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）

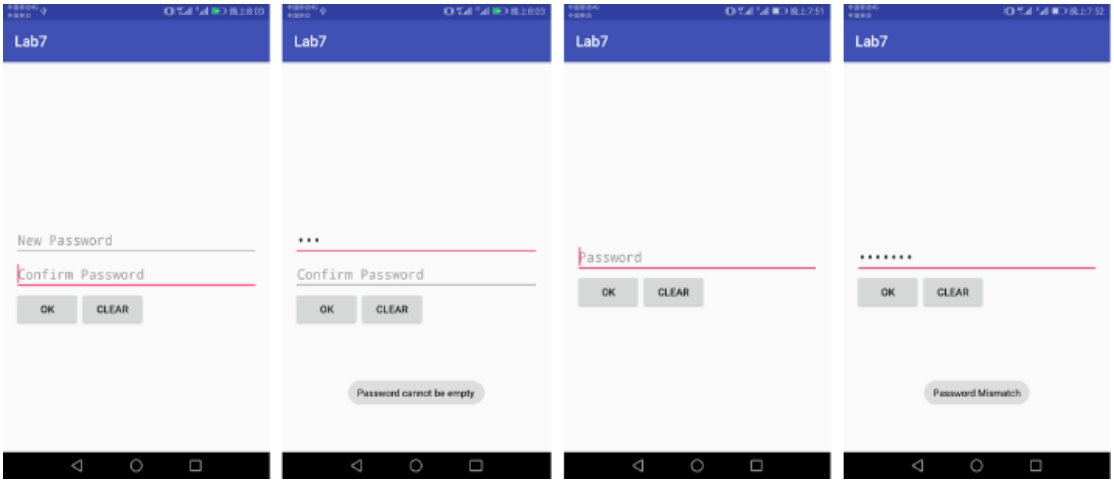
课程名称：移动应用开发　　任课教师： 郑贵峰

年级	15 级	专业（方向）	软件工程（移动信息工程） 互联网方向
学号	15352211	姓名	林苗
电话	13763360840	Email	554562948@qq.com
开始日期	2017.12.11	完成日期	2017.12.11

