Android 整理

2011---06---24 开始

1、建立 GPRS 连接

```
//Dial the GPRS link.
    private boolean openDataConnection() {
        // Set up data connection.
        DataConnection conn = DataConnection.getInstance();
            if (connectMode == 0) {
               return    ret = conn.openConnection(mContext, “cmwap”, “cmwap”, “cmwap”);
            } else {
               return    ret = conn.openConnection(mContext, “cmnet”, “”, “”);
            }
    }
```

2、判断网络状态是否可用

```
/**

*  判断网络状态是否可用

* @return true:网络可用; false:网络不可用

*/

public boolean isConnectInternet() {

ConnectivityManager
conManager=(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo networkInfo = conManager.getActiveNetworkInfo();

if(networkInfo != null){ //  注意，这个判断一定要的哦，要不然会出错

return networkInfo.isAvailable();

}

return false;
```

}

最后别忘要在加上

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

这个权限

3、获得惯性滑动的位置

（1）检测 touch 状态

```
public class ScrollOnTouchListener implements OnTouchListener {

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            int action = event.getAction();
            switch (action) {
            case MotionEvent.ACTION_DOWN:
                //Toast.makeText(MainActivity.this,
"getHeight="+horizontalScrollView.getScrollX() , 200).show();

            case MotionEvent.ACTION_MOVE:

                //Toast.makeText(MainActivity.this,
"getHeight="+horizontalScrollView.getScrollX() , 200).show();

                break;
            case MotionEvent.ACTION_UP:
                //Toast.makeText(MainActivity.this,
"getHeight="+horizontalScrollView.getScrollX() , 200).show();
                scrollX = horizontalScrollView.getScrollX();
                changeTextSwicher(scrollX);
                detectScrollX();
                break;
            }
            return false;
        }

    }
```

（2）手离开屏幕时的状态

```
public void detectScrollX(){
```

```
        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                int tempScrollX = horizontalScrollView.getScrollX();
                if(tempScrollX != scrollX) {
                    scrollX = tempScrollX;
                    changeTextSwicher(tempScrollX);
                }else {
                    return;
                }
            }
        }, DETECT_TIME);
    }
```

4、横竖屏切换对话框消失

在某些场合可能需要禁止横屏和竖屏切换，实现这个要求很简单，只要在 AndroidManifest.xml 里面加入这一行 android :screenOrientation="landscape "(landscape 是横向，portrait 是纵向)。不过 android 中每次屏幕的切换动会重启 Activity，所以应该在 Activity 销毁前保存当前活动的状态，在 Activity 再次 Create 的时候载入配置。在 activity 加上 android:configChanges="orientation"属性,就不会重启 activity.
改变方向调用的是 onConfigurationChanged(Configuration newConfig)方法。 重写如下：

```
        @Override
        public void onConfigurationChanged(Configuration newConfig) {
            super.onConfigurationChanged(newConfig);
            if (this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
                // land

            } else if (this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_PORTRAIT) {
                // port
            }
            super.onConfigurationChanged(newConfig);
        }
```

如果想禁止横屏，很简单：在 onCreate()方法里面加上 setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_NOSENSOR);
或者直接在清单列表文件（manifest.xml）文件中，需要禁止转向的 activity 配置中加入 android:screenOrientation=”landscape”属性即可(landscape 是横向，portrait 是纵向)。

