Android 软件开发

Broadcasts

秦兴国

xgqin@guet.edu.cn

计算机与信息安全学院
桂林电子科技大学

2018 年 11 月 14 日

Outline

Broadcast 简介

Broadcast (广播) 是 Android 系统与 App、App 与 App 之间进行消息通信的一种机制。Android 系统通过发送广播的形式 (System broadcasts, 系统广播) 通知 App，某些事件正在发生。例如：

- 系统启动完毕；
- 设备充电及断电；
- 网络发生切换等；

[1] 当 App 注册的广播后，该广播对应的事件发生时，Android 系统会通知注册该广播的 App，并调用广播接收器 (Broadcast receiver) 中相应的回调方法

Broadcast 简介

Broadcast (广播) 是 Android 系统与 App、App 与 App 之间进行消息通信的一种机制。Android 系统通过发送广播的形式 (System broadcasts, 系统广播) 通知 App，某些事件正在发生。例如：

- 系统启动完毕；
- 设备充电及断电；
- 网络发生切换等；

App 自身也可通过发送广播的形式 (Custom broadcasts, 自定义广播) 通知其他 App 某些事件正在发生 (例如：文件下载成功，数据同步成功等)。

[1] 当 App 注册的广播后，该广播对应的事件发生时，Android 系统会通知注册该广播的 App，并调用广播接收器 (Broadcast receiver) 中相应的回调方法

Broadcast 简介

Broadcast (广播) 是 Android 系统与 App、App 与 App 之间进行消息通信的一种机制。Android 系统通过发送广播的形式 (System broadcasts, 系统广播) 通知 App，某些事件正在发生。例如：

- 系统启动完毕；
- 设备充电及断电；
- 网络发生切换等；

App 自身也可通过发送广播的形式 (Custom broadcasts, 自定义广播) 通知其他 App 某些事件正在发生 (例如：文件下载成功，数据同步成功等)。

App 可通过注册广播的方式对其感兴趣的事件进行响应[1]。

[1] 当 App 注册的广播后，该广播对应的事件发生时，Android 系统会通知注册该广播的 App，并调用广播接收器 (Broadcast receiver) 中相应的回调方法

系统广播

系统广播 (System broadcast) 由 Android 系统进行发送，通常表示系统级事件发生。

[2] 可通过查询 SDK 目录下的broadcast_actions.txt文件查看该版本下支持的系统广播完整列表

系统广播

系统广播 (System broadcast) 由 Android 系统进行发送，通常表示系统级事件发生。

系统广播由 **Intent** 意图对象进行封装。该 **Intent** 对象中的 *action* 字段包含了表示事件的详细信息字符串签名。

[2] 可通过查询 SDK 目录下的broadcast_actions.txt文件查看该版本下支持的系统广播完整列表

系统广播

系统广播 (System broadcast) 由 Android 系统进行发送，通常表示系统级事件发生。

系统广播由 **Intent** 意图对象进行封装。该 **Intent** 对象中的 *action* 字段包含了表示事件的详细信息字符串签名。

例如：**android.intent.action.HEADSET_PLUS** 表示有线耳塞接入或断开。**intent** 对象还可在 *extra* 字段中包含事件的额外信息。在此例中，**intent** 对象 *extra* 字段包含一个 *boolean* 类型数据表示耳机接入状态。

[2] 可通过查询 SDK 目录下的broadcast_actions.txt文件查看该版本下支持的系统广播完整列表

系统广播

系统广播 (System broadcast) 由 Android 系统进行发送，通常表示系统级事件发生。

系统广播由 **Intent** 意图对象进行封装。该 **Intent** 对象中的 *action* 字段包含了表示事件的详细信息字符串签名。

例如：**android.intent.action.HEADSET_PLUS** 表示有线耳塞接入或断开。**intent** 对象还可在 *extra* 字段中包含事件的额外信息。在此例中，**intent** 对象 *extra* 字段包含一个 *boolean* 类型数据表示耳机接入状态。

