中山大学数据科学与计算机学院软件工程（移动信息工程）本科生实验报告

移动应用开发实验报告

姓名	学号	班级	电话	邮箱	日期
蔡烨	15352010	1501班	15989010669	caiy29@mail2.sysu.edu.cn	2017//12/3

1. 实验题目

服务与多线程--简单音乐播放器

2. 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 √
5. 学会申请动态权限读取内置sd卡中音乐文件 √

3. 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

4. 实验过程

1. 写xml文件，用约束布局，把排版排好
2. 新建class文件，写MusicService.java文件，实现基本的音乐播放器功能
    1. 重写onCreate函数，在第一次启动服务的时候，设置好音乐文件的地址，循环播放以及就绪

```
super.onCreate();
try{
    mp = new MediaPlayer();
//    mp.setDataSource(Environment.getExternalStorageDirectory()+"/Download/melt.mp3"); //手机7.1
//    mp.setDataSource(Environment.getExternalStorageDirectory()+"/Music/melt.mp3"); //手机4.4
    mp.setDataSource("/storage/0D08-3902/melt.mp3"); //虚拟机高版本
//    mp.setDataSource("/data/melt.mp3");  //虚拟机低版本
    mp.prepare();
    mp.setLooping(true);
```

    2. 重写onStartCommand函数，每一次启动服务都让文件处于就绪状态

```
public int onStartCommand(Intent intent,int flags,int startId){
    try{
        mp.prepare();
    }catch (IOException e){
        e.printStackTrace();
    } catch (IllegalStateException e){
        e.printStackTrace();
    }
    return super.onStartCommand(intent,flags,startId);
```

3. 定义类mBinder，使用transact和onTransact函数在service和activity之间进行信息传递。重写onTransact函数，根据接收到的不同code，做不同的响应操作，暂停/播放音乐，停止音乐，退出，返回当前音乐文件的播放位置、长度，改变音乐的播放位置。几乎所有的用户交互（按键、拖动）都是在这个函数中进行响应。

```
switch (code){
    case 101: //播放
        if(mp.isPlaying())
            mp.pause();
        else{
            mp.start();
            reply.writeInt(mp.getCurrentPosition());
        }
        break;
    case 102: //停止
        if(mp != null){
            mp.pause();
            mp.seekTo(0);
        }
        reply.writeInt(mp.getCurrentPosition());
        break;
    case 103: //退出
        mp.stop();
        break;
    case 104: //获取当前播放位置
        if(mp!=null){
            reply.writeInt(mp.getCurrentPosition());
            reply.writeInt(mp.getDuration());
        }
        break;
    case 105: //进度条拖动
        mp.seekTo(data.readInt());
        break;
}
return super.onTransact(code,data,reply,flags);
```

4. 重写onDestroy函数和onBind函数，后者返回的是自身的iBinder

3. 实现mainActivity.java

1. 申请动态权限读取文件，需要在AndroidManifest.xml中声明，然后在mainActivity中调用申请权限的函数（安卓6.0以上的版本才需要权限申请）

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

```
//申请用户权限
public void verifyStoragePermissions(){
    int permission = ActivityCompat.checkSelfPermission(MainActivity.this,Manifest.permission.READ_EXTERNAL_STORAGE);
    if(permission != PackageManager.PERMISSION_GRANTED){
        ActivityCompat.requestPermissions(MainActivity.this,new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},1);
    }
}
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults){
    if(grantResults.length>0 && grantResults[0]==PackageManager.PERMISSION_GRANTED){ }
    else {
        System.exit(0);
    }
    return;
}

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    verifyStoragePermissions();
```

2. 开启并绑定服务MusicService

```
//绑定服务
conn= new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        mBinder = service;
    }
    @Override
    public void onServiceDisconnected(ComponentName name) { conn = null; }
};
Intent musicIntent =  new Intent(MainActivity.this,MusicService.class);
startService(musicIntent);
bindService(musicIntent,conn, Context.BIND_AUTO_CREATE);
```

3. 为图片写一个旋转的动画

```
//动画旋转
final ObjectAnimator animation = ObjectAnimator.ofFloat(img,"rotation",0,360);
animation.setDuration(30000);//旋转一圈的时间
animation.setRepeatCount(-1);//设定无限循环
animation.setInterpolator(new LinearInterpolator());//中间线性补充帧数
```

