中山大学数据科学与计算机学院本科生实验报告

（2017 年秋季学期）

课程名称：手机应用平台开发　　任课教师： 刘宁

年级	15 级	专业（方向）	计应
学号	15331421	姓名	郑佳锋
电话	13719325472	Email	1773643139@qq.com
开始日期	11.23	完成日期	11.25

一、 实验题目

Lab8：多媒体播放器

二、 实现内容

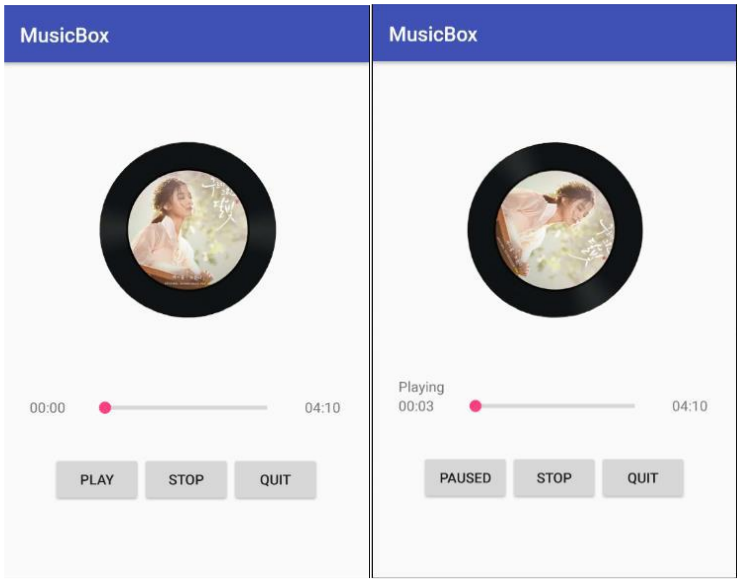
实现一个简单的播放器，要求功能有：

1. 播放、暂停，停止，退出功能；

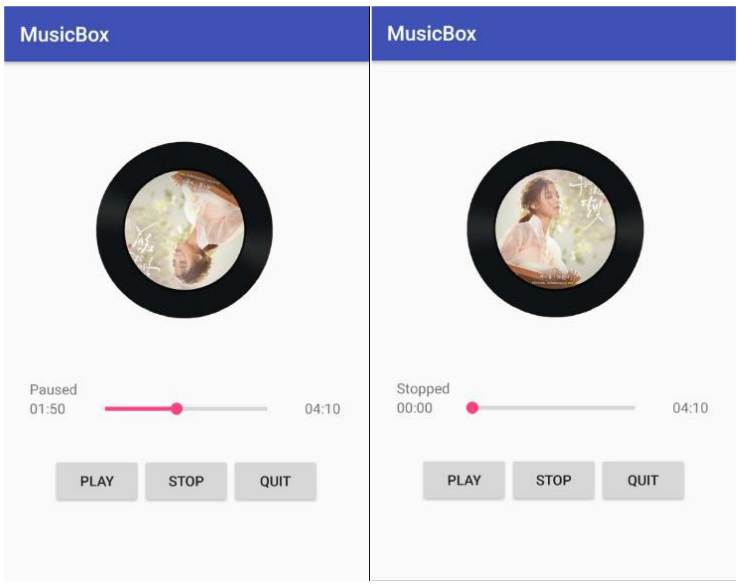
2. 后台播放功能；

3. 进度条显示播放进度、拖动进度条改变进度功能；

4. 播放时图片旋转，显示当前播放时间功能；



打开程序主界面　　开始播放

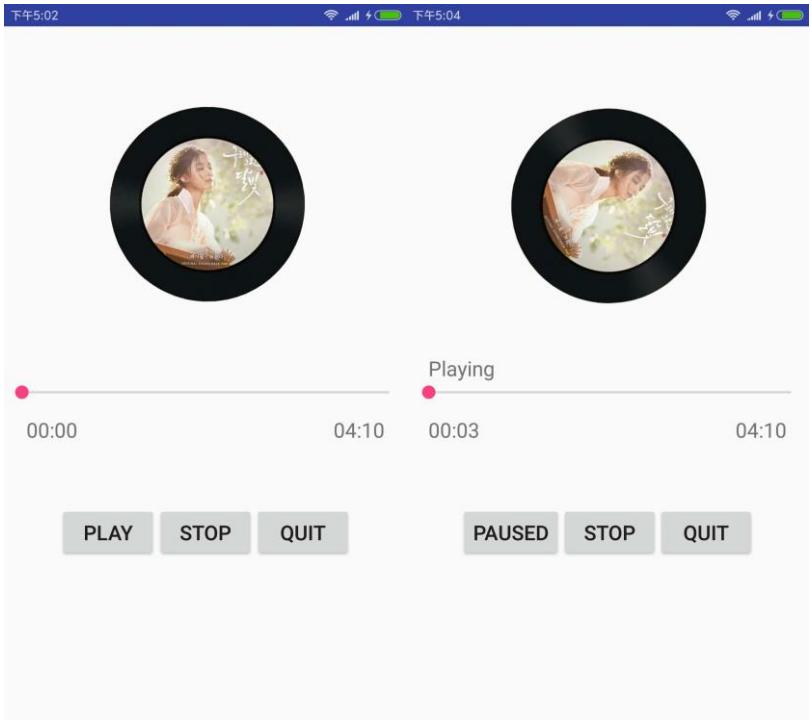


暂停　　停止

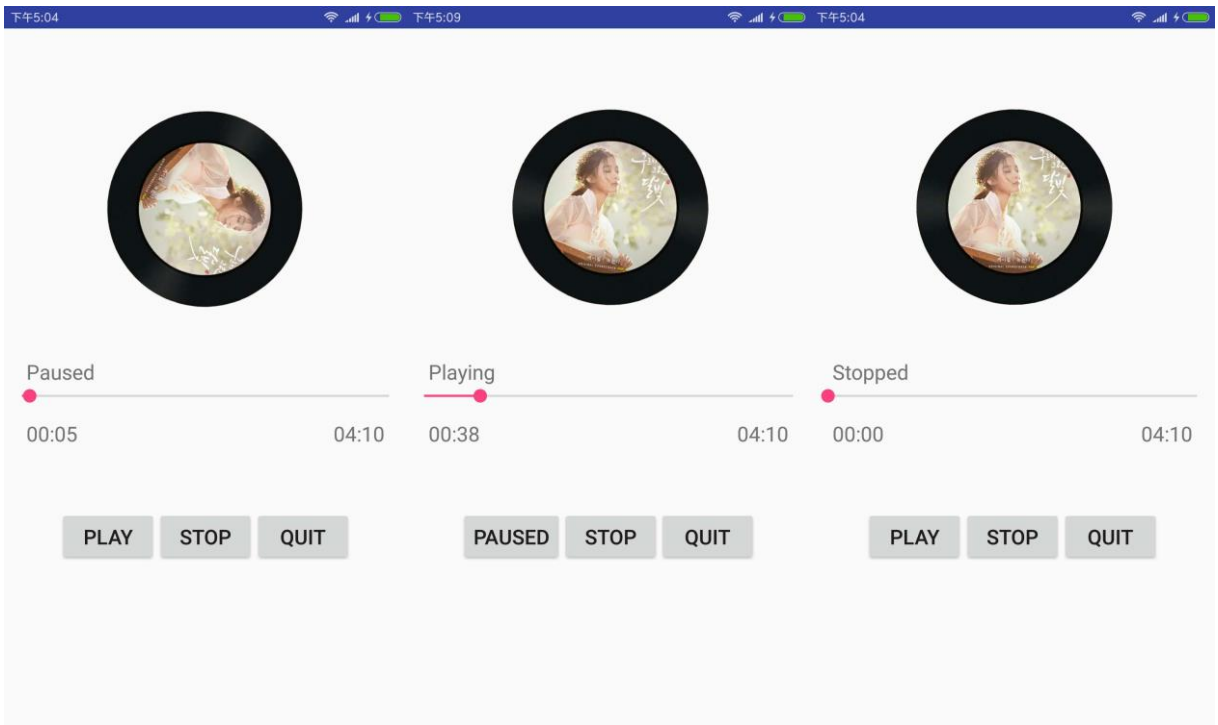
三、 实验结果

（1） 实验截图

打开程序主界面：　　开始播放：



暂停：　　继续播放并拖动：　　停止：



（2） 实验步骤以及关键代码

1.UI 设计：

首先放置旋转图片，整体居中，使用 guideline 让它位于距页面顶部 40%处。添加进度条 SeekBar，在进度条上下共设置 3 个 TextView，分别用来显示播放状态，播放时间和总时间。最后是三个按钮，使用链条 chain 将它们连成一个整体，再整体居中。对 3 个按钮设置监听器，当点击不同按钮时向 Service 发送不同 Action 的 intent，促发播放器的不同事件。

代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.zhengjiafeng.mediaplayer.MainActivity">
    <android.support.constraint.Guideline
        android:id="@+id/horizonal_guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.4"
        />
    <ImageView
        android:id="@+id/image_rotate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/image"
        app:layout_constraintRight_toLeftOf="parent"
        app:layout_constraintLeft_toRightOf="parent"
        app:layout_constraintBottom_toBottomOf="@+id/horizonal_guideline"
        />
    <SeekBar
        android:id="@+id/bar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        android:layout_marginTop="70dp"
        />
    <TextView
        android:id="@+id/show_state"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="45dp"