如果 App 对特定的系统广播感兴趣，需要通过广播接收器 (Broadcast Receiver) 对系统广播进行注册[2]。

[2] 可通过查询 SDK 目录下的broadcast_actions.txt文件查看该版本下支持的系统广播完整列表

自定义广播 I

自定义广播 (Custom Broadcasts) 由 App 自身进行发送。当你希望你的 App 在不启动活动 (Activity) 的情况下执行某一个行为或动作，自定义广播是一个不错的选择。

[3] 需要注意的是注册同一广播的广播接收器数量不受限制

[4] 自定义广播的 *action* 字段时，使用 App 的包名 (例如:) 作为 *action* 的前缀可以有效的避免你的自定义广播 *action* 与其他 App 或系统广播相冲突。

自定义广播 I

自定义广播 (Custom Broadcasts) 由 App 自身进行发送。当你希望你的 App 在不启动活动 (Activity) 的情况下执行某一个行为或动作，自定义广播是一个不错的选择。

App 发送自定义广播，通常是为了通知其他 App 或本 App 中的其他组件某一事件已经发生。对这类事件感兴趣的 App 或组件可以及注册该自定义广播的广播接收器[3]。

[3] 需要注意的是注册同一广播的广播接收器数量不受限制

[4] 自定义广播的 *action* 字段时，使用 App 的包名 (例如:) 作为 *action* 的前缀可以有效的避免你的自定义广播 *action* 与其他 App 或系统广播相冲突。

自定义广播 I

自定义广播 (Custom Broadcasts) 由 App 自身进行发送。当你希望你的 App 在不启动活动 (Activity) 的情况下执行某一个行为或动作，自定义广播是一个不错的选择。

App 发送自定义广播，通常是为了通知其他 App 或本 App 中的其他组件某一事件已经发生。对这类事件感兴趣的 App 或组件可以及注册该自定义广播的广播接收器[3]。

创建一个自定义的广播，只需要自定义一个 **Intent** 对象的 *action* 字段[4]。

[3] 需要注意的是注册同一广播的广播接收器数量不受限制

[4] 自定义广播的 *action* 字段时，使用 App 的包名 (例如:) 作为 *action* 的前缀可以有效的避免你的自定义广播 *action* 与其他 App 或系统广播相冲突。

自定义广播 - 标准广播 (Normal Broadcast)

sendBroadcast() 方法将广播以无序的方式 (undefined order) 同时发送到注册该广播的接收器中。

自定义广播 - 标准广播 (Normal Broadcast)

sendBroadcast() 方法将广播以无序的方式 (undefined order) 同时发送到注册该广播的接收器中。

这类方式称为标准广播，标准广播是发送广播的最有效 (most efficient) 的方式。在标准广播中广播接收器无法取消该广播事件。

自定义广播 - 标准广播 (Normal Broadcast)

sendBroadcast() 方法将广播以无序的方式 (undefined order) 同时发送到注册该广播的接收器中。

这类方式称为标准广播，标准广播是发送广播的最有效 (most efficient) 的方式。在标准广播中广播接收器无法取消该广播事件。

```
1       ...
2       Intent intent = new Intent();
3       intent.setAction("com.example.myproject.ACTION_SHOW_TOAST");
4       // Set the optional additional information in extra field.
5       intent.putExtra("data", "This is a normal broadcast");
6       sendBroadcast(intent);
7       ...
```

自定义广播 - 有序广播 (Ordered Broadcast)

sendOrderedBroadcast()将广播依次发送给注册了该广播的广播接收器。

[5] 如果在同一个广播中存在多个相同优先级的广播接收器，则在这类广播接收器中传递广播的顺序是随机的。

[6] 如果广播接收器取消该广播，则广播的 **Intent** 对象将不再传播给下一个广播接收器。

自定义广播 - 有序广播 (Ordered Broadcast)

sendOrderedBroadcast()将广播依次发送给注册了该广播的广播接收器。

在广播接收器的 **intent filter** 中通过android : priority 属性指明该广播接收器的优先级顺序[5]。

[5] 如果在同一个广播中存在多个相同优先级的广播接收器，则在这类广播接收器中传递广播的顺序是随机的。

[6] 如果广播接收器取消该广播，则广播的 **Intent** 对象将不再传播给下一个广播接收器。

自定义广播 - 有序广播 (Ordered Broadcast)

