电 子 科 技 大 学

实　验　报　告

学生姓名：Lolipop 学号：2018091202000　指导教师：xxx

实验地点：信软学院楼西 305　　实验时间：2020.11.16

一、实验名称：广播与通知

二、实验学时：4

三、实验目的：掌握移动广播与通知有关的知识

四、实验原理：

1. 广播。
    a) 静态注册就是在AndroidManifest.xml文件中定义，注册的广播接收器必须继承BroadReceiver，动态注册就是在程序中使用Context.registerReceiver注册。
    b) 发送广播事件：通过Context.sendBroadcast来发送，由Intent来传递注册时用到的Action。
    c) 接收广播：当发送的广播被接收器监听到后，会调用onReceive()方法，并将包含消息的Intent对象传回。
2. Notification使用流程。
    a) 获得NotificationManager对象。
    b) 创建一个通知栏的Builder构造类。
    c) 对Builder进行相关的设置，比如标题，内容，图标，动作等。
    d) 调用Builder的build()方法为notification赋值。

e) 调用NotificationManager的notify()方法发送通知

3. Intent是一个消息对象，通过它，Android系统的四大组件能够方便的通信，并且保证解耦。Intent可以说明某种意图，携带一种行为和相应的数据，发送到目标组件。
4. PendingIntent是对Intent的封装，但它不是立刻执行某个行为，而是满足某些条件或触发某些事件后才执行指定的行为。

五、实验内容：

1. 动态注册监听网络变化
2. 静态注册开机启动
3. 发送和接收自定义广播
4. 状态栏显示通知
5. 使用 PendingIntent 打开通知 Activity
6. 收发短信

六、实验器材（设备、元器件）：

个人 PC 机一台

七、实验步骤：

1. 编写自己的 Recivier 类，命名为 MyReceiver，继承 BroadcastReceiver 类，使用 Toast.makeText()方法，显示广播信息，在 AdroidManifest 中加如 receiver 以及给与相应权限。
2. 编写动态 Recivier 类，并赋予相应权限。
3. 编写自定义 Recivier 类，并编写相应按钮触发事件。
4. 编写自己的 Notification，并使用 PendingIntent 方法从通知跳转到

活动页。

5. 使用 sendmessage()方法实现收发短信功能。

八、实验结果与分析（含重要数据结果分析或核心代码流程分析）

打开 Android studio，选择创建项目 experiment_3.1,首先编写自己的 Receiver 类 MyReceiver，编写 onReceiver()方法，使用 makeText 显示需要显示的内容，如代码 1 所示。

代码 1 MyReceiver 类

```
package com.example.myapplication;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "jh 的开机启动",
Toast.LENGTH_SHORT).show();
        Log.i("saa","jh 的开机启动");
    }
}
```

接下来，在 AdroidManifest 文件中，加入<receiver>以及获取相应权限 android:name="android.permission.RECEIVE_BOOT_COMPLETED"，文件代码如代码 2 所示。重启虚拟机，查看静态注册开机启动的结果，结果如图 1

所示。

代码 2 注册开机启动 AdroidManifest 配置

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <receiver
            android:name=".MyReceiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
    </application>

</manifest>
```

图 1 开机启动静态注册

完成静态注册开机启动，接下来开始实现动态注册监听网络变化，新建项目experiment_3.2,首先，编写 NetworkChangeReceiver 类，实现代码部分如代码 3 所示。

代码 3 NetworkChangeReceiver 类

```
package com.example.expdriment_31;

import androidx.appcompat.app.AppCompatActivity;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private IntentFilter intentFilter;

    private NetworkChangeReceiver networkChangeReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        intentFilter = new IntentFilter();

intentFilter.addAction("android.net.conn.CONNECTIVITY_CHANGE");
        networkChangeReceiver = new NetworkChangeReceiver();
```

```
        registerReceiver(networkChangeReceiver,intentFilter);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver(networkChangeReceiver);
    }


    class NetworkChangeReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            ConnectivityManager connectivityManager=(ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo networkInfo
=connectivityManager.getActiveNetworkInfo();
            if (networkInfo != null && networkInfo.isAvailable()) {
                Toast.makeText(context, "已经连接上网络",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(context,"当前没有网络
",Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

