中山大学移动信息工程学院本科生实验报告

课程名称：移动应用开发　　任课教师：郑贵锋

一、 实验题目

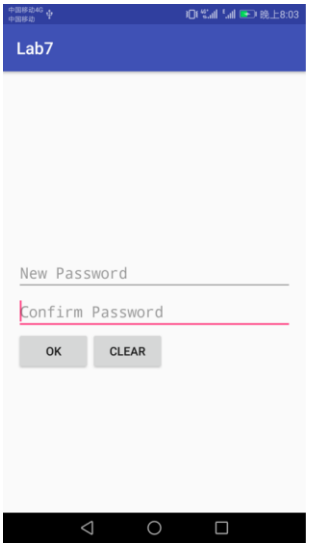
数据存储

二、 实验目的

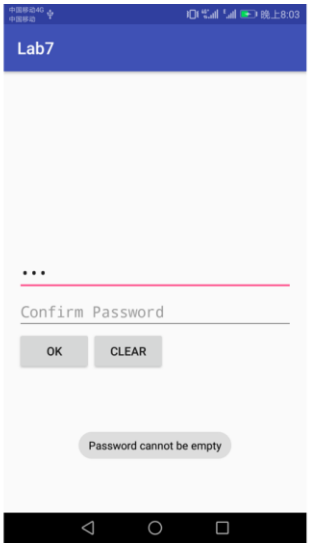
1. 学习 SharedPreferences 的基本使用
2. 学习 Android 中常见的文件操作方法
3. 复习 Android 界面编程

