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	大三	专业（方向）	移动互联网
学号	15352408	姓名	张镓伟
电话	13531810182	Email	709075442@qq.com
开始日期	2017.11.21	完成日期	2017.11.26

一、 实验题目

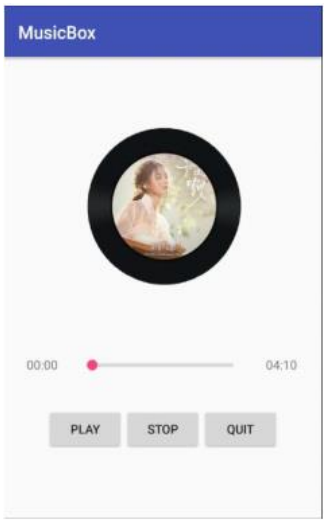
服务与多线程—简单音乐播放器

二、 实验目的

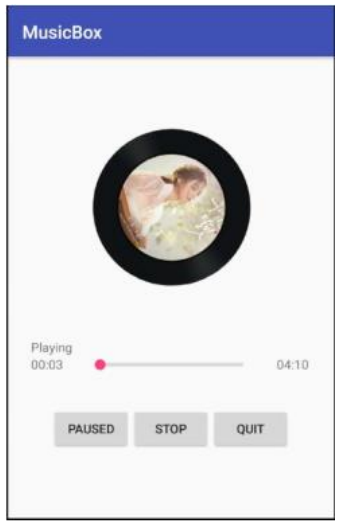
1. 学会使用 MediaPlayer。
2. 学会简单的多线程编程，使用 Handle 更新 UI。
3. 学会使用 Service 进行后台工作。
4. 学会使用 Service 与 Activity 进行通信。

三、 实验内容

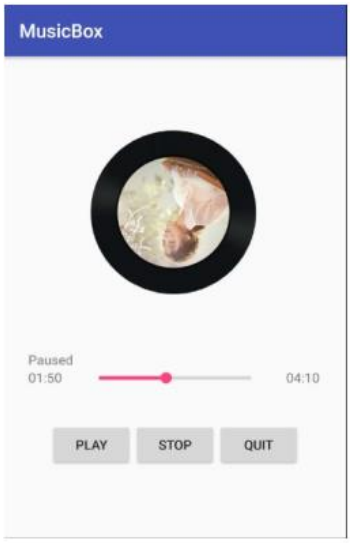
实现一个简单的播放器，要求功能有:
1. 播放、暂停、停止，退出功能;
2. 后台播放功能;
3. 进度条显示播放进度、拖动进度条改变进度功能;
4. 播放时图片旋转，显示当前播放时间功能;