然后修改 AdroidManifest 文件，加入<receiver>和赋予相应的权限 android:name="android.permission.ACCESS_NETWORK_STATE"，实现代码如代码 4 所示。动态注册监听网络变化实现结果如图 2 所示。

代码 4 监听网络变化 AdroidManifest 配置

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.expdriment_31">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver
            android:name=".BootCompletedReceiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>

    </application>

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

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

图 2 动态监听网络状态结果

接下来实现发送和接收自定义广播，新建项目 experiment_3.3，首先再 layout 文件 activity_main.xml 中创建按钮“发送广播”，如代码 4 所示。

代码 4 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <Button
        android:id="@+id/mbtn1"
        android:text="点击发送广播"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

然后创建自定义 receiver 类，MyBroadcastReceiver 类，如代码 5 所示。

代码 5 MyBroadcastReceiver 类

```
package com.example.experiment_32;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MyBroadcastReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "收到来自小枫的自定义广播",
Toast.LENGTH_SHORT).show();
```

```
    }
}
```

接下来，在 Adroidmanifest 文件中，加入<receiver>并赋予相应权限，如代码 6 所示。

代码 6 Adroidmanifest 配置自定义广播

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.experiment_32">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <receiver
            android:enabled="true"
            android:exported="true"
            android:name=".MyBroadcastReceiver">
            <intent-filter>
                <action android:name="com.xiaofeng.cn"></action>
            </intent-filter>
        </receiver>
```

```
    </application>
</manifest>
```

最后，在 MainActivity 中加入相应的按钮点击事件，如代码 7 所示。完成自定义发送和接收广播，实现结果如图 3 所示。

代码 7 MainActivity 实现自定义广播

```
package com.example.experiment_32;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
    private Button mbtn1;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mbtn1 = (Button) findViewById(R.id.mbtn1);
        mbtn1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent=new
Intent(MainActivity.this,MyBroadcastReceiver.class);
                sendBroadcast(intent);
            }
        });
    }
}
```

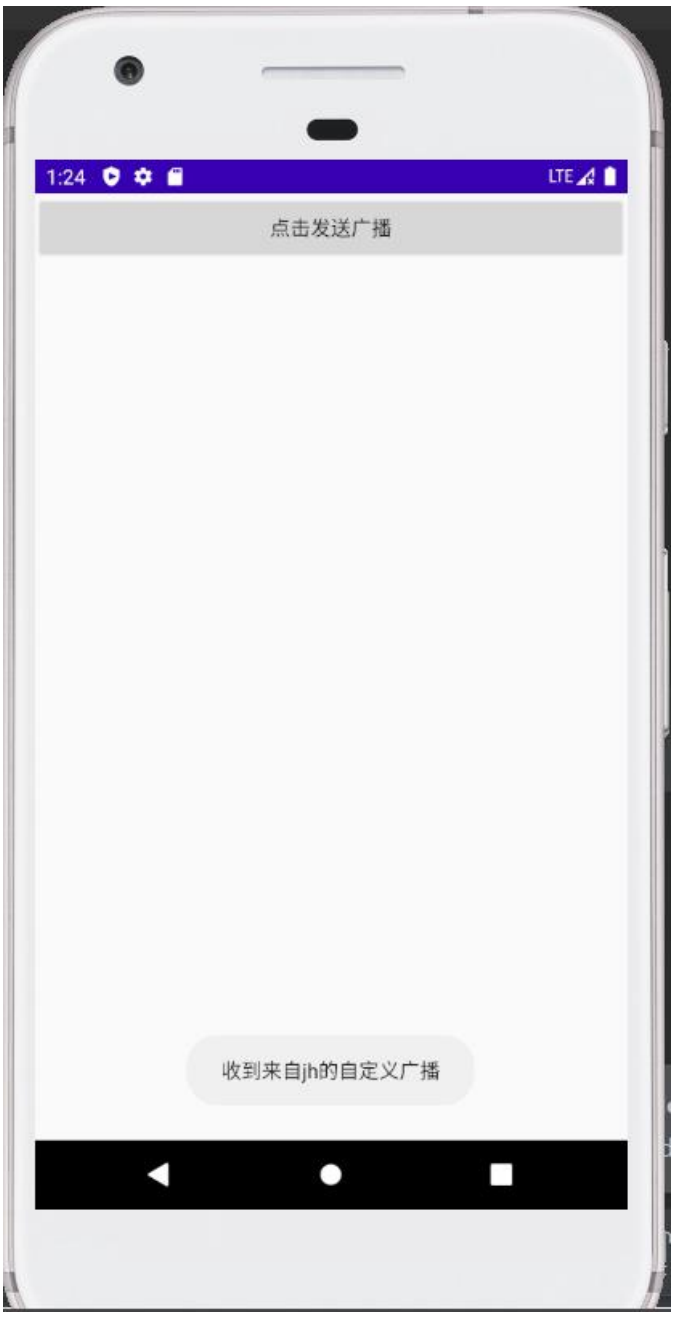

图 3 实现自定义广播

接下来实现状态栏显示通知以及使用 PendingIntent 打开通知 Activity，创建项目 experiment_3.4，首先调用 getSystemService()方法获取系统的 NotificationManager 服务。然后创建一个 Notification 对象，并为其设置各种属性，最后通过 NotificationManager 类的 notify()方法发送 Notification 通知，在中间，设置一个启动 DetailActivity 的 intent，就可以实现用 PendingIntent 打开 DetailActivity 了，各部分代码如代码 8 至代码 10 所示。

代码 8 NotificationActivity