动画的属性

动画的对象

4. 为播放/暂停、停止、退出三个按钮添加监听器。在监听器里主要实现的：①改变第一个按钮上显示的字，改变进度条上方显示的播放状态 ②改变动画的状态（原地暂停、开始、回到原点后结束）③向服务发生消息，有需要时从服务那里拿到当前播放的时间点，将UI界面更新 ④退出按钮的监听器中还需要解绑服务，避免内存泄露，然后结束当前的activity.

```
//播放/暂停按钮的点击事件
btn_play.setOnClickListener((v) -> {
        if(btn_play.getText().equals("PLAY")){
            btn_play.setText("PAUSED");
            state.setText("Playing");                    ①
            state.setVisibility(View.VISIBLE);
            if(animation.isPaused())
                animation.resume();
            else                                         ②
                animation.start();
        } else{
            btn_play.setText("PLAY");
            state.setText("Paused");
            animation.pause();
        }

        try{
            Parcel reply = Parcel.obtain();
            mBinder.transact(101,Parcel.obtain(),reply,0);
            begin.setText(time.format(reply.readInt()));
        }catch(RemoteException e){
            e.printStackTrace();
        }
});
//退出按钮的点击事件
btn_quit.setOnClickListener((v) -> {
        try{                                             ③
            mBinder.transact(103,Parcel.obtain(),Parcel.obtain(),0);
        }catch(RemoteException e){
            e.printStackTrace();
        }
        unbindService(conn);
        conn = null;                                     ④
        try{
            MainActivity.this.finish();
            System.exit(0);
        }catch (Exception e){
            e.printStackTrace();
```

```
//停止按钮的点击事件
btn_stop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        btn_play.setText("PLAY");
        state.setText("Stopped");                        ①
        animation.setCurrentPlayTime(0);
        animation.end();                                 ②
        try{
            Parcel reply = Parcel.obtain();              ③
            mBinder.transact(102,Parcel.obtain(),reply,0);
            begin.setText(time.format(reply.readInt()));
        }catch(RemoteException e){
            e.printStackTrace();
        }
```

5. 为进度条设置监听器，重写onProgressChanged函数，一旦进度条的位置被用户改变，及时向服务告知当前的位置，以便MusicService改变音乐的播放时间点

```
//进度条监听事件
seekBar.setEnabled(true);
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        if(fromUser){
            try{
                Parcel data = Parcel.obtain();
                data.writeInt(progress);
                mBinder.transact(105,data,Parcel.obtain(),0);
            }catch(RemoteException e){
                e.printStackTrace();
            }
        }
    }
    //开始拖动进度条
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}
    //结束拖动进度条
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) { }
```

6. 创建子线程，让子线程sleep，每100ms醒一次，每次醒来就向handler发生消息，让其去更新UI

```
//创建子线程，在子线程中处理耗时工作
Thread mThread = run() -> {
        while (true){
            try{
                Thread.sleep(100);
            } catch (InterruptedException e){
                e.printStackTrace();
            }
            mHandler.obtainMessage(123).sendToTarget();
        }
};
mThread.start();
```

7. 在mainActivity自身的mHandler中重写handleMessage函数，如果是来自子线程的消息，则在和服务绑定的情况下，向服务发送消息，获取到当前音乐播放的时间点和进度条应当所处的位置，然后设置UI上表示播放时间的TextView和表示音乐文件总时长的TextView，注意此处要用SimpleDateFormate的实例将获取到的数据格式化。再为进度条setProgress和setMax。

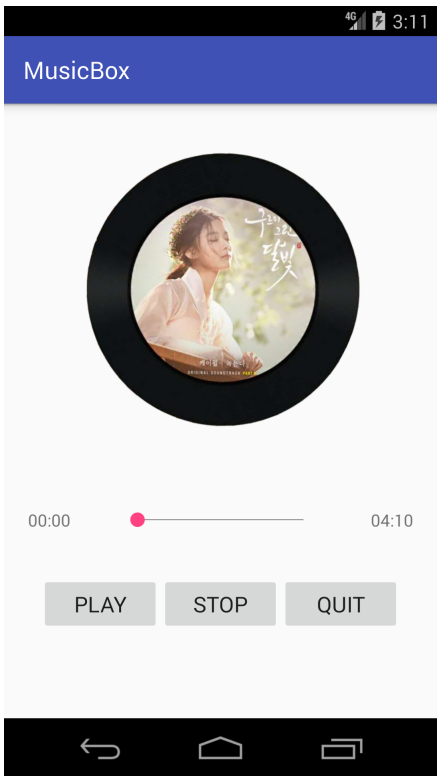
```
mHandler = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (msg.what == 123) {
            if(mBinder != null) {
                try {
                    Parcel reply = Parcel.obtain();
                    mBinder.transact(104, Parcel.obtain(), reply, 0);
                    int location = reply.readInt();
                    begin.setText(time.format(location));
                    seekBar.setProgress(location);
                    int max = reply.readInt();
                    length.setText(time.format(max));
                    seekBar.setMax((max));
                } catch (RemoteException e) {
                    e.printStackTrace();
```