        />
    <TextView
        android:id="@+id/current"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="100dp"
```

```
        android:text ="00:00"
        />
    <TextView
        android:id="@+id/duration"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginRight="20dp"
        android:layout_marginTop="100dp"
        />
    <Button
        android:id="@+id/play"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="PLAY"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/stop"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="180dp"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        />
    <Button
        android:id="@+id/stop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="STOP"
        android:layout_marginTop="180dp"
        app:layout_constraintLeft_toRightOf="@+id/play"
        app:layout_constraintRight_toRightOf="@+id/quit"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        />
    <Button
        android:id="@+id/quit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="QUIT"
        android:layout_marginTop="180dp"
        android:layout_marginRight="50dp"
        app:layout_constraintLeft_toRightOf="@+id/stop"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/horizonal_guideline"
        />

</android.support.constraint.ConstraintLayout>
```

按钮点击事件：

```
@Override
public void onClick(View v) {
    Intent intent = new Intent(this,MyService.class);
    if (btPlay == v) {
        if(flag == 0) {
            intent.setAction(MyService.ACTION_PLAY);
            intent.putExtra("progress", bar.getProgress());
            btPlay.setText("PAUSED");
            show_state.setText("Playing");
            if(last_state == 0 || last_state == 2) {
                animator.start();
            }
            else if(last_state == 1){
                animator.resume();
            }
            flag = 1;
        }
        else if(flag == 1){
```

```
            intent.setAction(MyService.ACTION_PAUSE);
            btPlay.setText("PLAY");
            show_state.setText("Paused");
            animator.pause();
            flag = 0;
            last_state = 1;
        }
    } else if (btQuit == v) {
        intent.setAction(MyService.ACTION_STOP);
        MainActivity.this.finish();
    } else if (btStop == v) {
        intent.setAction(MyService.ACTION_STOP);
        btPlay.setText("PLAY");
        show_state.setText("Stopped");
        flag = 0;
        animator.end();
        bar.setProgress(0);
        current.setText("00:00");
        last_state = 2;
    }
    startService(intent);
    if(btStop == v){
        Intent intent1 = new Intent(this,MyService.class);
        intent1.setAction(MyService.ACTION_PLAY);
        startService(intent1);
    }
}
```

2.设计图片的动画：

使用类 ObjectAnimator 类，setScaleType 方法设置旋转中心，setDuration 设置每圈的时间，setRepeatCount 设置循环次数，当设为-1 时表示无限循环。当点击播放按钮时，需要判断上一次的点击状态，所以实验中用 last_state 变量保存上一次的状态，若上一次是暂停，则调用 resume 方法；若刚启动应用或者之前点击暂停，则调用 start 方法。点击暂停按钮时，调用 pause 方法；点击停止按钮时，调用 stop 方法。主要代码如下：

