="18sp"
        android:inputType="textPassword"
        android:hint="New password" />
```

```
        <EditText
            android:id="@+id/text2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20sp"
            android:layout_marginLeft="10sp"
            android:layout_marginRight="10sp"
            android:textSize="18sp"
            android:inputType="textPassword"
            android:hint="Confirm password" />
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <Button
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="10sp"
                android:layout_marginLeft="10sp"
                android:text="OK"/>
            <Button
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="10sp"
                android:layout_marginLeft="10sp"
                android:text="CLEAR"/>
        </LinearLayout>
</LinearLayout>
```

File Editor 界面布局：

先使用一个大的 LinearLayout，这个线性布局是按照垂直布局方向布局的，从上到下依次是一个 TextView 显示标题“File Editor”；两个 EditText 分别是文件名、文件内容、再通过一个横向的 LinerLayout 控制三个按钮在同一水平线上均匀分布，最后再一个 delete 按钮。然后可以给每一个部件设一个 weight，使得文件内容可以占据除了其他控件之外的整个地方。设 weight 的原理就是将空间按百分比分配给各个控件。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="10sp"
    android:layout_marginLeft="10sp"
    android:layout_marginRight="10sp"
    android:orientation="vertical"
    tools:context="com.example.freedom.lab7.FileEdit">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:text="TextView"
        android:textSize="30dp"
        android:textColor="@color/title"
        android:gravity="center"
        tools:text="File Editor"
        android:layout_weight="0.11" />

    <EditText
        android:id="@+id/filename"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="1sp"
        android:hint="File Name Here"
        android:textSize="18sp" />
```

```
    <EditText
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="5.5"
        android:textSize="20sp"
        android:gravity="top|left"
        android:hint="File Content Here" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="horizontal"
        android:layout_weight="1">
        <Button
            android:id="@+id/save"
            android:layout_marginTop="10sp"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="SAVE"
            android:textSize="18sp"/>
        <Button
            android:id="@+id/load"
            android:layout_marginTop="10sp"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="LOAD"
            android:textSize="18sp"/>
        <Button
            android:id="@+id/clear"
            android:layout_marginTop="10sp"
            android:layout_width="0dp"
            android:layout_weight="1"
            android:layout_height="wrap_content"
            android:text="CLEAR"
            android:textSize="18sp"/>

    </LinearLayout>
    <Button
        android:id="@+id/delete"
        android:layout_marginTop="10sp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="DELETE"
        android:textColor="@color/red"
        android:textSize="18sp"/>
</LinearLayout>
```

2.JAVA 代码

登录界面部分

先绑定控件：

```
final EditText text1 = (EditText) findViewById(R.id.text1);
final EditText text2 = (EditText) findViewById(R.id.text2);
Button button1 = (Button) findViewById(R.id.button1);
Button button2 = (Button) findViewById(R.id.button2);
final SharedPreferences sp = getSharedPreferences("MyText", MODE_PRIVATE);
final SharedPreferences.Editor editor = sp.edit();
boolean ischecked = sp.getBoolean("checked",false);
```

登录界面的密码我们采用的是 SharedPreferences 来完成，这个方法会保存一个键值对，我默认存储了一个键值对（“checked”，false），这个用来判断是否已经设置过密码了，默

认是 false 就是没设置，否则是 true。然后密码对是（“password”，密码）。一开始是先取出 SharedPreferences 存储的 checked 的值存到 ischecked 中，其他功能如下：

清空按钮：将 filename 和 file content 两个 EditText 的内容清空，通过 setText(“”)完成。

```
button2.setOnClickListener(new View.OnClickListener() {//清空按钮
    @Override
    public void onClick(View v) {
        text1.setText("");
        text2.setText("");
    }
});
```

ischecked=false: 说明此时没有设置密码。

设置 OK 按钮的点击事件：按照文档要求判断输入是否合法，不合法弹出相应的 Toast 提示。不然就通过 ShardPreferences 写入要存储的密码对，并且更新 checked 的值为 true，同时切换到文件编辑的 activity

ischecked=true:说明此时已经设置了密码。

先通过 setVisibility(View.INVISIBLE)将原来第一个输密码的框设置为不可见。

设置 OK 按钮的点击事件：判断第二个密码框输入的密码是否与 ShardPreferences 存储的密码一致，一致则切换到文件编辑的 activity

```
if(!ischecked) {
    button1.setOnClickListener((v) -> {
            if (TextUtils.isEmpty(text1.getText().toString()) ||TextUtils.isEmpty(text2.getText().toString()) ) {
                Toast.makeText(MainActivity.this, "Password cannot be empty", Toast.LENGTH_SHORT).show();
            } else if (!text1.getText().toString().equals(text2.getText().toString())) {
                Toast.makeText(MainActivity.this, "Password Missmatch", Toast.LENGTH_SHORT).show();
            } else if (text1.getText().toString().equals(text2.getText().toString())){
                editor.putString("password", text1.getText().toString());
                editor.putBoolean("checked", true);
                editor.commit();
                Intent intent = new Intent(MainActivity.this,FileEdit.class);
                startActivity(intent);

            }
    });
}
if(ischecked){
    text1.setVisibility(View.INVISIBLE);
    text2.setHint("Password");
    button1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(text2.getText().toString() .equals(sp.getString("password",text1.getText().toString()))){
                Intent intent = new Intent(MainActivity.this,FileEdit.class);
                startActivity(intent);
            }else{
                Toast.makeText(MainActivity.this,"Password Missmatch",Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

FileEditor 部分:

先绑定控件