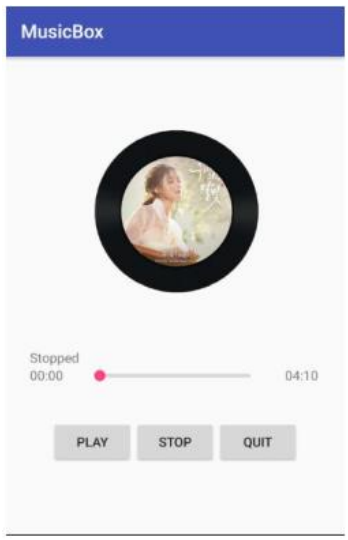
打开程序主页面



开始播放



暂停

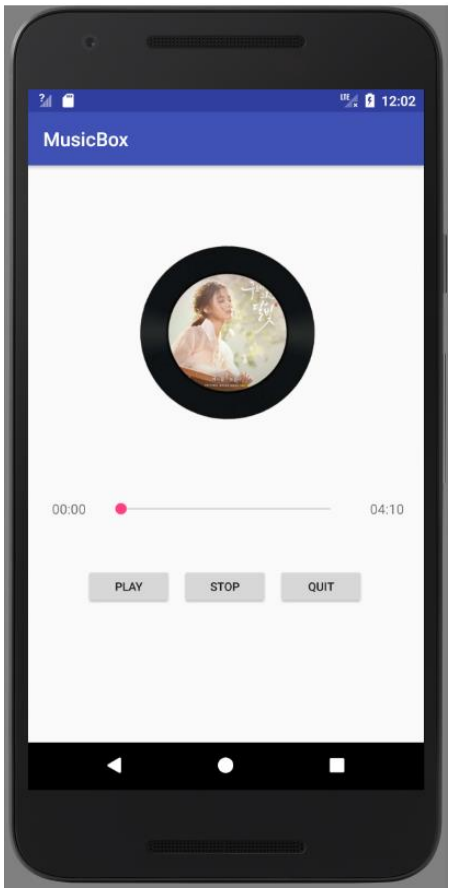


停止

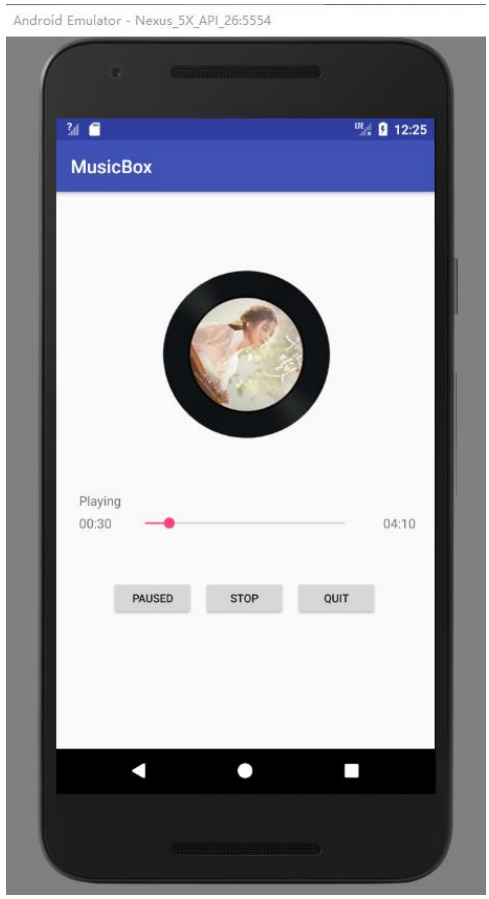
四、 课堂实验结果

（1） 实验截图

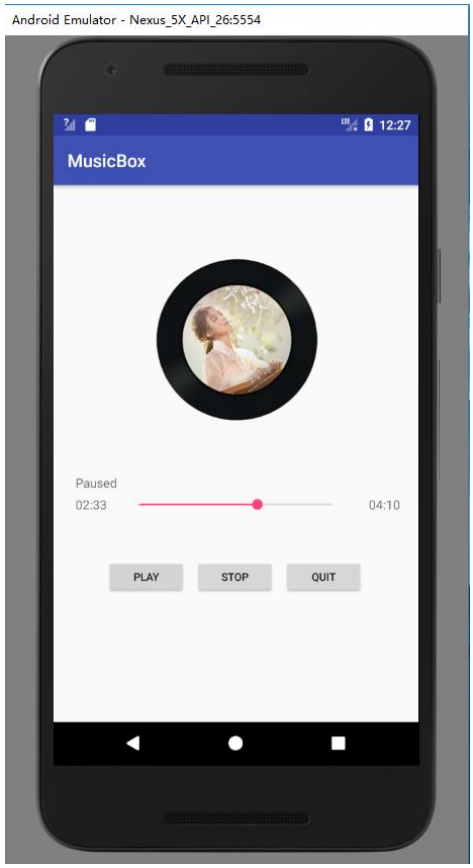
1. 打开主程序界面



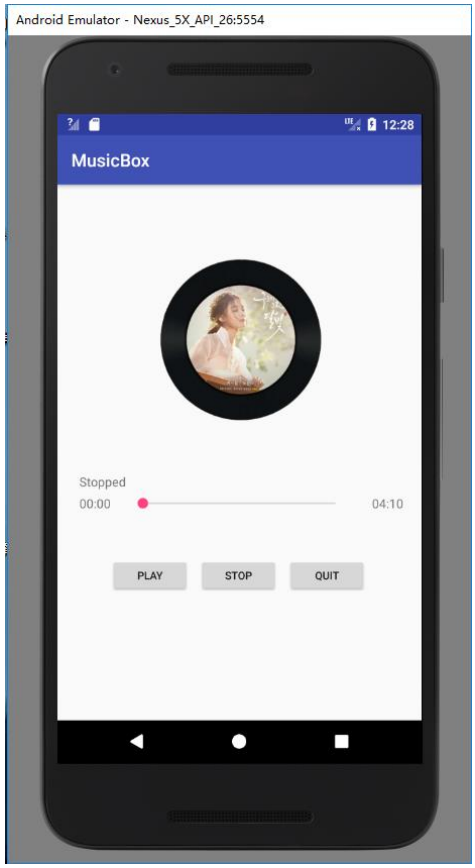
2.开始播放



3. 暂停

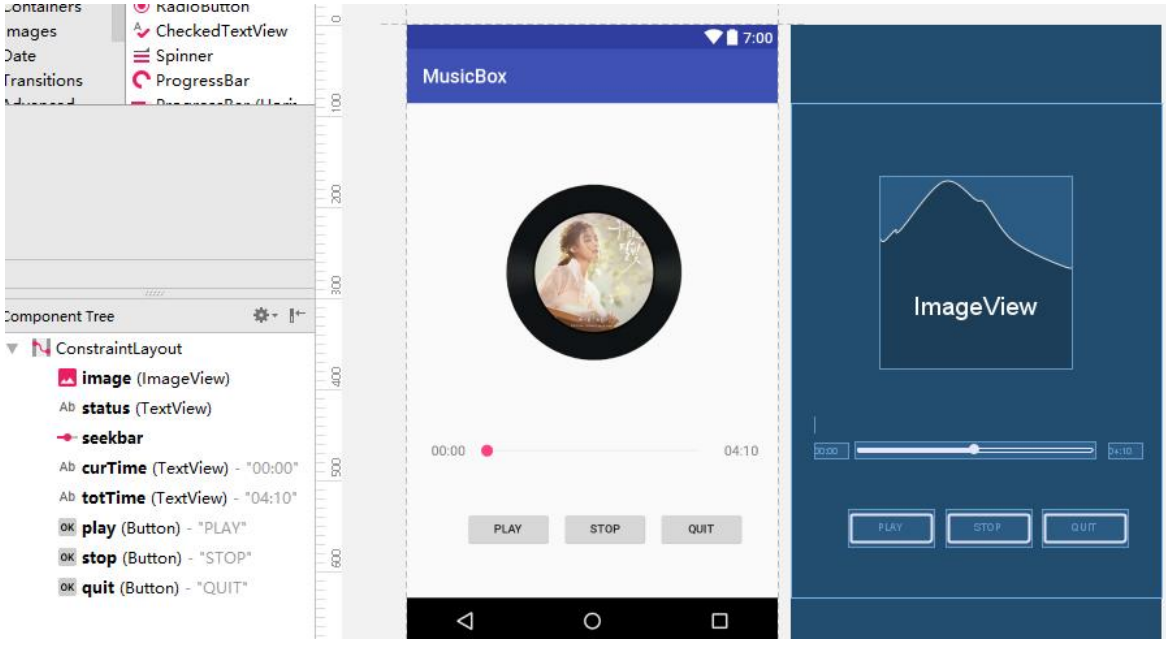


4.停止



（2） 实验步骤以及关键代码

1.XML 布局代码



由上图可以知道，布局使用了约束布局，一共包括一个 ImageView 显示图片。当前时间和总时间以及默认空白内容的显示当前状态的是 TextView。三个按钮是 Button、进度条是 seekbar。

XML 代码如下：

ImageView:

```
<ImageView
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:id="@+id/image"
    android:src="@mipmap/image"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="75dp" />
```

SeekBar:

```
<SeekBar
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:id="@+id/seekbar"
    android:max="0"
    android:progress="0"
    android:secondaryProgress="0"
    app:layout_constraintTop_toBottomOf="@id/image"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginTop="75dp"/>
```

TextView(从上到下依次是状态、当前时间、总时间)：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/status"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@id/image"
    android:layout_marginTop="50dp"
    android:layout_marginLeft="25dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/curTime"