```
animator = ObjectAnimator.ofFloat(image_rotate, "rotation", 360);
image_rotate.setScaleType(ImageView.ScaleType.CENTER_CROP);
animator.setDuration(15000);
animator.setRepeatCount(-1);
```

2. 创建 MyService 类，继承自 Service，私有成员有一个 MediaPlayer。当接收到来自 MainActivity 的 intent 时，在 checkMediaPlayer 方法中通过异步加载的方式实例化 MediaPlayer，并设置监听器 setOnCompletionListener，当歌曲播放完之后重新开始播放。其中，获取播放资源时，在 MainActivity 中通过申请动态权限读取内置 sd 卡中音乐文件。主要代码如下：

实例化 MediaPlayer：

```
1.  private void checkMediaPlayer() {
2.       if (player == null) {
3.           // 完成多媒体的初始化
4.           player = new MediaPlayer();
5.           state = PlayerState.Idle;
6.           player.setAudioStreamType(AudioManager.STREAM_MUSIC);
7.
8.           // 判断 SDcard 正常挂载
9.           if (Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())) {

10.              try {
11.                  player.setDataSource(Environment
12.                          .getExternalStorageDirectory().getAbsolutePath()
13.                          + "/melt.mp3");
14.                  state = PlayerState.Initialized;
15.                  //异步准备
16.                  player.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
17.
18.                      @Override
19.                      public void onPrepared(MediaPlayer mp) {
20.                          state = PlayerState.Prepared;
21.                          // 设定进度条的 max 值
22.                          send(WHAT_DURATION, player.getDuration());
23.                          // 准备完成
24.                      }
25.                  });
26.                  //播放完成
27.                  player.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {

28.
29.                      @Override
30.                      public void onCompletion(MediaPlayer mp) {
31.                          state = PlayerState.PlaybackCompleted;
32.                          play(0);
33.                      }
34.                  });
35.
36.              } catch (Exception e) {
37.                  e.printStackTrace();
38.              }
39.          }
40.      }
41.  }
```

申请动态权限读取文件：

```
1.  public static void verifyStoragePermissions(Activity activity) {
2.
3.      try {
4.          //检测是否有写的权限
5.          int permission = ActivityCompat.checkSelfPermission(activity,
6.                  "android.permission.WRITE_EXTERNAL_STORAGE");
7.          if (permission != PackageManager.PERMISSION_GRANTED) {
8.              // 没有写的权限，去申请写的权限，会弹出对话框
9.              ActivityCompat.requestPermissions(activity, PERMISSIONS_STORAGE,REQUEST_EXTE
    RNAL_STORAGE);
10.         }
11.     } catch (Exception e) {
12.         e.printStackTrace();
13.     }
14. }
15.
```

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

2.使用 Ibinder 实现 MainActivity 和 Service 之间的通信。首先在 MainActivity 调用 bindService 方法绑定对应的 Service。bindService 的第二个参数是一个实现了 ServiceConnection 接口的类对象。该类的 onServiceConnected 方法中调用了 setHandler 方法。MainActiviyt 的 onDestroy 方法中解除绑定。在 Service 中同样设置了 handle，并重写了 onBind 方法。主要代码如下：

MainActivity 中与 IBinder 相关：