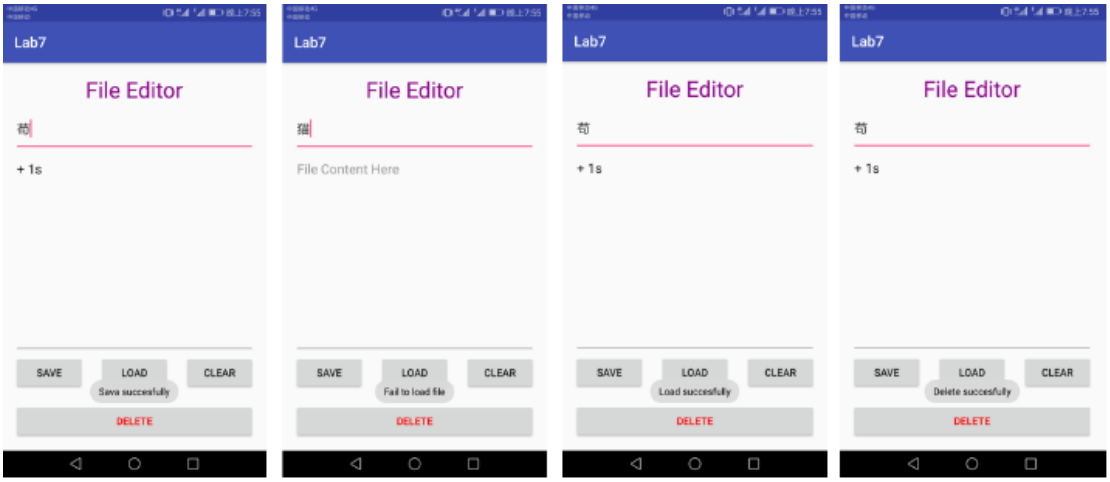
一、 实验题目

数据存储（一）

二、 实现内容



从左往右依次是初始密码界面、密码为空提示、密码匹配后重新进入界面、密码错误提示。



从左往右依次是：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。

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；

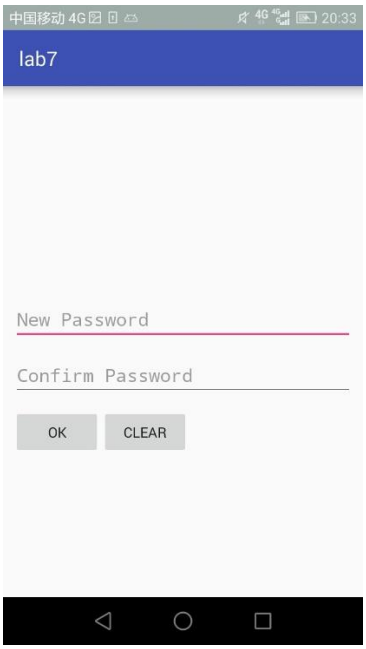
d、 点击 LOAD 按钮，能够按照文件名从内存中读取文件内容，并将文件内容写入到编辑框中。如果成功导入，则弹出成功的 Toast 提示，如果导入失败（例如：文件不存在），则弹出读取失败的 Toast 提示。

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。

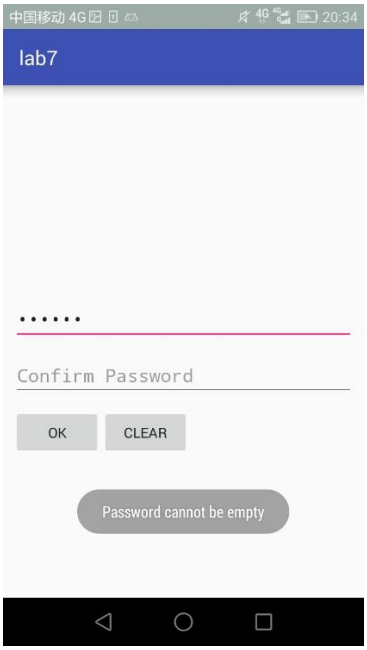
4、 特殊要求：进入文件编辑的 Activity 之后，如果点击返回按钮，则直接返回 Home 界面，不再返回密码输入界面。

