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

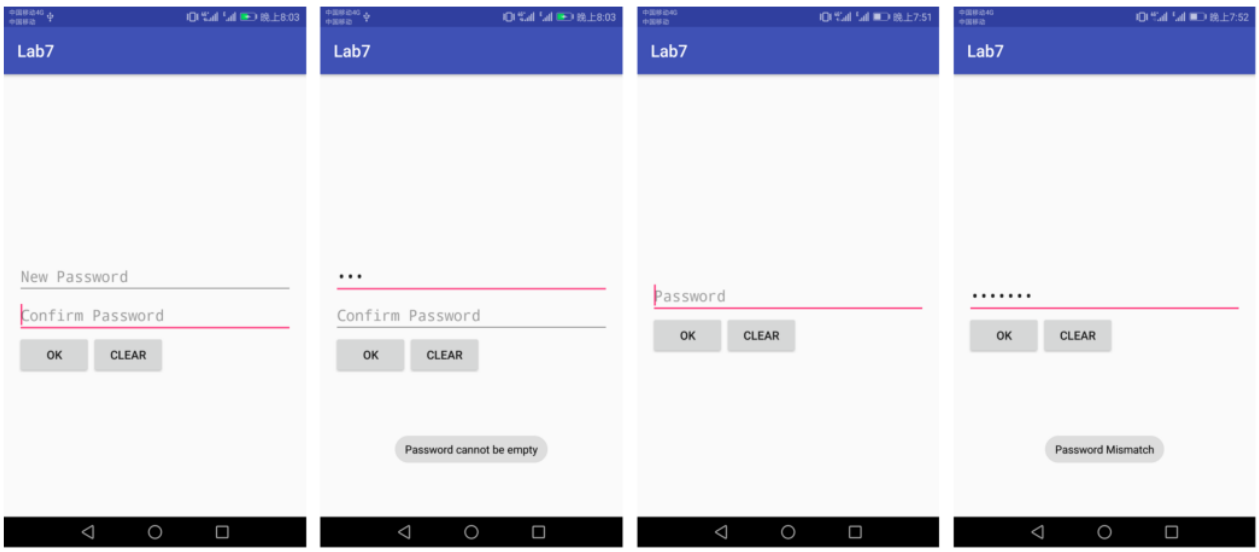
年级	15 级	专业（方向）	软件工程(移动)
学号	15352133	姓名	黄少豪
电话	13727024545	Email	328730316@qq.com
开始日期	2017.12.4	完成日期	2017.12.5

