实验五：服务与多线程——简单音乐播放器

13331388　庄梓嘉

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。

实验内容

实现一个简单的播放器，要求功能有：

1.播放、暂停功能；

2.进度条显示播放进度、拖动进度条改变进度功能；

3.后台播放功能；

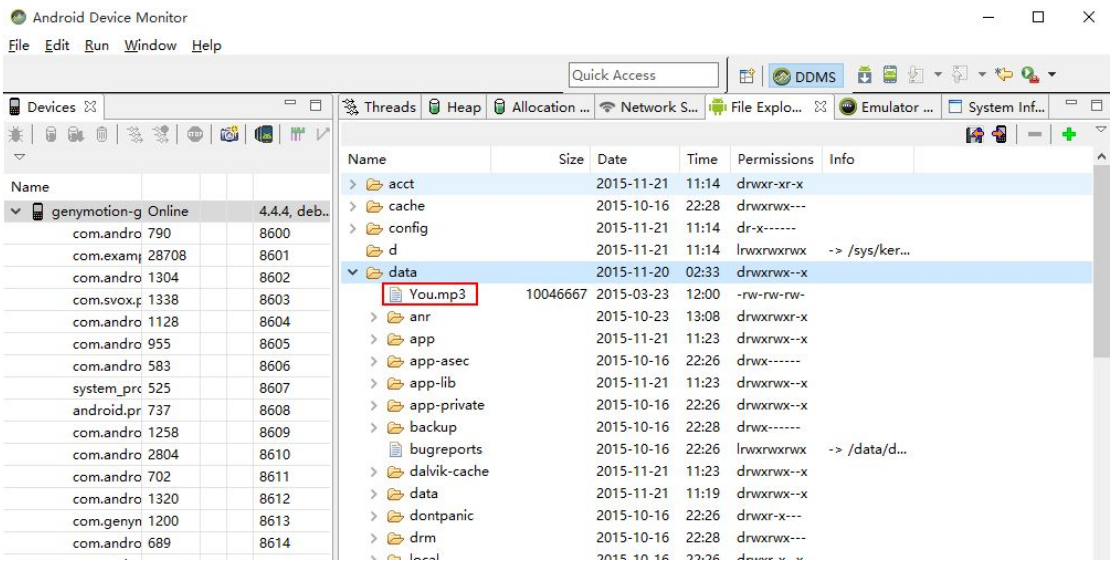
4.停止功能；

5.退出功能。

【拓展项】在原实验基础上，使用 Handler 实时更新 UI。

实验步骤

1.添加音频文件，打开 Android Device Monitor， 打开 file explorer 选择 data 文件夹点击右上角导入文件，将“You.mp3”导入。



2.创建一个 Service 类文件“MusicService”，在里面添加 Binder 来保持 Activity 和 Service 的通信。

```
public final IBinder binder = new MyBinder();
public class MyBinder extends Binder {
    MusicService getService() {return MusicService.this;}
}
```

3.在 MainActivity.java 中添加 bindService 保持 Service 的通信。

```
//Activity启动时绑定MusicService
private void bindServiceConnection() {
    Intent intent = new Intent(MainActivity.this, MusicService.class);
    bindService(intent, sc, BIND_AUTO_CREATE);
}

//bindService成功后回调onServiceConnection函数，通过IBinder获取musicService对象
//实现Activity与MusicService的绑定
private ServiceConnection sc = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        musicService = ((MusicService.MyBinder) service).getService();
    }
    @Override
    public void onServiceDisconnected(ComponentName name) { musicService = null; }
};
```