sendOrderedBroadcast()将广播依次发送给注册了该广播的广播接收器。

在广播接收器的 **intent filter** 中通过android : priority 属性指明该广播接收器的优先级顺序[5]。

广播接收器在处理完成该广播后可以选择将该广播的 **Intent** 对象传播给下一个广播接收器或取消该广播[6]。

[5] 如果在同一个广播中存在多个相同优先级的广播接收器，则在这类广播接收器中传递广播的顺序是随机的。

[6] 如果广播接收器取消该广播，则广播的 **Intent** 对象将不再传播给下一个广播接收器。

自定义广播 - 有序广播 (Ordered Broadcast)

sendOrderedBroadcast()将广播依次发送给注册了该广播的广播接收器。

在广播接收器的 **intent filter** 中通过android: priority 属性指明该广播接收器的优先级顺序[5]。

广播接收器在处理完成该广播后可以选择将该广播的 **Intent** 对象传播给下一个广播接收器或取消该广播[6]。

```
1
2     Intent intent = new Intent();
3
4     // Set a unique action string prefixed by your app package name.
5     intent.setAction("com.example.myproject.ACTION_NOTIFY");
6
7     // Deliver the Intent.
8     sendOrderedBroadcast(intent);
```

[5] 如果在同一个广播中存在多个相同优先级的广播接收器，则在这类广播接收器中传递广播的顺序是随机的。

[6] 如果广播接收器取消该广播，则广播的 **Intent** 对象将不再传播给下一个广播接收器。

自定义广播 - 本地广播 (Local Broadcast)

本地广播使用LocalBroadcastManager.sendBroadcast() 方法进行发送。本地广播仅对本 App 内组件可见。

自定义广播 - 本地广播 (Local Broadcast)

本地广播使用LocalBroadcastManager.sendBroadcast() 方法进行发送。本地广播仅对本 App 内组件可见。

本地广播不涉及进程间通信 (IPC)，因此其效率较高。使用本地广播可有效保护 App 内的广播避免被其他恶意 App 使用。

自定义广播 - 本地广播 (Local Broadcast)

本地广播使用LocalBroadcastManager.sendBroadcast() 方法进行发送。本地广播仅对本 App 内组件可见。

本地广播不涉及进程间通信 (IPC)，因此其效率较高。使用本地广播可有效保护 App 内的广播避免被其他恶意 App 使用。

发送本地广播的方法：

- 通过LocalBroadcastManager类的getInstance()静态方法获取LocalBroadcastManager实例。
- 调用实例的sendBroadcast()方法，发送本地广播；

```
1        ...
2        LocalBroadcastManager lbm = LocalBroadcastManager.getInstance(this);
3        lbm.sendBroadcast(customBroadcastIntent);
4        ...
```

Broadcast Receiver 简介

Broadcast Receiver (广播接收器) 是 Android 应用开发四大组件之一，其用于注册系统或 App 的广播事件。

Broadcast Receiver 简介

Broadcast Receiver (广播接收器) 是 Android 应用开发四大组件之一，其用于注册系统或 App 的广播事件。

当事件发生时，注册了该事件的广播接收器被系统通过 **Intent** 对象激活。

Broadcast Receiver 简介

Broadcast Receiver (广播接收器) 是 Android 应用开发四大组件之一，其用于注册系统或 App 的广播事件。

当事件发生时，注册了该事件的广播接收器被系统通过 **Intent** 对象激活。

使用广播接收器的步骤包括：

1. 子类化 BroadcastReceiver并实现onReceive()方法；

Broadcast Receiver 简介

Broadcast Receiver (广播接收器) 是 Android 应用开发四大组件之一，其用于注册系统或 App 的广播事件。

当事件发生时，注册了该事件的广播接收器被系统通过 **Intent** 对象激活。

使用广播接收器的步骤包括：

1. 子类化 BroadcastReceiver并实现onReceive()方法；
2. 通过静态或动态方式注册广播接收器；

子类化 Broadcast Receiver I

创建广播接收器的第一步是定义BroadcastReceiver的子类。

子类化 Broadcast Receiver I

创建广播接收器的第一步是定义BroadcastReceiver的子类。

当广播事件的 **Intent** 对象与该子类注册时的 **intent filter** 匹配时，该广播接收器子类将接收 **Intent** 对象，并通过onReceive()方法响应广播事件。