一、 实验题目

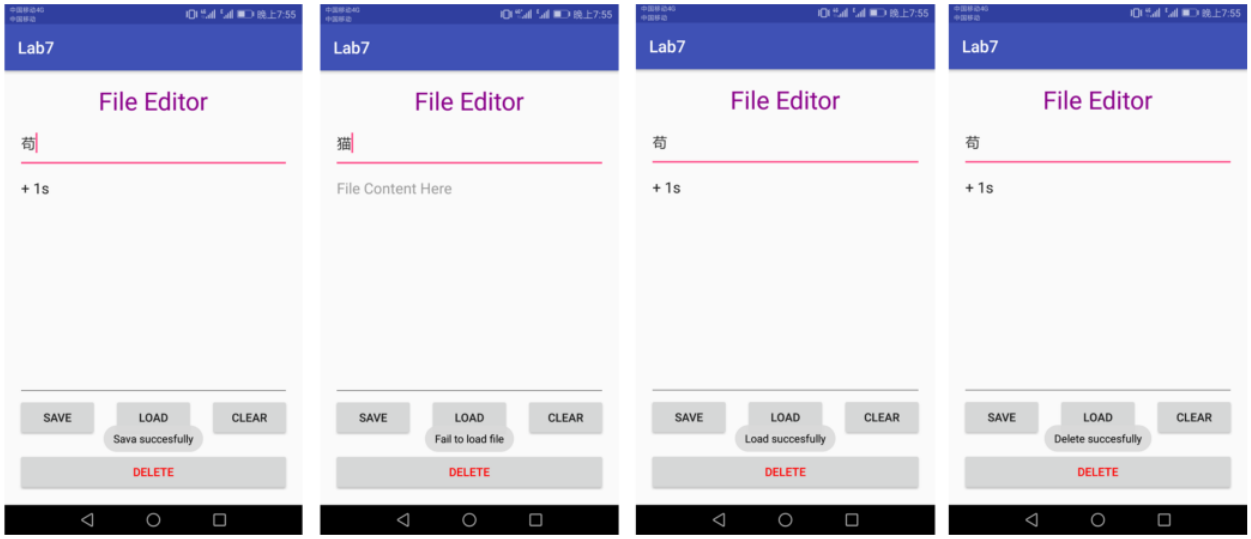
数据存储（一）

二、 实现内容

初始化密码界面，输入密码和确认密码后进入密码输入界面，输入正确密码后可进入文件编辑界面。若其中某一环节出现错误，均以 Toast 提示，详细效果见下图：



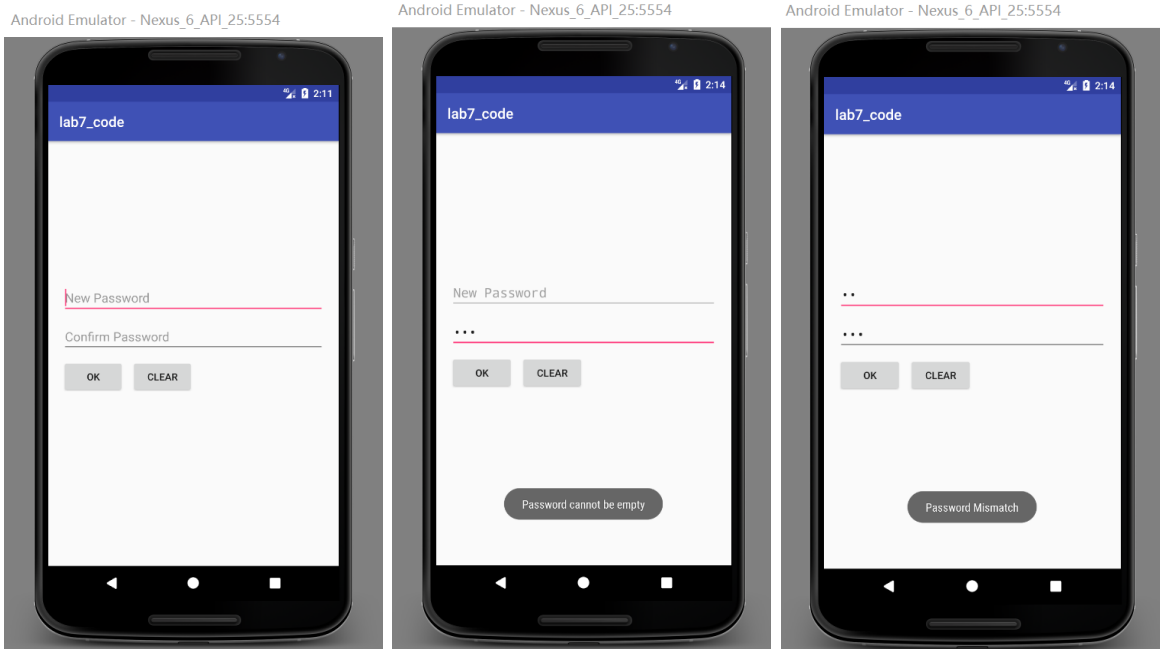
进入文件编辑界面，第一栏输入文件名，中间空白区域输入文本内容。



下面四个按钮，SAVE 表示将文件内容保存到文件名指向的文件；LOAD 表示将文件名指向的文件内容导入到文件内容区；CLEAR 表示清空文件内容区的内容；DELETE 表示删除文件名指向的文件。