5、TextProssBar 显示文字

```
public class TextProgressBar extends RelativeLayout implements OnChronometerTickListener {
    public static final String TAG = \"TextProgressBar\";
    static final int CHRONOMETER_ID = android.R.id.text1;
    static final int PROGRESSBAR_ID = android.R.id.progress;
    Chronometer mChronometer = null;
    ProgressBar mProgressBar = null;
    long mDurationBase = -1;
    int mDuration = -1;
    boolean mChronometerFollow = false;
    int mChronometerGravity = Gravity.NO_GRAVITY;

    public TextProgressBar(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }
    public TextProgressBar(Context context, AttributeSet attrs) {
        super(context, attrs);
    }
    public TextProgressBar(Context context) {
        super(context);
    }
    //关键部分在这里
    @Override
    public void addView(View child, int index, ViewGroup.LayoutParams params) {
        super.addView(child, index, params);

        int childId = child.getId();
        if (childId == CHRONOMETER_ID && child instanceof Chronometer) {
            mChronometer = (Chronometer) child;
            mChronometer.setOnChronometerTickListener(this);

            // Check if Chronometer should move with with ProgressBar
            mChronometerFollow = (params.width == ViewGroup.LayoutParams.WRAP_CONTENT);
            mChronometerGravity = (mChronometer.getGravity() & Gravity.HORIZONTAL_GRAVITY_MASK);

        } else if (childId == PROGRESSBAR_ID && child instanceof ProgressBar) {
            mProgressBar = (ProgressBar) child;
        }
    }

    @android.view.RemotableViewMethod
```

```
    public void setDurationBase(long durationBase) {
        mDurationBase = durationBase;

        if (mProgressBar == null || mChronometer == null) {
            throw new RuntimeException(\"Expecting child ProgressBar with id \" +
                    \"\'android.R.id.progress\' and Chronometer id \'android.R.id.text1\'\");
        }

        // Update the ProgressBar maximum relative to Chronometer base
        mDuration = (int) (durationBase - mChronometer.getBase());
        if (mDuration <= 0) {
            mDuration = 1;
        }
        mProgressBar.setMax(mDuration);
    }


    public void onChronometerTick(Chronometer chronometer) {
        if (mProgressBar == null) {
            throw new RuntimeException(
                \"Expecting child ProgressBar with id \'android.R.id.progress\'\");
        }

        // Stop Chronometer if we\'re past duration
        long now = SystemClock.elapsedRealtime();
        if (now >= mDurationBase) {
            mChronometer.stop();
        }

        int remaining = (int) (mDurationBase - now);
        mProgressBar.setProgress(mDuration - remaining);


        if (mChronometerFollow) {
            RelativeLayout.LayoutParams params;


            params = (RelativeLayout.LayoutParams) mProgressBar.getLayoutParams();
            int contentWidth = mProgressBar.getWidth() - (params.leftMargin +
params.rightMargin);
            int leadingEdge = ((contentWidth * mProgressBar.getProgress()) /
                    mProgressBar.getMax()) + params.leftMargin;
```

```
                int adjustLeft = 0;
                int textWidth = mChronometer.getWidth();
                if (mChronometerGravity == Gravity.RIGHT) {
                    adjustLeft = -textWidth;
                } else if (mChronometerGravity == Gravity.CENTER_HORIZONTAL) {
                    adjustLeft = -(textWidth / 2);
                }


                leadingEdge += adjustLeft;
                int rightLimit = contentWidth - params.rightMargin - textWidth;
                if (leadingEdge < params.leftMargin) {
                    leadingEdge = params.leftMargin;
                } else if (leadingEdge > rightLimit) {
                    leadingEdge = rightLimit;
                }

                params = (RelativeLayout.LayoutParams) mChronometer.getLayoutParams();
                params.leftMargin = leadingEdge;


                mChronometer.requestLayout();

            }
        }
}
```

6、TextView 的效果

1、TextView 的 Html 效果

如图：

代码如下：

```
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
```

```
TextView textView = (TextView) findViewById(R.id.cmd100);
textView.setText(Html.fromHtml("Hello <b>CMD100</b>,<font color=\"red\"> hello GM2.0!!</font> <font color=\"#006666\"><b>CMD100 GM2.0</b> 即将上线！大家加油，都把自己该准备的准备好咯。</font>"));
}
```

2、TextView 实现下划线效果：

方法 1、
tv.getPaint().setFlags(Paint.UNDERLINE_TEXT_FLAG);//下划线
方法 2、
tv.setText(Html.fromHtml("<u>使用 html 实现下划线样式</u>"));