4.在 MusicService 里添加 MediaPlayer 和添加音频文件路径，并依次添加播放/暂停、停止、销毁回收等代码。


```
//添加MediaPlayer
public static MediaPlayer mp = new MediaPlayer();
@Override
public void onCreate() {
    super.onCreate();
}

//添加音频文件路径
public MusicService() {
    try {
        mp.setDataSource("/data/You.mp3");
        mp.prepare();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

//播放/暂停
public void playOrPause() {
    if (mp.isPlaying()) {
        mp.pause();
    } else {
        mp.start();
    }
}
//停止
public void stop() {
    if (mp != null) {
        mp.stop();
        try {
            mp.prepare();
            mp.seekTo(0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
//销毁回收
public void onDestroy() {
    mp.stop();
    mp.release();
    super.onDestroy();
}
```

5.在 MainActivity 里先定义简单日期格式，用 time.format 的格式来显示所需要的数据。

```
//定义简单日期格式，用time.format的格式来显示所需要的数据
private SimpleDateFormat time = new SimpleDateFormat("m:ss");
```

6.定义 Handler，监听滑动条的进度变化

```
//定义Handler，监听滑动条的进度变化
Handler handler = new Handler();
Runnable r = new Runnable() {
    @Override
    public void run() {
        textView.setText(time.format(musicService.mp.getCurrentPosition()) + "/" + time.format(musicService.mp.getDuration()));
        seekBar.setProgress(musicService.mp.getCurrentPosition());
        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                if (fromUser) {
                    musicService.mp.seekTo(seekBar.getProgress());
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
        handler.postDelayed(r, 100);
    }
};
```

7.播放器三个按钮功能实现。

播放/暂停：

```
//播放/暂停
btn.setOnClickListener((v) -> {
        musicService.playOrPause();
        textView.setText("OK");
        if (musicService.mp.isPlaying()) {
            state.setText("Playing");
        } else {
            state.setText("Pausing");
        }
        textView.setText(time.format(musicService.mp.getCurrentPosition()) + "/" + time.format(musicService.mp.getDuration()));
        seekBar.setProgress(musicService.mp.getCurrentPosition());
        seekBar.setMax(musicService.mp.getDuration());
        handler.post(r);
});
```