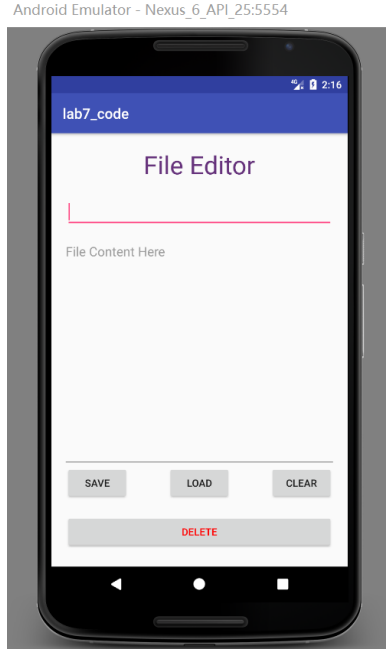
三、 课堂实验结果

（1） 实验截图

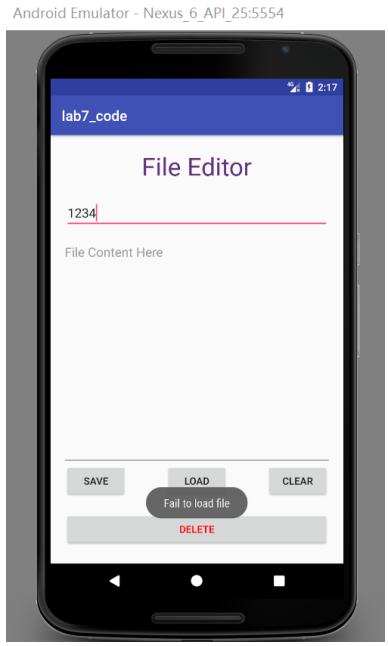


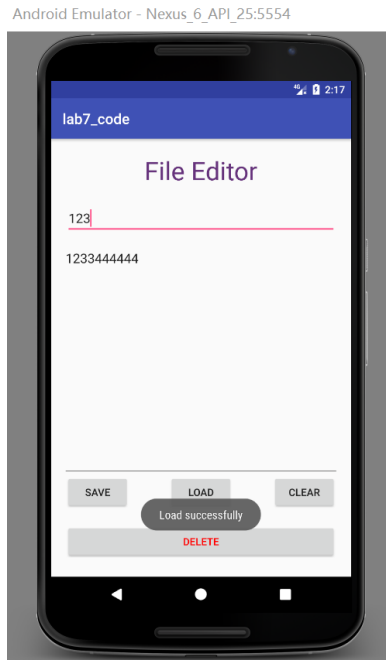

Android Emulator - Nexus_6_API_25:5554
2:15
lab7_code
Password
OK
CLEAR

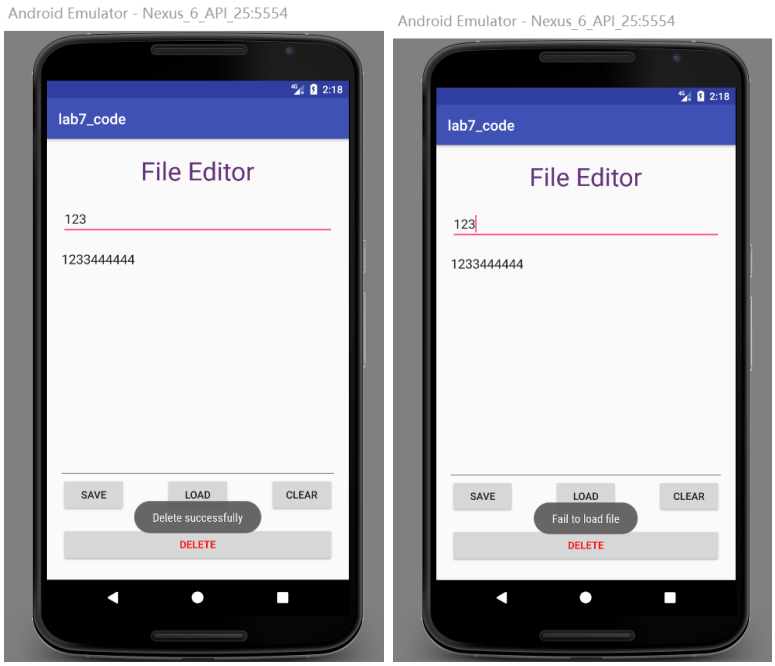

Android Emulator - Nexus_6_API_25:5554
2:15
lab7_code
OK
CLEAR
Password Mismatch


Android Emulator - Nexus_6_API_25:5554
2:16
lab7_code
File Editor
File Content Here
SAVE
LOAD
CLEAR
DELETE


Android Emulator - Nexus_6_API_25:5554
2:17
lab7_code
File Editor
123
1233444444
SAVE
LOAD
CLEAR
Save successfully
DELETE


Android Emulator - Nexus_6_API_25:5554
2:17
lab7_code
File Editor
1234
File Content Here
SAVE
LOAD
CLEAR
Fail to load file
DELETE


Android Emulator - Nexus_6_API_25:5554
2:17
lab7_code
File Editor
123
1233444444
SAVE
LOAD
CLEAR
Load successfully
DELETE