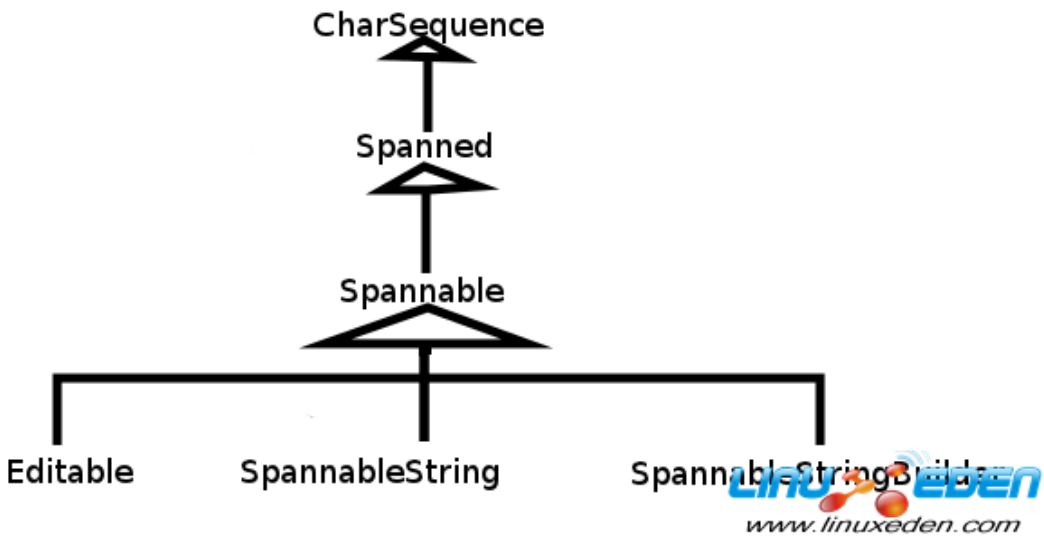
3、Spanned 实现 TextView 的各种样式

在开发应用过程中经常会遇到显示一些不同的字体风格的信息犹如默认的 LockScreen 上面的时间和充电信息。对于类似的情况，可能第一反应就是用不同的多个 TextView 来实现，对于每个 TextView 设置不同的字体风格以满足需求。

这里推荐的做法是使用 Android.text.*;和 android.text.style.*;下面的组件来实现 RichText：也即在同一个 TextView 中设置不同的字体风格。对于某些应用，比如文本编辑，记事本，彩信，短信等地方，还必须使用这些组件才能达到想到的显示效果。

主要的基本工具类有 Android.text.Spanned; android.text.SpannableString; android.text.SpannableStringBuilder;使用这些类来代替常规 String。SpannableString 和 SpannableStringBuilder 可以用来设置不同的 Span，这些 Span 便是用于实现 Rich Text，比如粗体，斜体，前景色，背景色，字体大小，字体风格等等，android.text.style.*中定义了很多的 Span 类型可供使用。

这是相关的 API 的 Class General Hierarchy：



因为 Spannable 等最终都实现了 CharSequence 接口，所以可以直接把 SpannableString 和 SpannableStringBuilder 通过 TextView.setText()设置给 TextView。

使用方法

当要显示 Rich Text 信息的时候，可以使用创建一个 SpannableString 或 SpannableStringBuilder，它们的区别在于 SpannableString 像一个 String 一样，构造对象的时候传入一个 String，之后再无法更改 String 的内容，也无法拼接多个 SpannableString；而 SpannableStringBuilder 则更像是 StringBuilder，它可以通过其 append()方法来拼接多个 String：

```
SpannableString word = new SpannableString("The quick fox jumps over the lazy dog");

SpannableStringBuilder multiWord = new SpannableStringBuilder();
multiWord.append("The Quick Fox");
multiWord.append("jumps over");
multiWord.append("the lazy dog");
```

创建完 Spannable 对象后，就可以为它们设置 Span 来实现想要的 Rich Text 了，常见的 Span 有：

AbsoluteSizeSpan(int size) ---- 设置字体大小，参数是绝对数值，相当于 Word 中的字体大小
RelativeSizeSpan(float proportion) ---- 设置字体大小，参数是相对于默认字体大小的倍数，比如默认字体大小是 x，那么设置后的字体大小就是 x*proportion，这个用起来比较灵活，proportion>1 就是放大(zoom in), proportion<1 就是缩小(zoom out)