子类化 Broadcast Receiver I

创建广播接收器的第一步是定义BroadcastReceiver的子类。

当广播事件的 **Intent** 对象与该子类注册时的 **intent filter** 匹配时，该广播接收器子类将接收 **Intent** 对象，并通过onReceive()方法响应广播事件。

```
1     //Subclass of the BroadcastReceiver class.
2     private class myReceiver extends BroadcastReceiver {
3         // Override the onReceive method to receive the broadcasts
4         @Override
5         public void onReceive(Context context, Intent intent) {
6             //Check the Intent action and perform the required operation
7             if (intent.getAction().equals(ACTION_SHOW_TOAST)) {
8                 CharSequence text = "Broadcast Received!";
9                 int duration = Toast.LENGTH_SHORT;
10
11                Toast toast = Toast.makeText(context, text, duration);
12                toast.show();
13            }
14        }
15    }
```

子类化 BroadcastReceiver II

当 App 收到一个已注册的广播 **Intent** 对象时，BroadcastReceiver类中的onReceive()方法被回调。

[7] 可通过NotificationManager显示通知

[8] 使用 WorkManager 进行任务调度时，任务将会确保被执行。WorkManager 基于设备的 API 版本及 App 状态选择合适的方式执行任务。

子类化 BroadcastReceiver II

当 App 收到一个已注册的广播 **Intent** 对象时，BroadcastReceiver类中的onReceive()方法被回调。

onReceive()方法默认主线程中执行，除非在注册时显式的指定该方法在其他线程执行。

[7] 可通过NotificationManager显示通知

[8] 使用 WorkManager 进行任务调度时，任务将会确保被执行。WorkManager 基于设备的 API 版本及 App 状态选择合适的方式执行任务。

子类化 BroadcastReceiver II

当 App 收到一个已注册的广播 **Intent** 对象时，BroadcastReceiver类中的onReceive()方法被回调。

onReceive()方法默认主线程中执行，除非在注册时显式的指定该方法在其他线程执行。

onReceive()方法中的代码执行时间若超过 10 秒，Android 系统将判定广播接收器被阻塞，此时系统将可能向用户提示 ANR (application not responding) 错误。

---

[7] 可通过NotificationManager显示通知

[8] 使用 WorkManager 进行任务调度时，任务将会确保被执行。WorkManager 基于设备的 API 版本及 App 状态选择合适的方式执行任务。

子类化 BroadcastReceiver II

当 App 收到一个已注册的广播 **Intent** 对象时，BroadcastReceiver类中的onReceive()方法被回调。

onReceive()方法默认主线程中执行，除非在注册时显式的指定该方法在其他线程执行。

onReceive()方法中的代码执行时间若超过 10 秒，Android 系统将判定广播接收器被阻塞，此时系统将可能向用户提示 ANR (application not responding) 错误。

特别注意：

- 不要在广播接收器中显示对话框[7]。
- 通过Context. startService ()启动服务。
- 使用WorkManager调度耗时的任务[8]。

[7] 可通过NotificationManager显示通知

[8] 使用 WorkManager 进行任务调度时，任务将会确保被执行。WorkManager 基于设备的 API 版本及 App 状态选择合适的方式执行任务。

注册广播接收器

注册广播接收器的两种方式：

- 静态注册，通过 Android 配置清单文件注册广播接收器；
- 动态注册，使用 Context 上下文类注册广播接收器；

注册广播接收器 - 静态注册

通过在AndroidManifest.xml中声明 <**receiver**> 标签的方式进行广播接收器的静态注册[9] [10]。

在 <**receiver**> 标签中需要指定如下属性值:

- **android:name** ，广播接收器子类的完整类名，如果该子类与配置清单文件指定的报名相同则可使用简写形式，如.AlarmReceiver;

注册广播接收器 - 静态注册

通过在AndroidManifest.xml中声明 <**receiver**> 标签的方式进行广播接收器的静态注册[9] [10]。

在 <**receiver**> 标签中需要指定如下属性值:

- **android:name** ，广播接收器子类的完整类名，如果该子类与配置清单文件指定的报名相同则可使用简写形式，如.AlarmReceiver;
- **android:exported**，该值被设为 *false* 时，其他 App 不可向该广播接收器发送广播;

注册广播接收器 - 静态注册

通过在AndroidManifest.xml中声明 <**receiver**> 标签的方式进行广播接收器的静态注册[9] [10]。

在 <**receiver**> 标签中需要指定如下属性值:

- **android:name** ，广播接收器子类的完整类名，如果该子类与配置清单文件指定的报名相同则可使用简写形式，如.AlarmReceiver;
- **android:exported**，该值被设为 *false* 时，其他 App 不可向该广播接收器发送广播;
- <**intent−filter**>，包含嵌套的标签用于指定该广播接收器监听的广播 **Intent** 对象的动作 (actions)[11]。

注册广播接收器 - 静态注册

通过在AndroidManifest.xml中声明 <**receiver**> 标签的方式进行广播接收器的静态注册[9] [10]。

```
1 <receiver android:name=".AlarmReceiver" android:exported="false">
2     <intent-filter>
3         <action android:name="com.example.myproject.intent.action.ACTION_SHOW_TOAST"/>
4     </intent-filter>
5 </receiver>
```

[9] 对于 Android 8.0 (API level 26) 以上版本，静态注册形式注册的隐式广播 (implicit broadcasts) 的行为是无效的。

[10] 部分隐式广播 (ACTION_BOOT_COMPLETED、ACTION_TIMEZONE_CHANGED) 仍可通过静态注册形式进行注册。

注册广播接收器 - 动态注册 I

动态注册也称为上下文注册 (context-registered)。通过应用上下文 (Application Context) 或活动上下文 (Activity Context) 对广播进行动态注册。

注册广播接收器 - 动态注册 I

动态注册也称为上下文注册 (context-registered)。通过应用上下文 (Application Context) 或活动上下文 (Activity Context) 对广播进行动态注册。

动态注册的广播接收器在注册的上下文存续期可接收到其注册的广播事件：

- 如果使用应用上下文注册广播，则在应用运行时 (前台运行及后台运行)，广播接收器均能接收到注册的广播事件；

注册广播接收器 - 动态注册 I

动态注册也称为上下文注册 (context-registered)。通过应用上下文 (Application Context) 或活动上下文 (Activity Context) 对广播进行动态注册。

动态注册的广播接收器在注册的上下文存续期可接收到其注册的广播事件：

- 如果使用应用上下文注册广播，则在应用运行时 (前台运行及后台运行)，广播接收器均能接收到注册的广播事件；
- 如果使用活动上下文注册广播，在活动被销毁前 (onDestroy 回调方法执行前)，广播接收器均能接收到注册的广播事件；

注册广播接收器 - 动态注册 II

动态注册广播接收器的步骤：

1. 创建 IntentFilter 对象，并添加需要注册的广播Intent动作 (可以添加多个Intent动作)。

```
1 IntentFilter intentFilter = new IntentFilter();
2 filter.addAction(Intent.ACTION_POWER_CONNECTED);
3 filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
```

注册广播接收器 - 动态注册 II

动态注册广播接收器的步骤：

1. 创建 IntentFilter 对象，并添加需要注册的广播Intent动作 (可以添加多个Intent动作)。

```
1 IntentFilter intentFilter = new IntentFilter();
2 filter.addAction(Intent.ACTION_POWER_CONNECTED);
3 filter.addAction(Intent.ACTION_POWER_DISCONNECTED);
```

2. 通过上下文对象的 registerReceiver ()方法注册所需的广播接收器。

```
1  mReceiver = new AlarmReceiver();
2  this.registerReceiver(mReceiver, intentFilter);
```

注册广播接收器 - 动态注册 III

本地广播需要通过动态注册方式进行注册。注册本地广播的步骤：

1. 通过LocalBroadcastManager类的静态方法 *getInstance ()* 获取实例对象；
2. 调用 registerReceiver ()方法进行本地广播注册；

注册广播接收器 - 动态注册 III

本地广播需要通过动态注册方式进行注册。注册本地广播的步骤：

1. 通过LocalBroadcastManager类的静态方法 *getInstance()* 获取实例对象；
2. 调用 registerReceiver ()方法进行本地广播注册；

```
1  LocalBroadcastManager.getInstance(this)
2      .registerReceiver(mReceiver,
3              new IntentFilter(CustomReceiver.ACTION_CUSTOM_BROADCAST));
```