    app:layout_constraintBottom_toBottomOf="@id/seekbar"
    app:layout_constraintTop_toTopOf="@id/seekbar"
    app:layout_constraintLeft_toLeftOf="@id/status"
    android:text="00:00" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/totTime"
    app:layout_constraintBottom_toBottomOf="@id/seekbar"
    app:layout_constraintTop_toTopOf="@id/seekbar"
    app:layout_constraintRight_toRightOf="parent"
    android:layout_marginRight="20dp"
    android:text="04:10" />
```

Button (从上到下依次是 PLAY、STOP、QUIT)，在这三个 Buttion 的 xml 代码中顺便指定了 onClick 方法作为点击触发事件：

```
<Button
    android:layout_width="90dp"
    android:layout_height="40dp"
    android:id="@+id/play"
    android:text="PLAY"
    android:textSize="12sp"
    app:layout_constraintTop_toBottomOf="@id/image"
    app:layout_constraintLeft_toLeftOf="parent"
    android:layout_marginTop="145dp"
    android:layout_marginLeft="60dp"
    android:onClick="onClick"/>

<Button
    android:layout_width="90dp"
    android:layout_height="40dp"
    android:id="@+id/stop"
    android:text="STOP"
    android:textSize="12sp"
    app:layout_constraintTop_toBottomOf="@id/image"
    app:layout_constraintLeft_toRightOf="@id/play"
    android:layout_marginTop="145dp"
    android:layout_marginLeft="10dp"
    android:onClick="onClick"/>