```
private PlayerConn conn;
private IMyBinder binder;

conn = new PlayerConn();
Intent intent = new Intent(this, MyService.class);
bindService(intent, conn, BIND_AUTO_CREATE);


    private class PlayerConn implements ServiceConnection {

      @Override
      public void onServiceConnected(ComponentName name, IBinder service) {
          binder = (IMyBinder) service;
          binder.setHandler(handler); //设定 handler
      }

      @Override
      public void onServiceDisconnected(ComponentName name) {

      }
  }

  @Override
  protected void onDestroy() {
      super.onDestroy();
      //解除服务
      unbindService(conn);
  }
```

Service 中与 IBinder 相关：

```
private class MyBinder extends Binder implements IMyBinder {
    public void setHandler(Handler handler) {
        MyService.this.handler = handler;
    }
}
@Override
public IBinder onBind(Intent intent) {
```

```
        return (IBinder) binder;
    }
```

3.播放过程进度条更新，拖动进度条实现跳跃。MainActivity 实现了 SeekBar.OnSeekBarChangeListener，该接口有几个主要方法如下：

```
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
        handler.removeMessages(MyService.WHAT_CURRENT);
        isDrag = true;
    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
            Intent intent = new Intent(this, MyService.class);
            if (flag == 0) {
                intent.setAction(MyService.ACTION_CHANGE);
                intent.putExtra("progress", bar.getProgress());
            }
            else{
                intent.setAction(MyService.ACTION_PLAY);
                intent.putExtra("progress", bar.getProgress());
            }
            startService(intent);
            isDrag = false;
        current.setText(calculateTime((int)bar.getProgress()/1000));
    }
```

onStartTrackingTouch 在进度条刚被拖动时调用，此时清除从 Service 发过来的待处理的信息。

OnStopTrackingTouch 在进度条完成拖动时调用，向 Service 发送 intent，进度条的长度作为参数。

当 Service 接收到进度条更新的消息并且处于播放状态时，调用 toPlay 方法。toPlay 中判断播放器是否准备完毕。若是，调用 play 方法；否则异步等待直至资源加载完成。Play 方法中，采用 seekTo 方法跳到进度条的位置。此时新建一个线程，不断获取当前的播放位置并发送到 MainActivity。主要代码如下：

toPlay 方法：

```
    private void toPlay(int pos) {
        checkMediaPlayer();
        if (state == PlayerState.Initialized) {
            state = PlayerState.Preparing;
            player.prepareAsync();
        } else {
            play(pos);
        }
    }
```

play 方法：

```
1.  private void play(int pos) {
2.       if (state == PlayerState.Started || state == PlayerState.Prepared || state == Playe
    rState.Initialized
3.               || state == PlayerState.Paused
4.               || state == PlayerState.PlaybackCompleted) {
5.           player.seekTo(pos);
6.           player.start();
7.           state = PlayerState.Started;
8.           new Thread() {
9.               public void run() {
10.                  while (state == PlayerState.Started) {
11.                      send(WHAT_CURRENT, player.getCurrentPosition());
12.                      try {
13.                          Thread.sleep(500);
14.                      } catch (InterruptedException e) {
15.                          e.printStackTrace();
16.                      }
17.                  }
18.              }
19.          }.start();
20.      } else {
21.      }
22.  }
```

4.进度条更新，并设置播放时间和总时间。当播放资源加载完毕后，Service 会向 MainActivity 发送歌曲的 Duration；播放过程中，Service 也会不断发送当前的播放时间。所有这些信息，在 MainActivity 中通过 handle 进行处理。

```
1.  private  Handler handler = new Handler() {
2.       public void handleMessage(Message msg) {
3.           switch (msg.what) {
4.               case MyService.WHAT_DURATION:
5.                   bar.setMax(msg.arg1);
6.                   dur = msg.arg1;
7.                   duration.setText(calculateTime(msg.arg1/1000));
8.                   break;
9.               case MyService.WHAT_CURRENT:
10.                  if (!isDrag) {
11.                      bar.setProgress(msg.arg1);
12.                      current.setText(calculateTime(msg.arg1/1000));
13.                  }
14.                  break;
15.          }
16.      }
17.  };
```

使用 calculateTime 将时间转化为想要的格式：

