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	2015	专业（方向）	互联网
学号	15352194	姓名	梁杰鑫
电话	15113959962	Email	Alcanderian@gmail.com
开始日期	2017.12.8	完成日期	2017.12.12

一、 实验题目

本次实验要求实现一个有密码保护的笔记本。

二、 实现内容

1、 本次实验需要实现两个 ctivity ；

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：
  1. 若 new password 为空，则弹出密码为空的提示；
  2. 若 new password 与 comfirm password 不匹配，则弹出不匹配的提示；
  3. 若密码不为空且互相匹配，则保存密码，进入文件编辑界面。
  4. 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。

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；

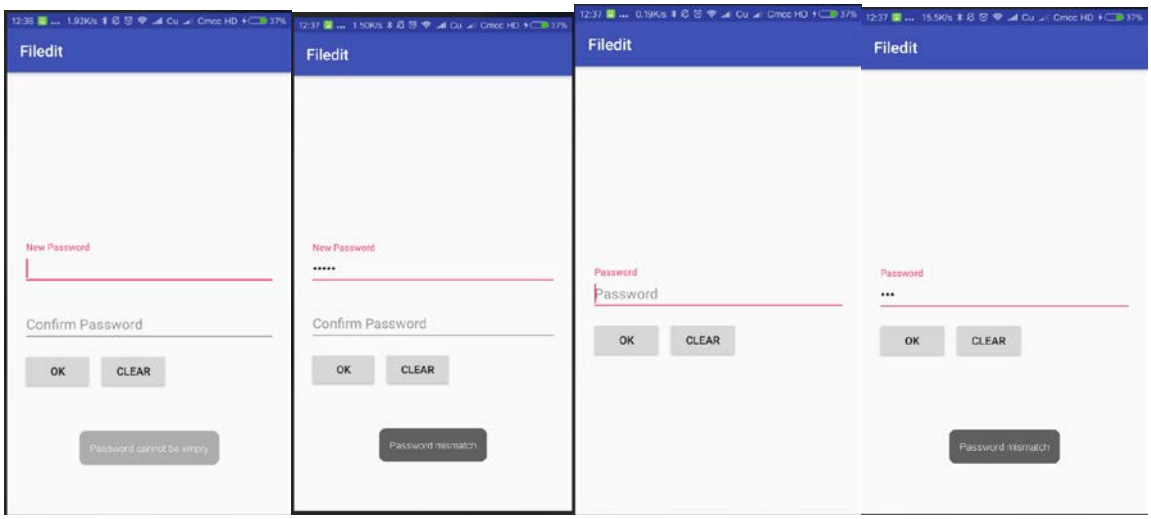
d、 点击 LOAD 按钮，能够按照文件名从内存中读取文件内容，并将文件内容写入到编辑框中。如果成功导入，则弹出成功的 Toast 提示，如果导入失败（例如：文件不存在），则弹出读取失败的 Toast 提示。

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。

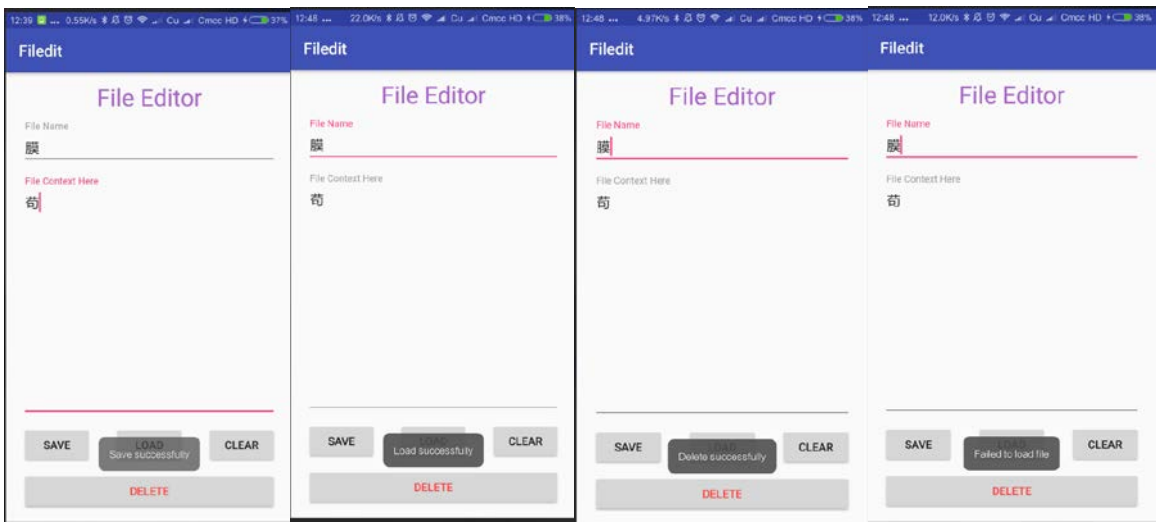
4、 特殊要求：进入文件编辑的 Activity 之后，如果点击返回按钮，则直接返回 Home 界面，不再返回密码输入界面。