三、 实验内容

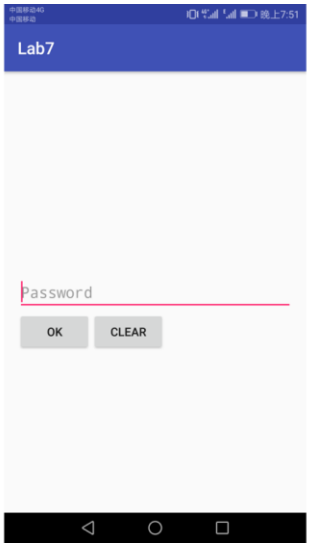
实现一个简单的数据存储界面如下：



初始密码界面

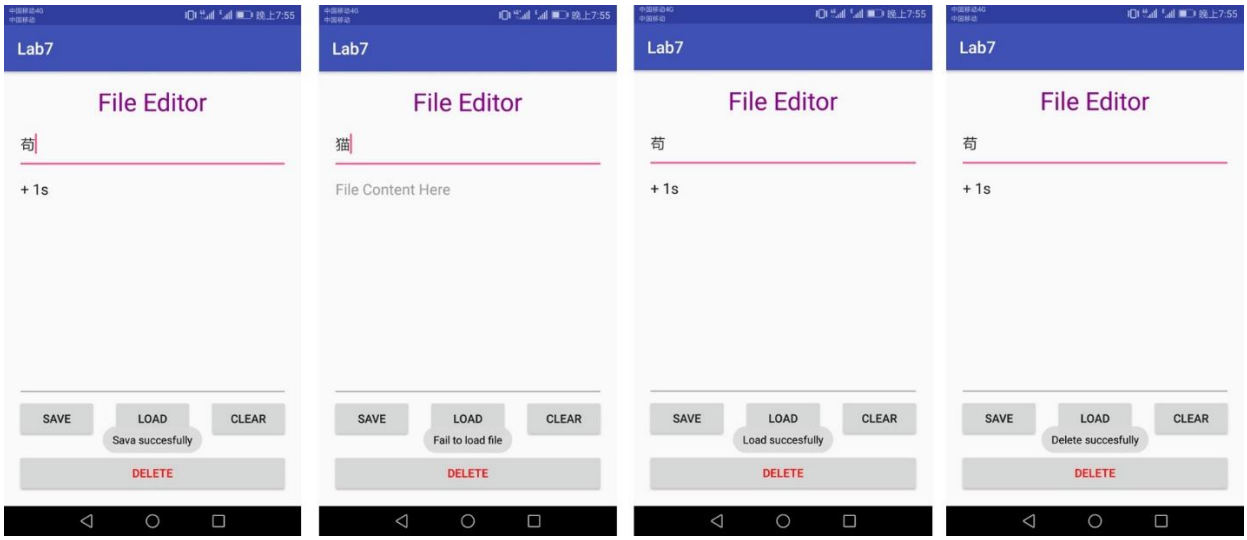


密码为空提示界面



密码匹配后重新进入

密码错误提示



保存成功　　写入失败　　写入成功　　删除成功

具体要求如下：

需要实现一个密码输入 activity：

1. 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；
2. 输入框下方有两个按钮：
   a) OK 按钮，点击之后：
   b) 若 new password 为空，则弹出密码为空的提示；
   c) 若 new password 与 comfirm password 不匹配，则弹出不匹配的提示；
   d) 若密码不为空且互相匹配，则保存密码，进入文件编辑界面。
   e) CLEAR 按钮，点击之后清除所有输入框的内容。
3. 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；
   a) 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；
   b) 点击 CLEAR 按钮后，清除密码输入框的内容。

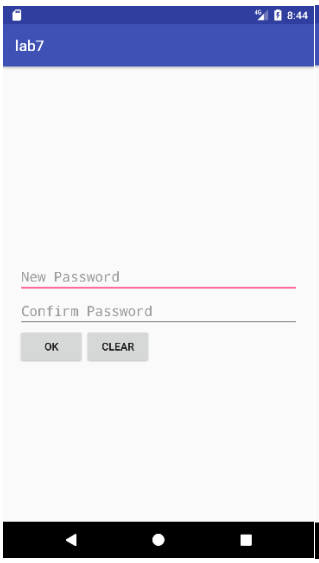
实现一个文件编辑 activity

1. 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；
2. 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；
3. 点击 CLEAR 按钮，能够清空文件内容编辑区域内的内容；
4. 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；
5. 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。