<Button
    android:layout_width="90dp"
    android:layout_height="40dp"
    android:id="@+id/quit"
    android:text="QUIT"
    android:textSize="12sp"
    app:layout_constraintTop_toBottomOf="@id/image"
    app:layout_constraintLeft_toRightOf="@id/stop"
    android:layout_marginTop="145dp"
    android:layout_marginLeft="10dp"
    android:onClick="onClick"/>
```

2.JAVA 代码

MusicService 部分，

按照文档提示在其中的 Binder 类的 onTransact 方法里写好不同情况对 Music 的处理。其中：

播放按钮：判断当前状态，决定是开始还是暂停

停止按钮：停止播放、将歌曲进度归 0

退出按钮：释放音乐资源

界面刷新：获取当前播放进度、通过 reply 返回给主 activity 的 handle 去更新 UI 的进度条。

拖动进度条：从 data 中获取人为拖动进度条的位置，调整音乐的当前播放位置。

初始情况：获取音乐资源、设置无限循环、使音乐处于就绪状态。

```
public class MyBinder extends Binder {
    @Override
    protected boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws RemoteException {
```

```
        if (code == 101){ //播放按钮，服务处理函数
            if (mp.isPlaying()) mp.pause();
            else mp.start();
        }
        else if (code == 102){//停止按钮，服务处理函数
            mp.stop();
            try {
                mp.prepare();
                mp.seekTo(0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        else if (code == 103) mp.release();//退出按钮
        else if (code == 104) reply.writeInt(mp.getCurrentPosition());//界面刷新
        else if (code == 105) mp.seekTo(data.readInt());//拖动进度条
        else{//初始情况，绑定音乐资源
            try{
                AssetManager AM = getAssets();
                AssetFileDescriptor AFD = AM.openFd("melt.mp3");
                mp.setDataSource(AFD.getFileDescriptor(), AFD.getStartOffset(), AFD.getLength());
                mp.prepare();
                mp.setLooping(true);
                reply.writeInt(mp.getDuration());
                reply.writeInt(mp.getCurrentPosition());
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return super.onTransact(code, data, reply, flags);
    }
}
```

最后要返回一个通信通道给 service

```
@Override
public IBinder onBind(Intent intent) {
    // Return the communication channel to the service.
    return mBinder;
}
```

MainActicity 部分:

动画初始化部分、绑定 image、设置旋转方式、时间、变化率、循环次数：

```
//初始化动画
animation = ObjectAnimator.ofFloat(image, "rotation", 0.0f, 360.0f);
//第一个参数是空间，第二个是变化方式，第三个是可变长参数，第4个是变化角度
animation.setDuration(10000); //设定转一圈的时间
animation.setInterpolator(new LinearInterpolator());//定义动画的变化速率，这里是线性
animation.setRepeatCount(Animation.INFINITE); //设定无限循环
```

Service 与 bindService 的绑定：

```
//bindService 成功后回调 onServiceConnected 函数，通过 IBinder 获取 Service 对
// 象，实现 Activity 与 Service 的绑定
sc = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        mBinder = service;
        try {
            int code = 106;
            Parcel data = Parcel.obtain();
            Parcel reply = Parcel.obtain();
            mBinder.transact(code, data, reply, 0);
            seekBar.setMax(reply.readInt());  // 设置最大时长
            seekBar.setProgress(reply.readInt());  // 设置当前进度为0
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    @Override
    public void onServiceDisconnected(ComponentName name) {sc = null;}
};

//Activity 启动时绑定 Service
Intent intent = new Intent(this, MusicService.class);
startService(intent);
bindService(intent, sc, Context.BIND_AUTO_CREATE);
```

SeekBar 的监听，在进度条被拖动时改变音乐播放的进度：

```
//设置seekbar监听器
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        //该方法拖动进度条进度改变的时候调用
        if (fromUser){//用户触发
            try {
                int code = 105;
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                data.writeInt(progress);
                mBinder.transact(code, data, reply, 0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}//该方法拖动进度条开始拖动的时候调用
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {}//该方法拖动进度条停止拖动的时候调用
});
```

onClick 方法，设置点击 button 的事件，根据点击的 button，设置不同的状态值，后续会根据状态值在 Handler 中进行处理：

```
public void onClick(View v){
    if (v.getId() == R.id.play){
        if (status.getText().toString().equals("Playing")) sta = 2;
        else sta = 1;
    }
    else if (v.getId() == R.id.stop) sta = 3;
    else sta = 4;
}
```

Handle 根据状态值更新 UI、同时要通过 IBunder 与 Service 通信：