```
1.  public String calculateTime(int time) {
2.        int minute;
3.        int second;
4.
5.        String strMinute;
6.
7.        if (time >= 60) {
8.            minute = time / 60;
9.            second = time % 60;
10.
11.           //分钟在 0-9
```

```
            if(minute>=0&&minute<10)
            {
                //判断秒
                if(second>=0&&second<10)
                {
                    return "0"+minute+":"+"0"+second;
                }else
                {
                    return "0"+minute+":"+second;
                }
            }else
            //分钟在 10 以上
            {
                //判断秒
                if(second>=0&&second<10)
                {
                    return minute+":"+"0"+second;
                }else
                {
                    return minute+":"+second;
                }
            }

        } else if (time < 60) {
            second = time;
            if(second>=0&&second<10)
            {
                return "00:"+"0"+second;
            }else
            {
                return "00:" + second;
            }

        }
        return null;
    }
```

（3） 实验遇到困难以及解决思路

本次实验遇到的问题非常之多，仅列举一二。

首先是动画的问题。一开始不知道有 resume 这种方法，所以每次都是 start，导致动画每次都从头开始。后来知道 resume 方法后，琢磨着通过获取当前的播放时间来确定是 resume 和 start，发现行不通因为还有暂停状态下的快进快退要考虑。最后终于想出来使用一个变量来记录上一个状态的方法。当上次是暂停时，使用 resume；当第一次启动或者上次是 stop 时，使用 start。

另一个问题是同步异步、线程的问题。当点击开始按钮或者拖动进度条，然后迅速点击 stop 按钮时，常常会出现闪退。一开始百思不得其解，后来查看错误报告，出现的错误是 IllegalState，上网查发现有人遇到这种问题，于是反思自己的程序，后来发现原来

在 stop 方法中，有可能此时播放器还没加载好，所以是一个空对象，此时只需要对 player 的状态进行一下判断即可。如下：

```
1.  private void stop() {
2.      if (state == PlayerState.Started || state == PlayerState.Prepared
3.              || state == PlayerState.Paused
4.              || state == PlayerState.PlaybackCompleted) {
5.          try {
6.              player.stop();
7.          } catch (IllegalStateException e) {
8.              // TODO 如果当前 java 状态和 jni 里面的状态不一致，
9.              //e.printStackTrace();
10.             player = null;
11.             player = new MediaPlayer();
12.             player.stop();
13.         }
14.         player.release();
15.         player = null;
16.         state = PlayerState.End;
17.         send(WHAT_CURRENT,0);
18.     } else {
19.     }
20. }
```

还有一个问题是在项目基本正常工作后，安装到手机。播放时，若点击 Home 键再返回，可以保持原来的状态，但是点击返回键后在回到应用，发现进度条等都发生了错误。刚开始以为是自己的 Service 没写好，但是想想点击 Home 键是正常的。所以查了一下，原来返回键默认是关闭应用，但是后台程序不会被关，所以再次点开应用会出现前后台不一致的情况。所以有一种方法可以将返回键的功能变得跟 Home 键一样，这样就不会退出应用了。

```
1.  @Override
2.  public boolean onKeyDown(int keyCode, KeyEvent event) {
3.
4.      if (keyCode == KeyEvent.KEYCODE_BACK) {
5.          Intent home = new Intent(Intent.ACTION_MAIN);
6.          home.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
7.          home.addCategory(Intent.CATEGORY_HOME);
8.          startActivity(home);
9.          return true;
10.     }
11.     return super.onKeyDown(keyCode, event);
12. }
```

四、 实验思考及感想

本次实验设计线程和后台服务问题，容易出现各种奇奇怪怪的问题。花在调试上的时间非常多。所以特地去看了一些调试的方法，包括查看 log 等等。总得来说，每次的 android 作业都有许多难点要攻克，在这个过程中也学到了很多东西！