注销广播接收器

动态注册的广播接收器在不需要时需进行注销操作。注销广播接收器可节省系统资源及避免内存泄露。

[11] 你也可以在onStart()/onStop()或onCreate()/onDestroy()中进行广播接收器的注册及注销操作。

注销广播接收器

动态注册的广播接收器在不需要时需进行注销操作。注销广播接收器可节省系统资源及避免内存泄露。

注销标准广播接收器，调用 unregisterReceiver ()方法：

```
1    unregisterReceiver(mReceiver);
```

[11] 你也可以在onStart()/onStop()或onCreate()/onDestroy()中进行广播接收器的注册及注销操作。

注销广播接收器

动态注册的广播接收器在不需要时需进行注销操作。注销广播接收器可节省系统资源及避免内存泄露。

注销标准广播接收器，调用 unregisterReceiver ()方法：

```
1    unregisterReceiver(mReceiver);
```

注销本地广播接收器的方法：

```
1    LocalBroadcastManager.getInstance(this).unregisterReceiver(mReceiver);
```

[11] 你也可以在onStart()/onStop()或onCreate()/onDestroy()中进行广播接收器的注册及注销操作。

注销广播接收器

动态注册的广播接收器在不需要时需进行注销操作。注销广播接收器可节省系统资源及避免内存泄露。

注销标准广播接收器，调用 unregisterReceiver ()方法：

```
1    unregisterReceiver(mReceiver);
```

注销本地广播接收器的方法：

```
1    LocalBroadcastManager.getInstance(this).unregisterReceiver(mReceiver);
```

在 **Activity** 中，根据 **Activity** 的生命周期对广播接收器进行注册及注销操作。例如仅当 **Activity** 可见时才需要广播接收器的话，可以在onResume()中进行注册，在onPause()中进行注销[11]；

[11] 你也可以在onStart()/onStop()或onCreate()/onDestroy()中进行广播接收器的注册及注销操作。

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件的方式：

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件的方式：

- 如不需与其他 App 进行通信，在本地 App 内使用本地广播；

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件的方式：

- 如不需与其他 App 进行通信，在本地 App 内使用本地广播；
- 发送广播 **Intent** 对象时，通过setPackage()设置指定的应用包名，广播事件将被限定传播给指定的 App；

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件的方式：

- 如不需与其他 App 进行通信，在本地 App 内使用本地广播；
- 发送广播 **Intent** 对象时，通过setPackage()设置指定的应用包名，广播事件将被限定传播给指定的 App；
- 强制 (enforce) 广播事件发送端或接收端的访问权限：

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件的方式：

- 如不需与其他 App 进行通信，在本地 App 内使用本地广播；
- 发送广播 **Intent** 对象时，通过setPackage()设置指定的应用包名，广播事件将被限定传播给指定的 App；
- 强制 (enforce) 广播事件发送端或接收端的访问权限：
  - 强制发送端的权限，在sendBroadcast()方法中添加一个非空的权限参数。只有在AndroidManifest.xml文件中使用了<uses−permission>标签申请该权限的广播接收器才能接收到该广播；

限制广播事件

非受限的广播 (unrestricted broadcast) 可能会引起安全的问题，因为任何广播接收器均可对其进行注册。

限制广播事件的方式：

- 如不需与其他 App 进行通信，在本地 App 内使用本地广播；
- 发送广播 **Intent** 对象时，通过setPackage()设置指定的应用包名，广播事件将被限定传播给指定的 App；
- 强制 (enforce) 广播事件发送端或接收端的访问权限：
  - 强制发送端的权限，在sendBroadcast()方法中添加一个非空的权限参数。只有在AndroidManifest.xml文件中使用了<uses−permission>标签申请该权限的广播接收器才能接收到该广播；
  - 强制接收端的权限，动态注册的广播接收器，在 registerReceiver ()添加一个非空的权限参数；静态注册的广播接收器，在AndroidManifest.xml文件的<receiver>标签中增加android:permission属性；