三、 课堂实验结果

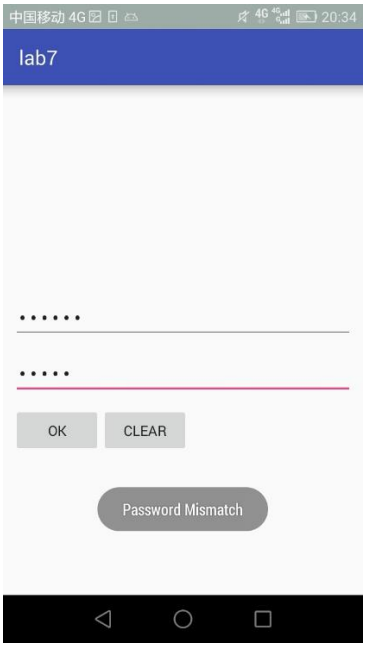
（1） 实验截图



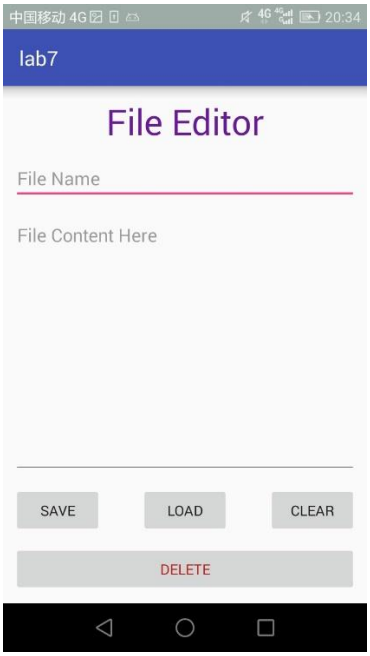
初始界面



密码为空

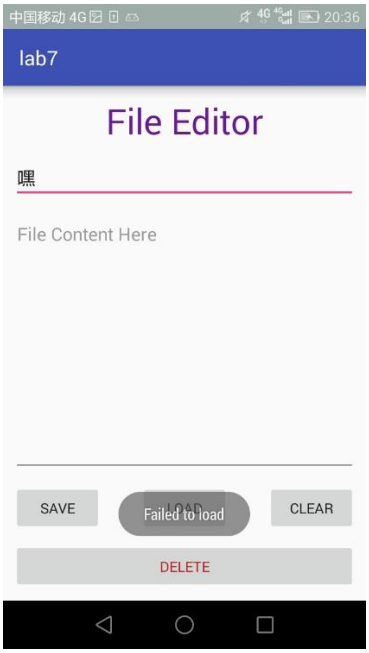


两个密码不匹配



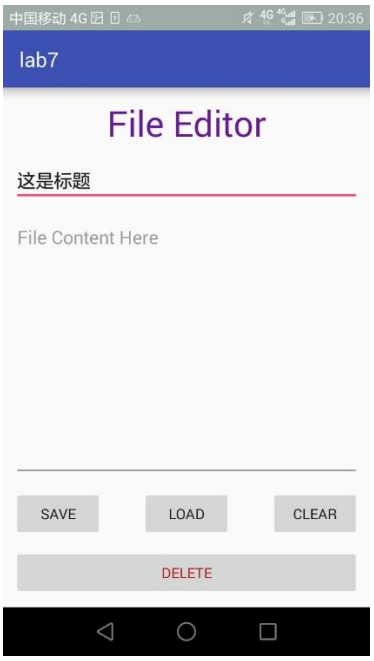
密码匹配，进入编辑界面

输入标题、内容，保存

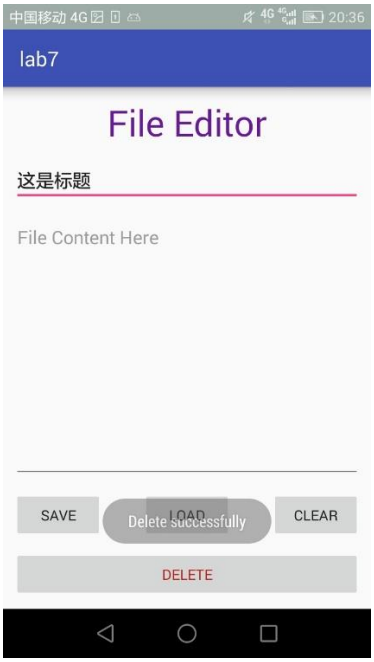


加载没有保存过的文件

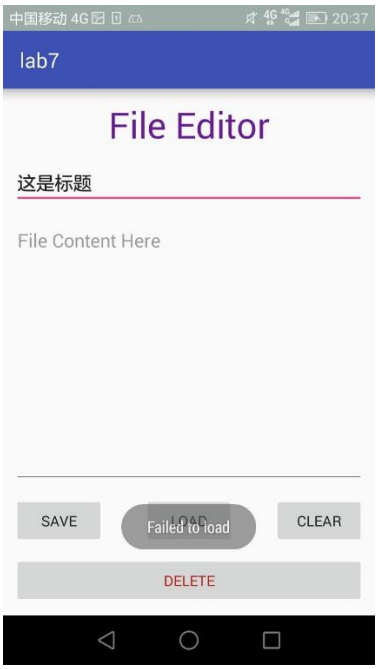
加载保存过的文件



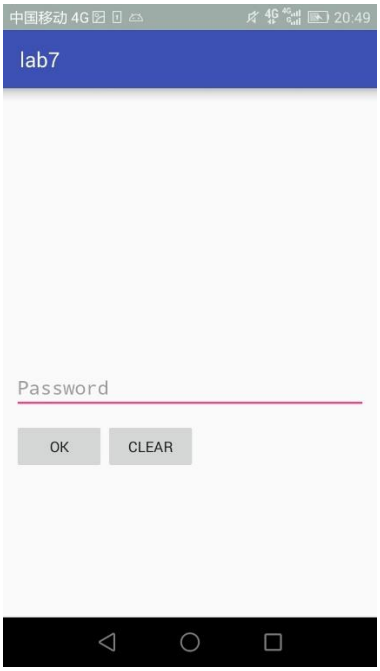
点击 CLEAR，清空内容



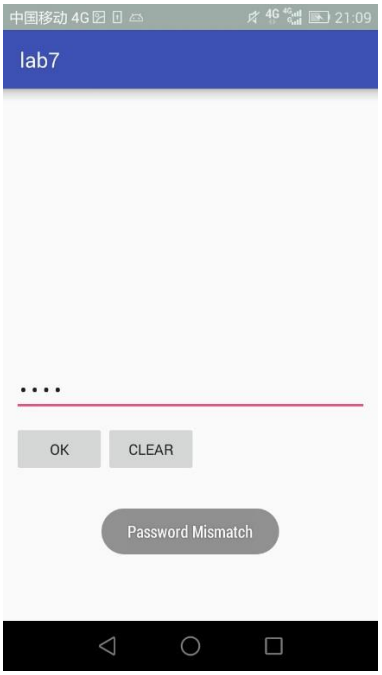
删除文件



加载已删除的文件



重新启动应用



密码不正确

（2） 实验步骤及关键代码

在 MainActivity 中，将变量与 xml 文件中的 id 绑定起来。

```
final EditText NP = (EditText) findViewById(R.id.newpw);
final EditText CP = (EditText) findViewById(R.id.confpw);
final Button OK = (Button) findViewById(R.id.ok);
final Button CLEAR =  (Button) findViewById(R.id.clear);
```

SharedPreference 对象及其编辑器。

```
final SharedPreferences myPreference = getSharedPreferences("password",Context.MODE_PRIVATE);
final SharedPreferences.Editor editor = myPreference.edit();
```

实现如果设置过密码，则以后启动应用只显示一个密码输入框，且 hint 为“Password”。

```
if(myPreference.getString("password",null)!=null){
    Log.e("debug",myPreference.getString("password",null));
    NP.setVisibility(View.INVISIBLE);
    CP.setHint("Password");
}
```

重写确定按钮的监听的监听事件，分为未设密码和已设密码两种情况。

```
  OK.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
          String newpassword = NP.getText().toString();
          String confirmpassword = CP.getText().toString();
```

对于未设密码，有两个输入框，需要实现两个密码一致才能够保存和跳转到编辑界面。

```
if(myPreference.getString("password",null)==null){
    if(newpassword.isEmpty() || confirmpassword.isEmpty()){ //密码为空