```
package com.example.experiment_3tongzhi;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;

public class NotificationActivity extends AppCompatActivity {
    final int NOTIFYID = 2018091202005;                    //通知的 ID
    @RequiresApi(api = Build.VERSION_CODES.JELLY_BEAN)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //新建通知管理器
        final NotificationManager notificationManager =
                (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
```

```
        // 创建一个 Notification 对象
        Notification.Builder notification = new Notification.Builder(this);
        // 设置打开该通知，该通知自动消失
        notification.setAutoCancel(true);
        // 设置通知的图标
        notification.setSmallIcon(R.drawable.dog);
        // 设置通知内容的标题
        notification.setContentTitle("这是一个通知");
        // 设置通知内容
        notification.setContentText("点击查看详情！");
        //设置使用系统默认的声音、默认震动
        notification.setDefaults(Notification.DEFAULT_SOUND
                | Notification.DEFAULT_VIBRATE);
        //设置发送时间
        notification.setWhen(System.currentTimeMillis());
        // 创建一个启动其他 Activity 的 Intent
        Intent intent = new Intent(NotificationActivity.this
                , DetailActivity.class);
        PendingIntent pi = PendingIntent.getActivity(
                NotificationActivity.this, 0, intent, 0);
        //设置通知栏点击跳转
        notification.setContentIntent(pi);
        //发送通知
        notificationManager.notify(NOTIFYID, notification.build());
    }
}
```

代码 9 DetailActivity

```
package com.example.experiment_3tongzhi;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
```

```
public class DetailActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_detail);
    }
}
```

代码 10 activity_detail

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".DetailActivity">

    <TextView
        android:layout_width="wrap_content"
        android:text="这就是一个通知而已"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

同时要在 AdroidManifest 文件中加入对应权限，部分内容如代码 11 所示。

代码 11 配置 AdroidManifest 状态栏通知

```
<!-- 添加操作闪光灯的权限 -->
<uses-permission android:name="android.permission.FLASHLIGHT"/>
```

```
<!-- 添加操作震动器的权限 -->
<uses-permission android:name="android.permission.VIBRATE"/>
```

最后，需要实现发送短信功能，新建项目 experiment_3.5，首先先设计布局文件设计，需要收件人、信息、以及发送按钮,布局文件 activity_main.xml 如代码 12 所示。

代码 12 activity_main 的布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/textview01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15dp"
            android:text="收件人" />
        <EditText
            android:id="@+id/edittext01"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:orientation="horizontal">
```

```
        <TextView
            android:id="@+id/textview02"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15dp"
            android:text="内容：" />
        <EditText
            android:id="@+id/edittext02"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp" />
    </LinearLayout>
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="30dp"
        android:text="发送信息" />
</LinearLayout>
```