8. 重写mainActivity的onDestroy函数，停止服务MusicService。

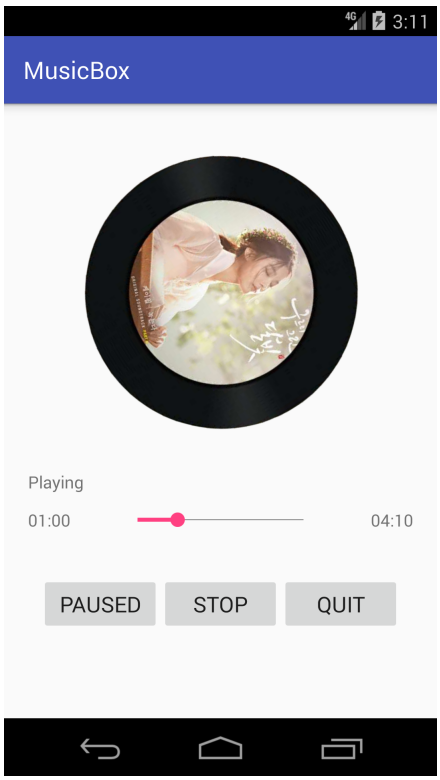
```
@Override
protected void onDestroy(){
    super.onDestroy();
    Intent intent = new Intent(MainActivity.this,MusicService.class);
    stopService(intent);
}
```

5. 实验结果

打开程序时：

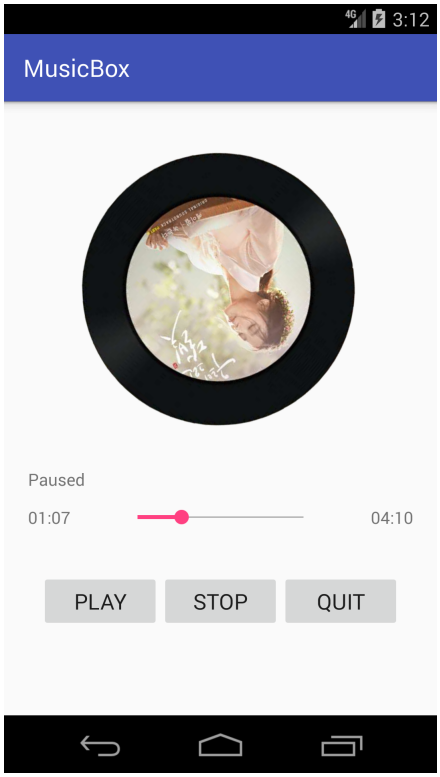


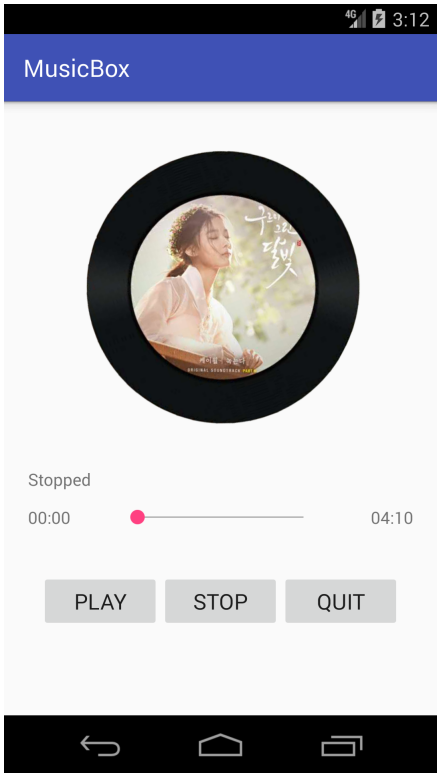
开始播放：



暂停：

停止：





6. 实验思考及感想

这次实验可以说是各种坑，写了不止一个周末。写申请权限的时候，因为把uses-permission看成user-permission，用了一天的时间来填坑 == 其次就是用SImpleDateFormat的时候，报错说需要minSdkVersion至少是24才能使用这个，于是改了app的build.gradle，装了sdk24的虚拟机，然后就因为权限问题还没解决所以无法读取音乐。后来发现，把SimpleDateFormat写成java.text.SimpleDateFormat就可以解决minSdkVersion的问题。这个故事告诉我们，报错的信息一定要仔细看完，有时候有不止一种解决方法。

一开始的时候，是打算一启动activity获取到音乐文件就把它的时长写好，后面不需要再改。但是因为，绑定服务需要一定的时间，这个时候用mBinder还是空的，无法于服务进行通信，因此踩了坑。后来把总时长写进UI的更新里，因为每0.1s更新一次，所以一打开app总时长就已经被写好了，只是这样子每0.1s都会写一次。

此外还学到其他一些知识，例如：①reply用来发送到服务/activity时，对方接收后可直接往里面写东西，不需要再传回来，本地就可以直接拿到被写进的信息。（感觉有点神奇，这个难道没有时间差）。②可以往reply写进两次int，只要拿的时候也按顺序拿就好。③handle发信息的时候，sendMessage和obtainMessage的区别在于，前者是自己新建一个新的message，然后发送，后者是在message资源池里拿到一个，写进信息然后发送，后者更节省资源。