```
final EditText filename = (EditText) findViewById(R.id.filename);
final EditText content = (EditText) findViewById(R.id.content);
Button save = (Button) findViewById(R.id.save);
Button load = (Button) findViewById(R.id.load);
Button clear = (Button) findViewById(R.id.clear);
Button delete = (Button) findViewById(R.id.delete);
TextView Title = (TextView) findViewById(R.id.textView);
```

Save 按钮点击事件，通过 FileOutputStream 向 filename 指定的文件写入 content 的内容，如果对应的文件不存在，则 openFileOutput（String，int）会新建一个文件，这个函数返回一个 FileOutputStream 对象，可以调用里面的 write 方法将 content 内容写入文件

```
save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        FileOutputStream outputStream;
        try {
            //打开文件
            outputStream = openFileOutput(filename.getText().toString(), Context.MODE_PRIVATE);
            outputStream.write(content.getText().toString().getBytes("UTF-8"));//写入文件
            outputStream.close();
            Toast.makeText(FileEdit.this, "Save successfully", Toast.LENGTH_SHORT).show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

Load 按钮点击事件，与 save 相反，文件读取使用 openFileInput（String）函数，它返回一个 FileInputStream 对象，调用里面的 read 方法可以读取文件中的内容。

```
load.setOnClickListener((v) -> {
        FileInputStream fileInputStream;
        try {
            fileInputStream = openFileInput(filename.getText().toString());
            byte[] initial = new byte[fileInputStream.available()];
            fileInputStream.read(initial);
            String str = new String(initial, "UTF-8");
            content.setText(str);
            fileInputStream.close();
            Toast.makeText(FileEdit.this, "Load successfully", Toast.LENGTH_SHORT).show();
        } catch (IOException ex) {
            Log.e("TAG", "Fail to read file.");
            Toast.makeText(FileEdit.this, "Fail to load file", Toast.LENGTH_SHORT).show();
        }

});
```

Clear 按钮点击事件，将 content 的内容通过 setText（""）清空。

```
clear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        content.setText("");
    }
});
```

Delete 按钮点击事件，通过 deleteFile（String）函数删除指定文件，如果删除成功，这个函数会返回 true，否则返回 false。

```
delete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (FileEdit.this.getApplicationContext().deleteFile(filename.getText().toString())) {
            Toast.makeText(FileEdit.this, "Delete Successfully", Toast.LENGTH_LONG).show();
        }else{
            Toast.makeText(FileEdit.this, "Fail to Delete File", Toast.LENGTH_LONG).show();
        }
    }
});
```

（3） 实验遇到困难以及解决思路

1. 如何指定 EditText 占据指定大小的空间，查阅了资料之后知道了，我们可以先使用一个 LinearLayout，给这个 LinearLayout 设置一个 weight = A。然后 EditText 和其他控件件包含在这个线性布局里面，并且给这些控件都设置一个 weight=wi（1<=i<=控件个数），且 wi 的总和=A。这样就会按照 Wi/A 的比例分配这个线性布局的空间给各个控件，从而实现指定 EditText 占据指定大小的空间的需求。

2.文档没有给如何删除一个文件，通过查资料知道了可以使用 deleteFile（String）这个函数，其中 String 是文件的路径，如果删除成功，会返回一个 true 的布尔值，否则返回一个 false。

五、 思考题

在实验报告中简要描述 Internal Storage 和 External Storage 的区别以及他们分别适用的场景

答：Internal Storage 把数据存储在设备内部存储器上，默认情况下在这里存储的数据为应用的私有数据，其他应用程序和用户都不能访问，只有应用本身可以看到和使用，当用户卸载 app 的时候，这些数据会跟着 app 一起被清除干净。所以当我们不希望被用户和其他 app 访问某些数据而且这些数据不是太大的话，可以选择使用 Internal Storage。

External storage 是一个公开的存储空间，也就是说用户和其他 app 也可以访问，当卸载 app 的时候，系统仅仅会删除 external 根目录（getExternalFilesDir()）下的相关文件，而不会删除 Environment.getExternalStorageDirectory()获取的 SD 卡路径中的数据。当数据不需要严格的访问权限，允许其他 app 或用户访问，又或者文件大小比较大时，选择 External storage 是个不错的选择。

六、 课后思考及感想

这次实验我学会了使用文件读取、写入和删除功能，并且学会使用 SharedPreferences 来保存一些信息。这为我们之后的期末 project 又提供了一大武器。这次实验不难，遇到一点问题（如前面所述）也可以通过查资料很快解决。收获还是挺大的。