（2） 实验步骤以及关键代码

1. 根据实验图示，将密码设置和输入界面设计好

2. 创建 SharedPreferences 对象，判断该 sharedPreferences 内部是否已保存 password。若有保存，说明之前已经设置过密码，此时隐藏密码设置界面，显示密码输入界面；否则，说明还没有设置密码，此时显示密码设置界面，隐藏密码输入界面。

```
context = MainActivity.this;
sharedPref = context.getSharedPreferences("PAWWORD", Context.MODE_PRIVATE);
String password = sharedPref.getString("password", "");
if (password.length() > 0) {
    constraintLayout = (ConstraintLayout) findViewById(R.id.second);
    constraintLayout.setVisibility(View.VISIBLE);
    constraintLayout = (ConstraintLayout) findViewById(R.id.first);
    constraintLayout.setVisibility(View.GONE);
} else {
    constraintLayout = (ConstraintLayout) findViewById(R.id.second);
    constraintLayout.setVisibility(View.GONE);
    constraintLayout = (ConstraintLayout) findViewById(R.id.first);
    constraintLayout.setVisibility(View.VISIBLE);
}
```

3. 实现密码界面各个按钮点击后的逻辑。

```
public void buttonClick(View view) {
    if (view.getId() == R.id.first_ok) {    //当密码设置界面的ok按钮被点击
        newPassword = (EditText) findViewById(R.id.first_new_password);
        confirmPassword = (EditText) findViewById(R.id.first_confirm_password);
        //若密码条不为空且两个密码相同
        if (newPassword.getText().toString()
                .equals(confirmPassword.getText().toString())
                && newPassword.getText().toString().length() > 0){
            SharedPreferences.Editor editor = sharedPref.edit();
            editor.putString("password", newPassword.getText().toString());
            editor.commit();    //将密码写入SharedPreferences的PASSWORD文件中，key为password
            constraintLayout = (ConstraintLayout) findViewById(R.id.second);
            constraintLayout.setVisibility(View.VISIBLE);
            constraintLayout = (ConstraintLayout) findViewById(R.id.first);
            constraintLayout.setVisibility(View.GONE);
            //进入密码输入界面
        }
```

```
else if (view.getId() == R.id.second_ok) {  //在密码输入界面点击ok按钮
    newPassword = (EditText) findViewById(R.id.second_password);
    String password = sharedPref.getString("password", "");
    //若输入密码等于存储在SharedPreferences的PASSWORD文件中的password
    if (password.equals(newPassword.getText().toString())) {
        //进入文件内容编辑界面
        Intent intent = new Intent(MainActivity.this, FileEdit.class);
        startActivity(intent);
    }
    else
        Toast.makeText(MainActivity.this, "Password Mismatch", Toast.LENGTH_SHORT).show();
        //弹出密码不匹配提示
}
```

4. 按照实验图示写好 FileEdit 的 Activity 界面。中间的文本编辑区使用 ScrollView 固定高度，这样多出的内容就可以通过滚动界面查看了，同时多出的内容也不会将底下的按钮往下面挤，界面的整体布局不会因为文本内容的增多或减少而变化。EdiText 的 background 属性设置成@null 可以除去下划线。

```
<ScrollView
    android:id="@+id/content"
    android:layout_width="match_parent"
    android:layout_height="300dp"
    app:layout_constraintTop_toBottomOf="@id/name"
    android:layout_marginTop="20dp">

    <EditText
        android:id="@+id/body"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:hint="File Content Here"
        android:background="@null"/>

</ScrollView>
```

5. 实现 FileEdit 界面各个按钮的点击逻辑

```
public void buttonClick(View view) {
    name = (EditText) findViewById(R.id.name);
    content = (EditText) findViewById(R.id.body);
    if (view.getId() == R.id.save) {    //点击SAVE按钮
        try (FileOutputStream fileOutputStream
                     = openFileOutput(name.getText().toString().trim(),
                MODE_PRIVATE)) {
            //将文本编辑区的内容输出到文件名指定的文件
            fileOutputStream.write(content.getText().toString().getBytes());
            Toast.makeText(this, "Save successfully", Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            Toast.makeText(this, "Fail to save file", Toast.LENGTH_SHORT).show();
        }
    }
```

```
else if (view.getId() == R.id.load) {   //点击LOAD按钮
    try (FileInputStream fileInputStream
                 = openFileInput(name.getText().toString().trim())) {
        byte[] contents = new byte[SIZE];
        StringBuilder sb = new StringBuilder();
        int len;
        //一次读取1K数据，循环读取直到文件所有内容读取完毕
        while ((len = fileInputStream.read(contents)) != -1)
            sb.append(new String(contents, 0, len));
        //将读取的文件内容写入文本编辑区
        content.setText(sb.toString());
        Toast.makeText(this, "Load successfully", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        Toast.makeText(this, "Fail to load file", Toast.LENGTH_SHORT).show();
    }
}
```

```
else if (view.getId() == R.id.delete) { //点击delete按钮
    try {
        Context context = getApplicationContext();
        //获取context，调用deleteFile方法删除文件名指定文件
        context.deleteFile(name.getText().toString().trim());
        Toast.makeText(this, "Delete successfully", Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        Toast.makeText(this, "Fail to delete file", Toast.LENGTH_SHORT).show();
    }
}
```