然后将 textview 中的数据使用 tostring（）的方法，带入 massage.sendTextMessage（）函数中，就可以完成短信的发送了，代码如代码 13 所示。执行结果如图 4 所示。

代码 13 MainActivity 实现短信发送

```
package com.example.experiment_duanxin;

import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.view.View.OnClickListener;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {


    private Button button;
    private EditText edittext01, edittext02;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new buttonListener());
    }
    class buttonListener implements OnClickListener {
        @Override
        public void onClick(View v) {
            edittext01 = (EditText) findViewById(R.id.edittext01);
            edittext02 = (EditText) findViewById(R.id.edittext02);
            String number = edittext01.getText().toString();
            //获取手机号码
            String messageOl = edittext02.getText().toString();
            //获取短信内容
            if (number.equals("") || messageOl.equals(""))
            //判断输入是否有空格
            {
                Toast.makeText(MainActivity.this, "输入有误，请检查输入",
                        Toast.LENGTH_LONG).show();
            } else {
                SmsManager massage = SmsManager.getDefault();
                massage.sendTextMessage(number, null, messageOl, null,
null);
                //调用 senfTextMassage 方法来发送短信
```

```
                Toast.makeText(MainActivity.this, "短信发送成功",
                        Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

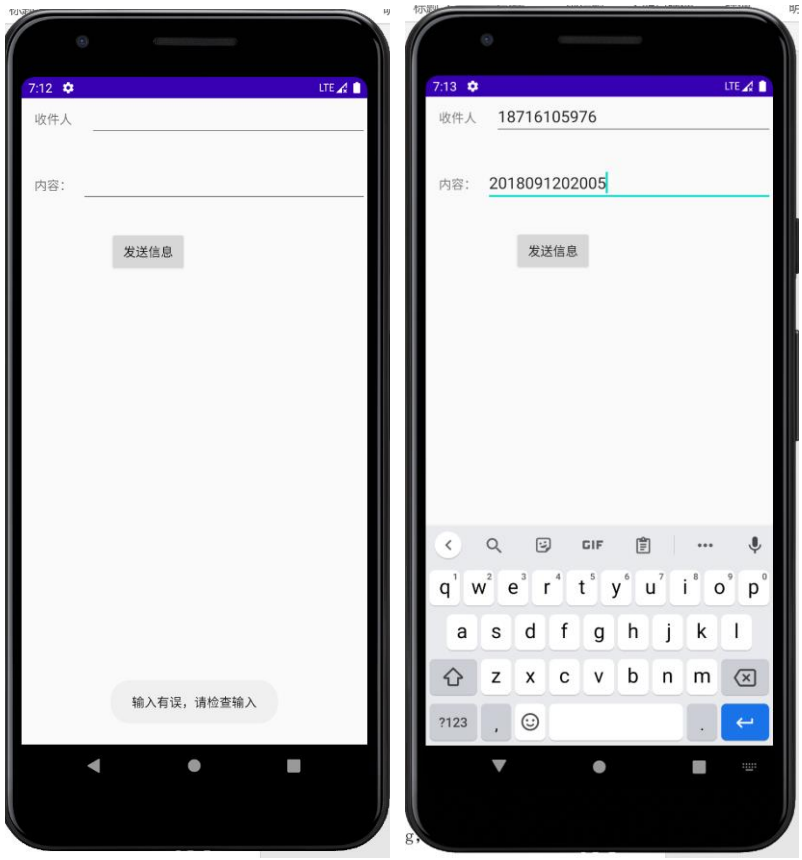

图 4 发送短信执行结果

九、总结及心得体会：

实验三主要是关于发送接收广播、通知和短信的练习。本次实验和一般的 Android 应用开发不同，涉及到的不是应用本身的界面设计和功能完善，而是与用户和其它应用进行联系交互。开发广播与通知可以完善应用的生态，增加应用的拓展性，使得用户能够更好地利用应用来满足自己的生产力需求。

十、对本实验过程及方法、手段的改进建议：

本实验并没有和之前所开发的基于 Flutter 的实验室代码部署系统所连接起来。作为改进，可以将广播与通知的功能加入到实验室代码部署系统中去，例如当实验室代码执行完成时，向系统发送一条通知展示执行结果，用户也可以点击通知来到该界面等。

报告评分：

指导教师签字：