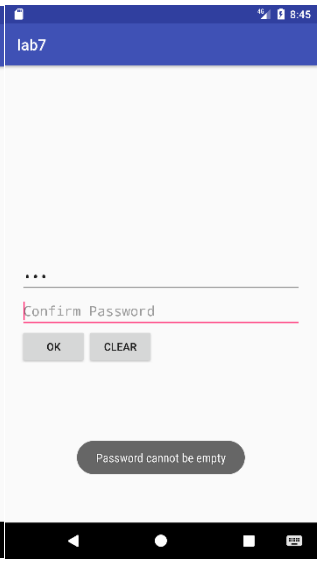
四、 课堂实验结果

（一） 实验截图：

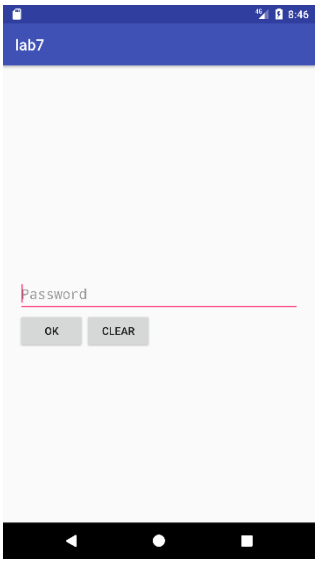
1. 虚拟机上效果如下：



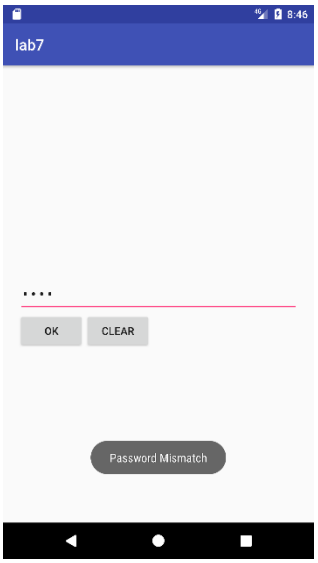
初始密码界面



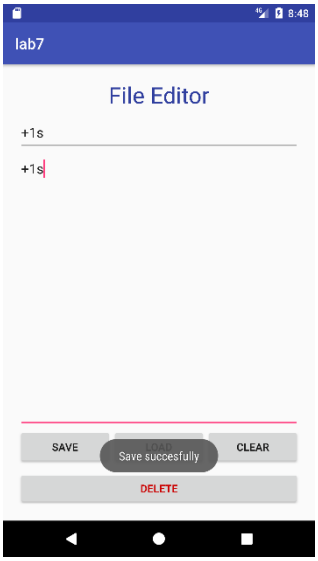
密码为空提示界面



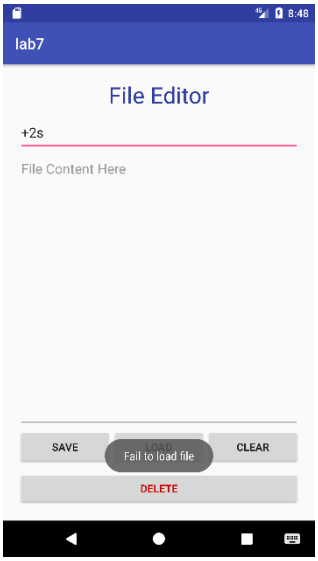
密码匹配后重新进入



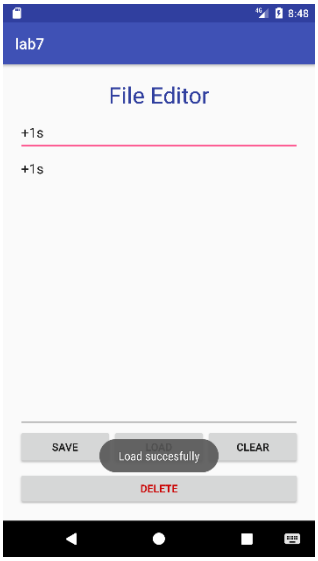
密码错误提示



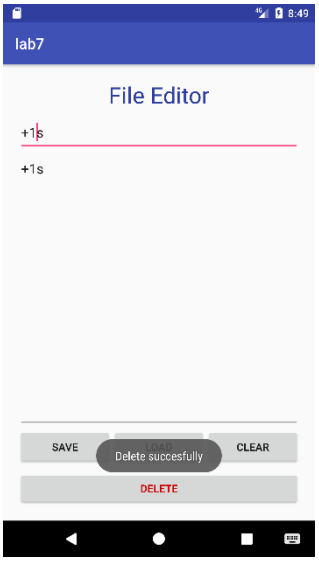
保存成功



写入失败



写入成功



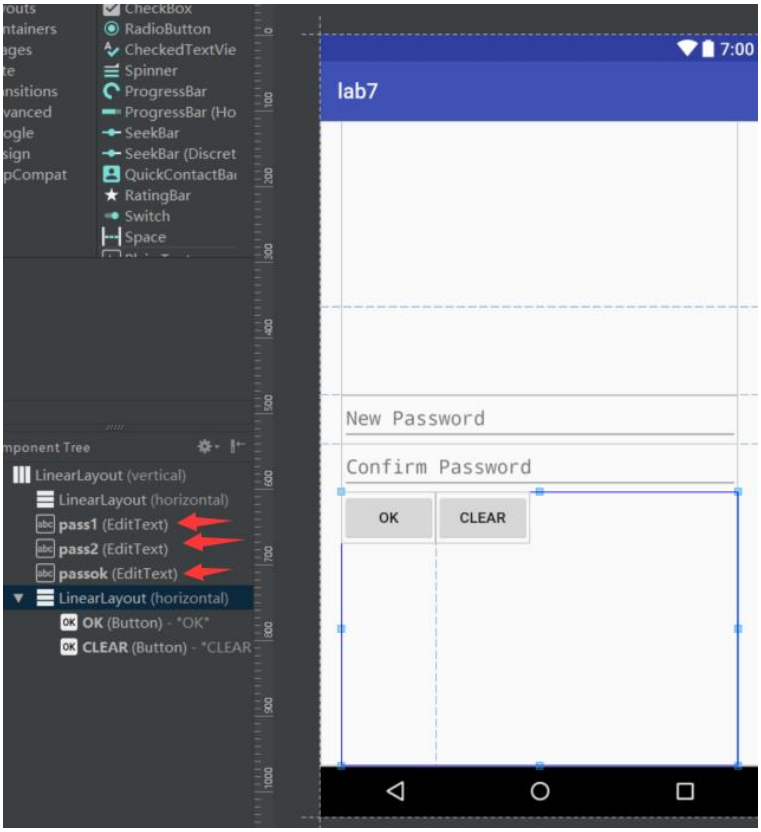
删除成功

（二） 关键步骤：

1. SharedPreferences 实现密码的存储

i. 界面的构建

主界面使用 LinearLayout 组合构建，使用空 LinearLayout 做到留白，通过控制 android:layout_weight 控制不同控件的所占的高度比，主界面中使用 3 个 EditText，根据不同情况显示其中两个或者其中一个。界面效果如下图：