```
final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (msg.what < 3) {
            if (msg.what == 1) {//播放
                play.setText("PAUSED");
                status.setText("Playing");
                if (animation.isPaused()) animation.resume();
                else animation.start();
            }
            else {//暂停
                play.setText("PLAY");
                status.setText("Paused");
                animation.pause();
            }
            try {
                int code = 101;
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                mBinder.transact(code, data, reply, 0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        else if (msg.what == 3) {//停止
            seekBar.setProgress(0);
            play.setText("PLAY");
            status.setText("Stopped");
            animation.setCurrentPlayTime(0);
            animation.cancel();
            try {
                int code = 102;
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                mBinder.transact(code, data, reply, 0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        else if (msg.what == 4) {//退出
            try {
                int code = 103;
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                mBinder.transact(code, data, reply, 0);
            } catch (Exception e) {
                e.printStackTrace();
            }
            //停止服务时要解除绑定
            unbindService(sc);
            sc = null;
            try{
                MainActivity.this.finish();
                System.exit(0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
            sta = 5;
            try {//音乐进度是要一直更新的
                int code = 104;
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                mBinder.transact(code, data, reply, 0);
                int progress = reply.readInt();
                seekBar.setProgress(progress);
                curTime.setText(time.format(progress));
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
```

定义一个新线程个给 handler 更新线程：

```
Thread mThread = new Thread() {
    @Override
    public void run() {
        while (true) {
            try {
                Thread.sleep(100);
            } catch (Exception e) {
                e.printStackTrace();
            }
            mHandler.obtainMessage(sta).sendToTarget();
        }
    }
};
mThread.start();
```

（3） 实验遇到困难以及解决思路

1. 一开始不知道怎么弄唱片旋转、查找了资料后知道要使用 ObjectAnimator 类去定义动画

2.写了 ObjectAnimator 类定义动画之后还是不会转，仔细排查之后发现定义操作动作的状态值传错了。

3.文档没有给出 SeekBar 的使用方法、查找资料后知道了可以通过设置 setOnSeekBarChangeListener 来进行事件监听、它一般有三种重要的方法：

(1)、onStartTrackingTouch 方法

该方法拖动进度条开始拖动的时候调用。

(2)、onStopTrackingTouch 方法

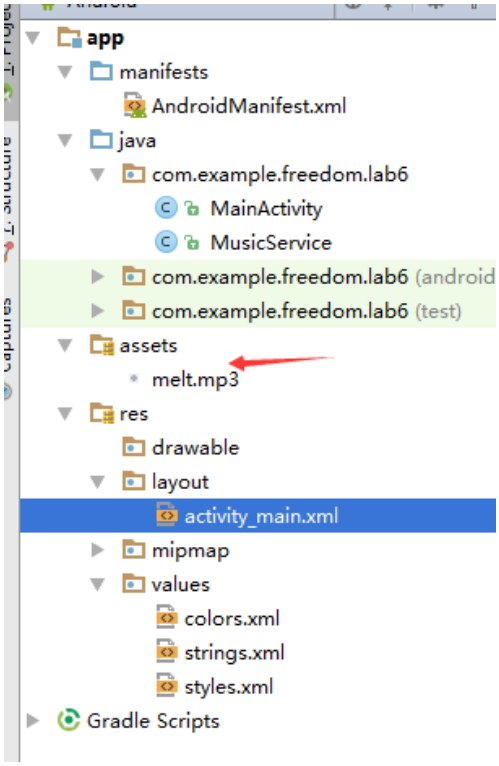
该方法拖动进度条停止拖动的时候调用

(3)、onProgressChanged

该方法拖动进度条进度改变的时候调用

五、 课后实验结果

文档中音乐是要另外导入的，但是我觉得应该要可以跟 apk 一起打包才比较方便，这样就不会像文档中写的那样那么麻烦去获取音乐文件的路径。经过百度查找资料，我知道了可以将 mp3 文件放入 assets 文件中：



然后通过 AssetManager 类获取资源管理器，最后使用 AssetFileDescriptor 去获取文件：

```
AssetManager AM = getAssets();
AssetFileDescriptor AFD = AM.openFd("melt.mp3");
mp.setDataSource(AFD.getFileDescriptor(), AFD.getStartOffset(), AFD.getLength());
```

六、 课后思考及感想

这次实验我学会了用 MediaPlayer 进行音乐播放、也学会了简单的多线程编程，使用 Handle 更新 UI。知道了如何使用 Service 进行后台工作。到现在我觉得自己已经可以写一个简单的多线程 app 了。希望随着实验的继续进行我能得到更多的知识、掌握更多的技巧。