6. 点击返回按钮，跳过密码输入界面到达 home

   在 AndroidManifest.xml 文件的 MainActivity 描述中将 noHistory 属性设置为 true

```
<activity android:name=".MainActivity" android:noHistory="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".FileEdit"></activity>
```

（3） 实验遇到困难以及解决思路

1. 删除文件一开始是使用 Java 的 File 对象，通过调用其 delete 方法来删除文件。但后来发现这种方法删除文件虽然 Toast 是显示 Delete successfully，但是 load 这个文件依然能成功且内容还是原来的样子。后来改用 Context 对象的 deleteFile 方法删除文件就能真正地成功地删除文件了。

2. 加载文件到文件编辑区时，加载进来的内容出现乱码。后来发现是使用了 toString 方法将 bytes 转换成 String 导致的。改成使用 new String(bytes)的构造方法来将 bytes 转换成 String 对象后就解决问题了。

3. 将 Text 对象当作 String 对象处理导致出现了各种问题。使用 toString 方法将 Text 对象先转换成 String 对象再处理后问题得到了解决。

四、 课后实验结果

五、 实验思考及感想

1. 使用 SharedPreferences 对象存储一些配置数据非常方便。SharePreferences 可以通过文件名区分数据，在一个文件内可以通过键值对存取数据且涵盖很多基本常用数据类型，使用起来非常方便。读取 SharePreferences 的数据时还需要加上一个相同数据类型的参数作为默认值。一旦不存在这个数据，SharePreferences 对象会返回这个参数作为默认值。这样我们就能很灵活地判断是否存在我们需要的数据并作出相应的处理了。

2. 使用 FileInputStream 对象读取文件内容时，读进来的是字节串，所以使用字节数组来存储读入的数据。为了不让程序占用内存过高，这个读数据的 byte 数组应该要限制一下一次读入的数据量，即使用 contents = new byte[SIZE](SIZE 可以设置为 1024，表示读入 1KB 的数据量)来代替 PPT 中的 contents = new byte[fileInputStream.available()]。因为文件内容可能很多，使用后者的话可能会使数组空间很大，导致程序占用内存高。而使用前者读取数据可能会出现一次读取不完的情况，所以可以使用一个循环读取数据直到读取完所有的文件数据。

3. 使用 File 对象的 delete 方法并没有成功删除文件，最后是使用 Context 的 deleteFile 方法实现的文件删除功能。查看网络有人说是因为文件流没有关闭导致删除失败，但后来尝试过发现并不是这个问题。而且如果是这个原因导致的，在 try 代码块内应该会 throw 一个异常，但实际上代码并没有 catch 到一个异常。综上，个人总结在 Android 中还是使用 Context 对象的 deleteFile 方法来删除文件吧。

4. Internal Storage：存储应用文件与数据，只对应用可见，系统文件管理器及其他应用均无法查看，修改或调用这一块存储的数据，只有应用自身可以操作这一块存储的数据，类似于类对象里面的 private 成员。当应用被卸载时，这一块存储的数据与文件也一起被删除。
适用场景：需要存储一些不希望被其他应用或用户看到的数据时，可以使用这种存储方式。确保文件与数据内容不容易被修改，需要保证数据安全的，可以使用这种存储方法。

External Storage： 是公共存储文件与数据的地方，这里面存储的数据与文件都是共享的，是可以被各种应用与用户访问的，甚至有些还可以直接打开修改里面的内容(可能需要权限)。存储在这里的文件与数据并不是安全的，一些误操作可能会将一些文件与数据篡改或删除。同时当外部存储设备被拔出时，存储于其中的数据与文件也会变得无法被应用使用。当应用被卸载时，这里面的部分数据并不会被删除，而是保留下来成为所谓“卸载剩余文件”。存储在这里的数据可以被多个应用复用，节约存储资源。同时一些占用空间大的媒体文件存储在这里也是很方便，可以被多种应用调用播放。
适用场景：一些大文件的存储，一些可以被多种应用复用的数据与文件可以使用这种存储方式。