停止：

```
//停止
btn2.setOnClickListener((v) -> { musicService.stop(); });
```

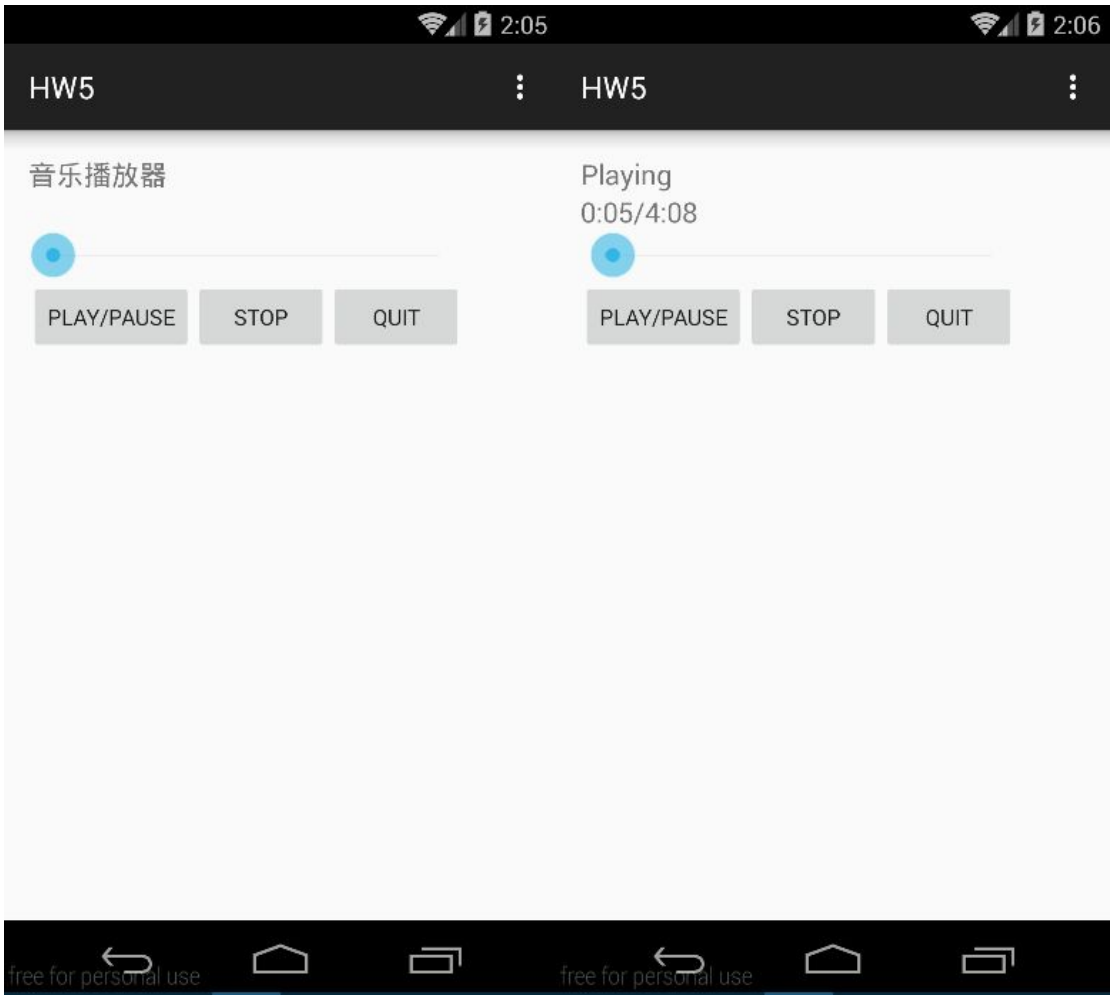
退出：

```
//退出
btn3.setOnClickListener((v) -> {
        handler.removeCallbacks(r);
        unbindService(sc);
        try {
            //MainActivity.this.finish();
            System.exit(0);
        } catch (Exception e) {
            e.printStackTrace();
        }
});
```