        Toast.makeText(MainActivity.this, "Password cannot be empty",Toast.LENGTH_SHORT).show();
    }
    else if(!newpassword.equals(confirmpassword)){ //密码不匹配
        Toast.makeText(MainActivity.this, "Password Mismatch",Toast.LENGTH_SHORT).show();
    }
    else{ //密码匹配，保存密码，跳转
        editor.putString("password",confirmpassword);
        editor.commit();
        Intent intent = new Intent(MainActivity.this,FileEdit.class);
        startActivity(intent);
    }
}
```

对于已经设置了密码的，则只有一个密码输入框，判断与保存的密码是否一致，进行相应操作。

```
else{
    if(confirmpassword.isEmpty()){  //密码为空
        Toast.makeText(MainActivity.this, "Password cannot be empty",Toast.LENGTH_SHORT).show();
    }
    else if(confirmpassword.equals(myPreference.getString("password",""))){  //密码匹配，跳转
        Toast.makeText(MainActivity.this, "Password Correct",Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(MainActivity.this,FileEdit.class);
        startActivity(intent);
    }
    else{ //密码不匹配
        Toast.makeText(MainActivity.this, "Password Mismatch",Toast.LENGTH_SHORT).show();
    }
}
```

重写 CLEAR 按钮的事件监听，当点击该按钮时，两个输入框均清空。

```
CLEAR.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        MP.setText("");
        CP.setText("");
    }
});
```

新建一个 Activity，实现界面。
将变量与 id 绑定起来：

```
final EditText TITLE = (EditText)findViewById(R.id.title);
final EditText MAIN = (EditText)findViewById(R.id.main);
final Button SAVE = (Button)findViewById(R.id.save);
final Button LOAD = (Button)findViewById(R.id.load);
final Button CLEAR = (Button)findViewById(R.id.clear);
final Button DELETE = (Button)findViewById(R.id.delete);
```