ScaleXSpan(float proportion) ---- 缩放字体，与上面的类似，默认为 1,设置后就是原来的乘以 proportion，大于 1 时放大(zoon in)，小于时缩小(zoom out)
BackgroundColorSpan(int color) ---- 背景着色，参数是颜色数值，可以直接使用 Android.graphics.Typeface 里面定义的常量，如 Typeface.BOLD，Typeface.ITALIC 等等。
StrikethroughSpan----如果设置了此风格，会有一条线从中间穿过所有的字，就像被划掉一样

对于这些 Sytle span 在使用的时候通常只传上面所说明的构造参数即可，不需要设置其他的属性，如果需要的话，也可以对它们设置其他的属性。
SpannableString 和 SpannableStringBuilder 都有一个设置上述 Span 的方法：

```
/**
 * Set the style span to Spannable, such as SpannableString or SpannableStringBuilder
 * @param what --- the style span, such as StyleSpan
 * @param start --- the starting index of characters to which the style span to apply
 * @param end --- the ending index of characters to which the style span to apply
 * @param flags --- the flag specified to control
 */
setSpan(Object what, int start, int end, int flags);
```

其中参数 what 是要设置的 Style span，start 和 end 则是标识 String 中 Span 的起始位置，而 flags 是用于控制行为的，通常设置为 0 或 Spanned 中定义的常量，常用的有：

Spanned.SPAN_EXCLUSIVE_EXCLUSIVE --- 不包含两端 start 和 end 所在的端点

Spanned.SPAN_EXCLUSIVE_INCLUSIVE --- 不包含端 start，但包含 end 所在的端点
Spanned.SPAN_INCLUSIVE_EXCLUSIVE --- 包含两端 start，但不包含 end 所在的端点
Spanned.SPAN_INCLUSIVE_INCLUSIVE--- 包含两端 start 和 end 所在的端点
这里理解起来就好像数学中定义区间，开区间还是闭区间一样的。这里要重点说明下关于参数 0,有很多时候，如果设置了上述的参数，那么 Span 会从 start 应用到 Text 结尾，而不是在 start 和 end 二者之间，这个时候就需要使用 Flag 0。

背景色和前景色设置案例：

```
textview=(TextView)findViewById(R.id.textview);
SpannableStringBuilder style=new SpannableStringBuilder(strs);
style.setSpan(new
BackgroundColorSpan(Color.RED),start,end,Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
style.setSpan(new
ForegroundColorSpan(Color.RED),7,9,Spannable.SPAN_EXCLUSIVE_INCLUSIVE);
textview.setText(style);
```

参数说明：
Spannable.SPAN_EXCLUSIVE_EXCLUSIVE
API 里面解释：Spans of type SPAN_EXCLUSIVE_EXCLUSIVE do not expand to include text inserted at either their starting or ending point. They can never have a length of 0 and are automatically removed from the buffer if all the text they cover is removed.
即在原文本头或尾追加新文本的样式不受原文本样式影响，原文本高亮，新追加文本不高亮。

Spannable.SPAN_EXCLUSIVE_INCLUSIVE
API 里面解释：Non-0-length spans of type SPAN_INCLUSIVE_EXCLUSIVE expand to include text inserted at their ending point but not at their starting point. When 0-length, they behave like

points.
即在原文本尾追加新文本的样式受原文本样式影响，原来文本尾高亮，新追加文本也高亮。

7、通过 HttpClient 从指定 server 获取数据

```
DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpGet method = new HttpGet("http://www.baidu.com/1.html");
        HttpResponse resp;
        Reader reader = null;
        try {
            // AllClientPNames.TIMEOUT
            HttpParams params = new BasicHttpParams();
            params.setIntParameter(AllClientPNames.CONNECTION_TIMEOUT,
10000);
            httpClient.setParams(params);
            resp = httpClient.execute(method);
            int status = resp.getStatusLine().getStatusCode();
            if (status != HttpStatus.SC_OK) return false;
            // HttpStatus.SC_OK;
            return true;
        } catch (ClientProtocolException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } finally {
            if (reader != null) try {
                reader.close();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
```

8、隐藏小键盘

程序启动后直接弹出软键盘，不能直接在 OnCreate 中设置，必须等 View 绘制事件完毕才可以弹出，需要用到 Timer 辅助实现，如果要实现输入的功能，必须让 EditText 获得焦点。