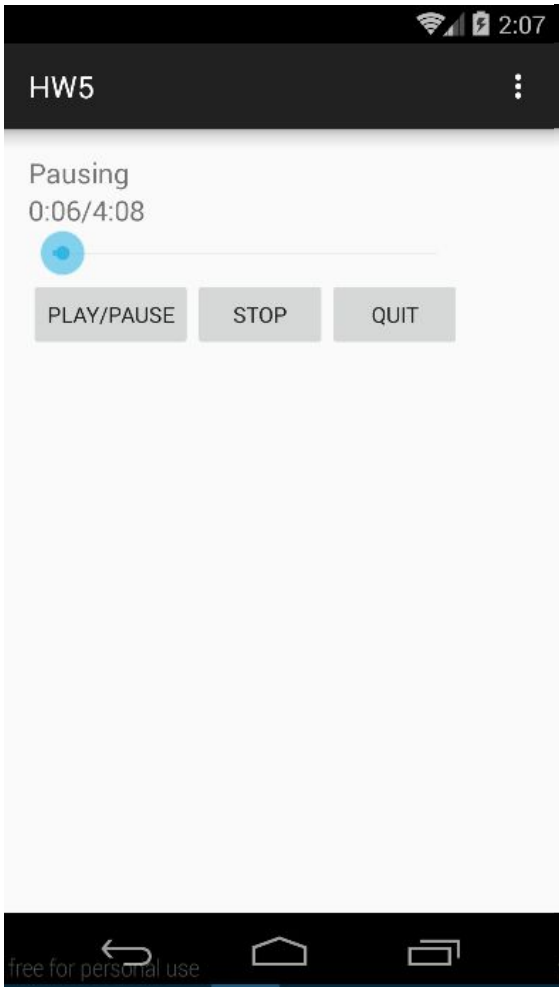
实验结果

初始界面：

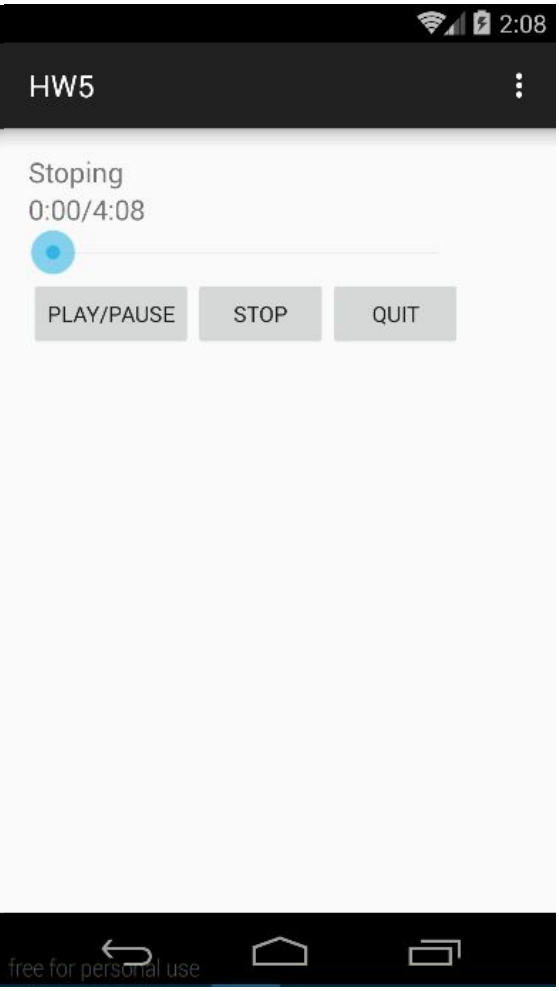
播放：



暂停：



停止：



参考资料

Android Service 使用方法

http://www.oschina.net/question/54100_31826

android 中 finish()与 system.exit(0)的区别

http://blog.csdn.net/xuyide54321/article/details/7049297

Android 中的 Service 全面总结

http://www.cnblogs.com/newcj/archive/2011/05/30/2061370.html

实验总结

本次实验主要是学会 MediaPlayer、Service 以及使用 Handler 更新 UI。知识点比较多。需要参考很多网上资料才能理解。根据实验文档，起码要顺理几次才知道哪些代码应用在哪一块。不然会造成一定的混乱，严重影响到程序。在理解代码的时候，对于 unbindService 执行后会不会继续执行后台程序有点困惑，然后查阅了许多资料之后得知，

```
MusicService    MusicServiceActivity
MusicService    MusicSerice onCreate()
MusicService    MusicSerice onStart()
MusicService    MusicSerice onDestroy()
MusicService    MusicSerice onCreate()
MusicService    MusicSerice onBind()
MusicService    MusicSerice onUnbind()
MusicService    MusicSerice onDestroy()
```

正常的 Service 生命周期应该是如上图这样的 ,unbindService 之后还是会继续在后台执行程序，只有当执行了 onDestroy()才会终止这个 Service。而 finish()只是结束当前这个 Avtivity，system.exit(0)才是终止整个进程，如果只执行了 finish()并没有终止这个程序，下次打开这个程序的时候会报错。在查阅资料之后，对比本次实验代码，感觉拓展项使用 Handle 实时更新 UI 和这次使用到的并没有什么区别，可能只是延时的差别，或许没理解清楚实时更新 UI，希望下次拓展项要求可以清楚明确一些。还有因为同学有问到怎样把音频文件导入真机，试验了一下实验文档给的方法，不可行，然后在网络上找到一个版本，是把音频文件放到 src 的文件夹里的，然后在创建 MediaPlayer 的时候就把音乐添加进去了。

```
mPlayer = MediaPlayer.create(getApplicationContext(), R.raw.music);
```

至于其他添加多首音乐或者添加到 SD 的情况以后有机会接触到再去发掘一下方法。