实现保存按钮的监听事件，点击保存，如果找到文件名，就直接对该文件修改，如果没有找到文件名，就创建新文件。

```
SAVE.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String title = TITLE.getText().toString();
        String main = MAIN.getText().toString();
        try(FileOutputStream fileOutputStream = openFileOutput(title,MODE_PRIVATE)) {
            fileOutputStream.write(main.getBytes()); //将数据写入文件
            fileOutputStream.flush(); //将所有剩下的数据写入文件
            fileOutputStream.close(); //关闭
            Toast.makeText(FileEdit.this, "Save Successfully",Toast.LENGTH_SHORT).show();
        }
        catch (IOException ex){
            Log.e("debug","Fail to save file");
        }
    }
});
```

实现 LOAD 监听事件，读取文件，显示在文本编辑框，如果读不到，就弹出 toast 信息。

```
LOAD.setOnClickListener((v) -> {
        String title = TITLE.getText().toString();
        try(FileInputStream fileInputStream = openFileInput(title)) {
            Log.e("debug","Success to load");
            byte[] contents = new byte[fileInputStream.available()];
            fileInputStream.read(contents);
            fileInputStream.close();
            MAIN.setText(new String(contents));
            Toast.makeText(FileEdit.this, "Load Successfully",Toast.LENGTH_SHORT).show();

        }
        catch (IOException ex){
            MAIN.setText("");
            Toast.makeText(FileEdit.this, "Failed to load",Toast.LENGTH_SHORT).show();
            Log.e("debug","Failed to load");
        }
});
```

实现 CLEAR 监听事件，当点击 CLEAR 的时候，文本编辑区置空。

```
CLEAR.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        MAIN.setText("");
    }
});
```

实现 DELETE 监听事件，当点击 DELETE 的时候，删除对应标题的文本。

```
DELETE.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        String title = TITLE.getText().toString();
        deleteFile(title);
        Toast.makeText(FileEdit.this, "Delete successfully",Toast.LENGTH_SHORT).show();
    }
});
```

本次实验要求实现点击返回键，回到 HOME 主页，即将 MainActivity 从 activity 的 stack 中弹出，直接在 Manifest.xml 中修改以下代码：

```
<activity android:name=".MainActivity"
    android:noHistory="true">
```

（3） 实验遇到的困难及解决思路

- 在编辑界面中，不知如何实现三个按钮间隔均匀分布。

  解决思路：用一个 Relativelayout 将三个按钮包起来，第一个与父元素的左边对齐，第二个相对于父元素水平居中，第三个与父元素的右边对齐。

- edit 主文本编辑框填充除了其它控件之外的位置，使充满整个屏幕。

解决思路：整体采用 Relativelayout，设置上面几个控件相对于父元素顶部的位置，设置下面几个控件相对于父元素底部的位置，然后设置主编辑框在标题编辑框之下，在按钮之上，将高度设置为 fill_parent。

- edit 主文本编辑框的提示显示在左上方。

解决思路：多加一个 gravity 属性，值为 top|left。

四、 实验思考及感想

简要描述 Internal Storage 和 External Storage 的区别以及它们分别适用的场景。

Internal Storage 中的文件只能被自己的应用访问到，且一个应用所创建的所有文件都在和应用包名相同的目录下，当应用被卸载之后，内部存储中的这些文件也会被删除。若将文件定义为私有，则即使其它 app 知道应用的包名，也无法访问。

因此 Internal Storage 适用于保存应用的私密数据。

External Storage 中的文件可以被用户或其它应用程序修改，当删除了应用之后，卸载前创建的应用仍然保留。

因此 External Storage 适用于保存数据量大的可共享的数据。

本次实验在文件读取方面没有遇到太大的难题，反而在布局方面考虑得比较多，但发现 Android Studio 的各种布局的功能都很强大，支持嵌套，也就可以利用多种布局的优点，从而有多种方法实现实验布局要求。

五、 实验参考资料

Android 属性设置 android:noHistory="true"
http://blog.csdn.net/zhangphil/article/details/45823501

android 中的文件操作详解以及内部存储和外部存储
http://blog.csdn.net/androidwifi/article/details/17725989/