三、 课堂实验结果

(1) 实验截图



密码为空　　密码不匹配　　初次输入密码后再打开 APP　　密码错误



保存文件　　读取文件　　删除文件　　删除后再读取

(2) 实验步骤以及关键代码

- 密码的 MD5 处理

利用 MD5 算法对密码串进行处理后再放入 SharedPreferences 中，可以提高密码的安全性。MD5 的创建和调用方法如下：

```
private MessageDigest md5;
```

```
try {
    md5 = MessageDigest.getInstance("md5");
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

MD5 加密要求我们输入 byte 数组。

```
String r_pwd = sp_pwd.getString("pwd", null);
String l_pwd = til_confirm_pwd.getEditText().getText().toString();
l_pwd = new String(md5.digest(l_pwd.getBytes()));
```

- SharedPreference

这是一个的属性库，通过读取指定的属性文件我们可以读取和存放我们的数据。

调出与读取数据：

```
sp_pwd = getSharedPreferences("alau_file_editor", MODE_PRIVATE);
has_pwd = sp_pwd.getString("pwd", null) != null;
```

存放数据：

```
String r_pwd = new String(md5.digest(c_pwd.getBytes()));
sp_pwd.edit().putString("pwd", r_pwd).commit();
```

- 文件的读写

在这次实验中我们使用内部存储来存储我们的笔记文件。

读取：

```
btn_load.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String filename = til_filename.getEditText().getText().toString().trim();

        if (checkFilename(filename)) {
            try (FileInputStream in = openFileInput(filename)) {
                byte[] context = new byte[in.available()];
                in.read(context);
                til_filectx.getEditText().setText(new String(context));
                in.close();
                makeToast("Load successfully");
            } catch (Exception e) {
                e.printStackTrace();
                makeToast("Failed to load file");
            }
        }
    }
});
```

利用 openFileInput 读取文件是默认读取内部储存中的文件的，读取出来的文件是 byte[]的形式，需要我们转换为 String。文件名不能为空而且不能包含路径分隔符“/”，所以我还提供一下函数判断文件名的合法性。

```
private Boolean checkFilename(String filename) {
    if (filename.isEmpty()) {
        makeToast("File name cannot be empty");
        return false;
    } else if (filename.contains("/")) {
        makeToast("File name cannot contains \"/\"");
        return false;
    }
    return true;
}
```

写入：

```
if (checkFilename(filename)) {
    try (FileOutputStream out = openFileOutput(filename, MODE_PRIVATE)) {
        out.write(filectx.getBytes());
        out.close();
        makeToast("Save successfully");
    } catch (Exception e) {
        e.printStackTrace();
        makeToast("Failed to save file");
    }
```

写入的时候需要调用 String.getBytes()将字符串转换为 byte[]再写入。

删除文件：直接调用 deleteFile(String filename)即可删除内部储存中的文件。

```
            if (checkFilename(filename)) {
                Boolean ok = deleteFile(filename);
                if (ok)
                    makeToast("Delete successfully");
                else
                    makeToast("Failed to delete file");

            }
        }
```

返回值表示删除成功与否。

四、 实验思考

- Internal Storage 和 External Storage 的区别

  一般情况下，内部存储中的文件只有应用程序可见，其他应用包括用户自身是无法访问这些文件的。这部分空间一般提供给应用存放配置文件、用户登陆信息这些只有本应用会使用的文件。

  外部储存的文件是对用户可见的。一般用于存放一些图片，音乐，文档这一类可以被其他应用打开来的，体积比较大的文件，或者应用的运行日志（提供给用户查看）等等。

  内部储存的文件会在应用被卸载之后删除，而外部储存中的文件会保留。