ii. 实现判断是否已经存在密码，根据对应情况修改对应的布局。

```
/**
 * 检测 是否已经存在密码
 */
private void Exist_Passw(){
    SharedPreferences preferences=getSharedPreferences("pw",MODE_PRIVATE);
    pass=preferences.getString("pw",null);
    flag=(pass!=null);
}
/**
 *控件初始化
 */
private void Init(){
    pass1=(EditText)findViewById(R.id.pass1);
    pass2=(EditText)findViewById(R.id.pass2);
    passok=(EditText)findViewById(R.id.passok);
    ok=(Button)findViewById(R.id.OK);
    clear=(Button)findViewById(R.id.CLEAR);
    Exist_Passw();
    //修改布局
    if(flag){//密码存在
        pass1.setVisibility(View.GONE);
        pass2.setVisibility(View.GONE);
        passok.setVisibility(View.VISIBLE);
    }
    else{//密码不存在
        pass1.setVisibility(View.VISIBLE);
        pass2.setVisibility(View.VISIBLE);
        passok.setVisibility(View.GONE);
    }
}
```

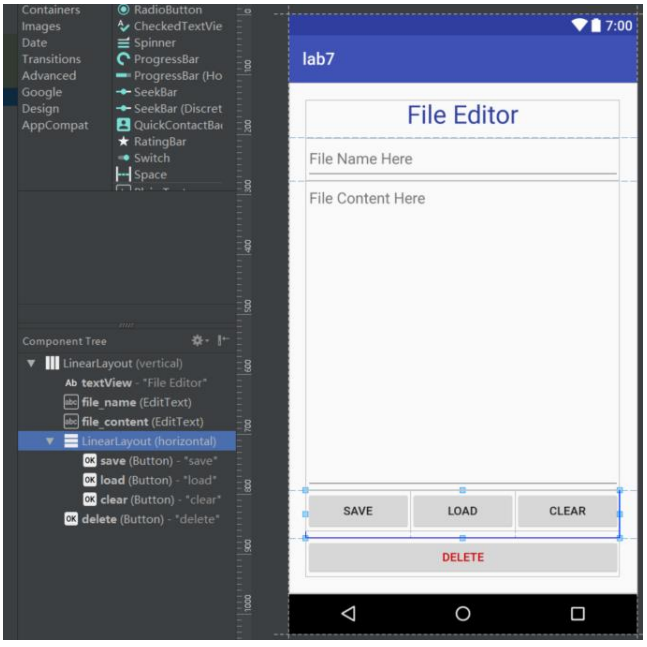
iii. 根据不同的布局设置相应的 OK 按钮点击事件和 CLEAR 点击事件。

```
/**
 * 设置点击事件监听器
 */
private void Click_init(){
    ok.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(flag){//密码存在
                String pw=passok.getText().toString();
                if(pw.length()==0)//密码为空
                    Toast.makeText(MainActivity.this,"Password cannot be empty",Toast.LENGTH_SHORT).show();
                else if(pass.equals(MD5Utils.md5Password(pw))){//不为空比较 MD5 判断
```

```
                    //密码匹配转跳
                    Intent i=new Intent(MainActivity.this,FileEdit.class);
                    startActivity(i);
                    finish();//直接销毁主界面
                }
                else //密码不匹配
                    Toast.makeText(MainActivity.this,"Password Mismatch",Toast.LENGTH_SHORT).show();
            }else{//密码不存在
                String pw1=pass1.getText().toString();
                String pw2=pass2.getText().toString();
                if(pw1.length()==0||pw2.length()==0)//密码为空
                    Toast.makeText(MainActivity.this,"Password cannot be empty",Toast.LENGTH_SHORT).show();
                else if(pw1.equals(pw2)){//两次输入密码相等
                    //存入密码
                    SharedPreferences preferences=getSharedPreferences("pw", MODE_PRIVATE);
                    SharedPreferences.Editor editor = preferences.edit();
                    editor.putString("pw", MD5Utils.md5Password(pw1));//MD5 加密
                    editor.commit();
                    //界面转跳
                    Intent i=new Intent(MainActivity.this,FileEdit.class);
                    startActivity(i);
                    finish();//销毁当前界面
                }
                else{//密码不匹配
                    Toast.makeText(MainActivity.this,"Password Mismatch",Toast.LENGTH_SHORT).show();
                    pass2.setText("");//清空一个
                }
            }
        }
    });
    //清空输入框
    clear.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            pass1.setText("");
            pass2.setText("");
            passok.setText("");
        }
    });
}
```

