前两天在论坛上看到有人发了一个帖子，询问一个Android GUI Event处理的问题：有一个LinearLayout，里面有很多的child view，他问如何监听这个LinearLayout的Click事件？他的做法是：

setClickable(true);

setOnClickListener(listener);

最后他发现listener中的回调函数根本不会被调用。

事实上，在Android的GUI系统的中，硬件触发的Event（KeyEvent、 TouchEvent、 TrackballEvent等）最开始是Window拿到了，Window将Event转发给了前台的Activity。但是Activity同样不能 马上自己处理掉，而是将Event传递给了它里面的ContentView。

如果ContentView是一个容器View（继承自ViewGroup类型），它一般都是先判断这个Event落在哪一个Child View上。然后将该Event Dispatch给这个Child View了。

我们以Touch Event为例，看看ViewGroup类中的dispatchTouchEvent()函数。  
        我们可以看到ViewGroup通过调用Child View的dispatchTouchEvent()函数，将Event按照View树状结构一级一级的Dispatch下去（Child View， GrandChild View ......）。

        最终，最里层的Child View拿到了这个Event，而它又没有Child View了。于是它就开始处理Event（也就是响应事件）。

        我们还以Touch Event为例，看看View类中的dispatchTouchEvent()函数。

         首先，它是判断这个Touch Event是否安全，如果不安全，返回。

        然后，判断是否设置了Touch Event监听器，如果设置了，就调用mOnTouchListener的处理函数，返回。

        如果没有Touch Event监听器，就调用自身定义onTouchEvent()方法，返回。

        在这种事件处理函数中，都要返回一个boolean类型的值。如果返回了false，它的父容器还能再次拿到了事件的。如果返回了true，也就相当于告诉它的父容器：这事我管了，你就别过问了。

        因此，Activity拥有的处理Event权限是最低级别的。

        我们同时也会注意到，除了硬件触发的Event（KeyEvent、 TouchEvent、 TrackballEvent等）外，还存在一些Event，如：Click、LongPress、DoubleClick等。这些Event通常是人为 的将"硬件触发的Event"封装得来的。例如Click Event就是使用了TouchEvent 中的MotionEvent.ACTION\_UP和MotionEvent.ACTION\_DOWN封装而来的。默认情况下，它们使得 onTouchEvent处理函数返回了true。因此，Linearlayout的孩子们一旦设置了自己的Click事件监听 器，Linearlayout本身就拿不到事件了，因为它的孩子已经进行了处理。这正是论坛中问题的答案所在！

        不过，这里也存在一种解决方案。我们在看ViewGroup的dispatchTouchEvent()函数时，应该注意到ViewGroup在将 Event Dispatch给Child View之前，先调用了自己的onInterceptTouchEvent（）函数（Intercept就是拦截的意思）。因此，我们可以通过重写 Linearlayout的onInterceptTouchEvent（）方法，来拦截我们想要处理的Event，它会在事件传给孩子之前被调用的。