2. 文件编辑界面的实现

i. 实现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。此处使用 LinearLayout，使用 android:weightSum 和 android:layout_weight，设计每个控件所占父控件的百分比。界面布局如下：



ii. 实现根据不同的按钮实现文件的保存、载入、修改、删除。

- 点击 SAVE 按钮后保存到文件，成功保存后弹出 Toast 提示

```
save.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { //保存
        String name=file_name.getText().toString().replace("/","_");
        String content=file_content.getText().toString();
        if(name.length()==0){//文件名为空 提示
            Toast.makeText(FileEdit.this,"File Name cannot be empty",Toast.LENGTH_SHORT).show();
        }
        else{//不为空保存
            try(FileOutputStream fileOutputStream = openFileOutput(name,MODE_PRIVATE)){
                fileOutputStream.write(content.getBytes());//写入
                Toast.makeText(FileEdit.this,"Save succesfully",Toast.LENGTH_SHORT).show();
            }catch (IOException e){//保存失败
                Toast.makeText(FileEdit.this,"Fail to save file",Toast.LENGTH_SHORT).show();
                Log.e("TAG","Fail to save file");
            }
        }
    }
});
```

- 点击 CLEAR 按钮清空文件内容编辑区域内的内容

```
clear.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {//清理
        file_content.setText("");
    }
});
```

- 点击 LOAD 按钮，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，弹出成功的 Toast 提示，如果导入失败弹出读取失败的 Toast 提示。

```
load.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {//载入
        String name=file_name.getText().toString().replace("/","_");
        if(name.length()==0){//文件名为空 提示
            Toast.makeText(FileEdit.this,"File Name cannot be empty",Toast.LENGTH_SHORT).show();
        }
        else{//读入文件载入
            try(FileInputStream fileInputStream =openFileInput(name)){
                byte[] contents= new byte[fileInputStream.available()];
                fileInputStream.read(contents);//填充控件
                file_content.setText(new String(contents));
                Toast.makeText(FileEdit.this,"Load succesfully",Toast.LENGTH_SHORT).show();
            }catch (IOException e){//载入失败
                Toast.makeText(FileEdit.this,"Fail to load file",Toast.LENGTH_SHORT).show();
                Log.e("TAG","Fail to save file");
            }
        }
    }
});
```

- 点击 DELETE 按钮，按照文件名从内容中删除文件

```
delete.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {//删除
        String name=file_name.getText().toString().replace("/","_");
        if(name.length()==0){//文件名为空提示
            Toast.makeText(FileEdit.this,"File Name cannot be empty",Toast.LENGTH_SHORT).show();
        }
        else{
            if(deleteFile(name)) {//删除成功
                Toast.makeText(FileEdit.this,"Delete succesfully",Toast.LENGTH_SHORT).show();
            }else {//删除失败
                Toast.makeText(FileEdit.this,"Fail to Delete file",Toast.LENGTH_SHORT).show();
            }
        }
    }
});
```

实验中遇到的困难和解决思路：

1. 对于特殊要求：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。
   一开始想要直接重载返回键实现，后来发现还是会有 bug，之后使用最暴力的方法转跳之后直接 finish 主界面，此时按了返回键之后会直接返回 Home，在进入的时候也会直接到全新的主界面。之后看了一下实验文档发现也可以在 AndroidManifest.xml 中设置 noHistory 属性<activity android:noHistory="true"/>

Internal Storage 和 External Storage 的区别以及适用的场景：

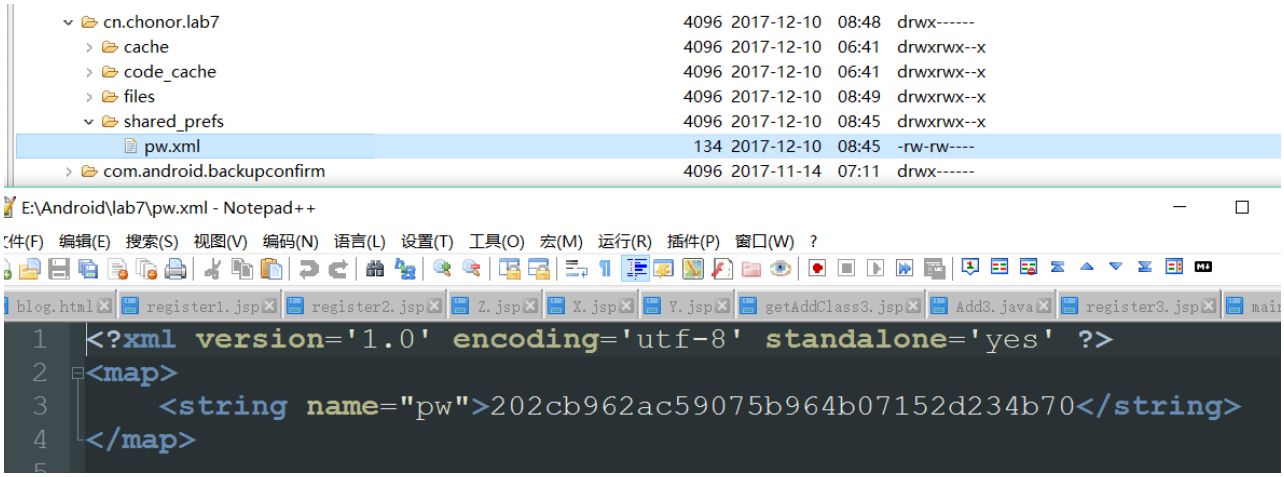
1. 内部存储 internal storage：
   内部存储位于系统中很特殊的一个位置，其总是可用的，如果将文件存储于内部存储中，那么文件默认只能被当前的应用访问到，一个应用所创建的所有文件都在和应用包名相同的目录下。当一个应用卸载之后，内部存储中的文件也被删除。如果我们想确保文件不被用户与其他应用所访问的最佳存储区域。
2. 外部存储 external storage：
   我们可以认为机身固有存储是内部存储，而扩展的 SD(TF)卡是外部存储，并不总是可用的，当取消挂载后，就无法对其进行访问了。外部存储中的公共文件是可以被自由访问，当应用被卸载之后，其卸载前创建的文件仍然保留。外部存储中的私有文件，也能被其他程序访问，外部存储上，卸载之后，这些文件也会被删除。类似于内部存储。外部存储是不需要严格的访问权限并且希望这些文件能够被其他应用所共享或者是允许用户访问时的最佳存储区域。

五、 课后实验结果

1. SharedPreferences 来保存密码时增加一个加密算法。
   此处加密算法使用简单的 MD5 加密匹配，虽然还是很容易解密但是至少在我们打开 xml 中不是明文密码。

```
public class MD5Utils {
    public static String md5Password(String password){
        StringBuffer sb = new StringBuffer();// 得到一个信息摘要器
        try {
            MessageDigest digest = MessageDigest.getInstance("md5");
            byte[] result = digest.digest(password.getBytes());// 把每一个 byte 做一个与运算 0xff
            for (byte b : result) {// 与运算
                int number = b & 0xff;
                String str = Integer.toHexString(number);
                if (str.length() == 1) {
                    sb.append("0");
                }
                sb.append(str);
            }
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return sb.toString();
    }
}
```

   在我们存入密码时调用，存入 MD5 加密过的密码。
   我们通过 Android Device Monitor 读取我们存入的密码，存入密码效果如下：



六、 实验思考及感想

这次实验非常简单，就是实现文件的存储，两种方式一种是使用 SharedPreferences，进行存储，另一种是使用 File 直接存储，实现也很简单，就是几句话的事情，也不用自己实现什么函数，直接调用即可。本次实验重点还是了解如何使用文件存储，虽然在 project 的时候已经用过了直接将整个数据库写入文件系统，避免了用 json 写入数据库缓慢的问题。