代码如下：

//应用启动后自动打开输入法

```
@Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

         Timer timer = new Timer();
         timer.schedule(new TimerTask() {
             @Override
             public void run() {
                  InputMethodManager imm =
(InputMethodManager)MainActivity.this.getSystemService(INPUT_METHOD_SERVICE);
                   imm.toggleSoftInput(0, InputMethodManager.HIDE_NOT_ALWAYS);
                    Toast.makeText(MainActivity.this, "show",
Toast.LENGTH_SHORT).show();
                }
            }, 1000);   //在一秒后打开
    }
```

自动关闭：

```
 InputMethodManager imm =
(InputMethodManager)getSystemService(Context.INPUT_METHOD_SERVICE);
 imm.hideSoftInputFromWindow(editTextField.getWindowToken(), 0);
```

禁止弹出：

查了一下 Android SDK 的说明，发现可以通过设置 Activity 的一个属性来解决这个问题，比如可以在 AndroidManifest.xml 中这样写：

```
< activity android:name=".CategoryList"
        android:label="@string/app_name"
        android:windowSoftInputMode="stateVisible|adjustPan" >
< /activity >
```

或

```
<activity
    android:name=".ClientSearchViewActivity"
    android:label="@string/app_name"
    android:windowSoftInputMode="adjustUnspecified|stateHidden"
    android:configChanges="orientation|keyboardHidden">
</activity>
```

//EditText 有焦点阻止输入法弹出

```
editText.setOnTouchListener(new OnTouchListener() {
        public boolean onTouch(View v, MotionEvent event) {
          // TODO Auto-generated method stub
          //记住 EditText 的 InputType 现在是 password
              int inType = editText.getInputType(); // backup the input type
              editText.setInputType(InputType.TYPE_NULL); // disable soft input
              editText.onTouchEvent(event); // call native handler
              editText.setInputType(inType); // restore input type
              editText.setSelection(editText.getText().length());
              return true;
          }
    });
```

9、响应 Touch

```
Public boolean onTouch(View v, MotionEvent event) {
//响应操作
      return true;
}
```

10、Activity 间的通信

1、Bundle 传值

例如：A 调用并将数据传给 B

1、A 中调用代码：

```
Intent intent = new Intent();
intent.setClass(A.this, B.class);
Bundle mBundle = new Bundle();
mBundle.putString("passData", "hello！B");//压入数据，passData 是自定义的识别标志，还可以 put 添加多个 Int、String 等类型的数据
intent.putExtras(mBundle);
startActivity(intent);
A.this.finish();
```

2、B 中接受数据的代码：

```
Bundle bundle = getIntent().getExtras();
String data = bundle.getString( "passData" );// data 的值为 hello！B
```

注：如果要传送的数据条目颇多的话，可以将识别标志抽取为一个个常量来引用。既避免出

错，又方便维护。

2、利用 startActivityForResult 与 onActivityResult 方法

例如：A 调用并传值给 B，B 得到 A 的值，还需在结束时返回一个值给 A；
这里有两种较为普遍的方式：
1、 显示调用：即 A 是可知的，B 也是可知的，A 直接通过 B 的名字来调用它，方法如下：
<1>A 中代码：

```
static final int A_REQUEST = 0;
//开始调用，没有 finish()
Intent intent = new Intent();
intent.setClass(A.this, B.class);
startActivityForResult(intent, A_REQUEST);
//回调函数，可以在 eclipse 中按快捷键 Alt+Shift+S ，弹出菜单后再按 V 键，勾选 Activity 类别中所要覆盖的方法 onActivityResult(), 便会自动生成
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == A_REQUEST) {
        //根据由 B 返回的不同结果，进而执行不同的操作，data 是由 B 返回来的数据
      if (resultCode == RESULT_CANCELED)

      else if(resultCode == RESULT_OK) {
            //由于<2>中 B 调用的 setResult 方法中第一个参数是 RESULT_OK，因此 A 执行此处，并且可以获得 B 返回的数据，如下：
            String getFromB = data.getExtras().getString("DataKey");//getFromB 的值为 ：Hello A!

       }
    }
}
```

<2>B 中的代码：

```
//在 onCreate 方法中添加
Bundle bundle = new Bundle();
bundle.putString("DataKey", "Hello A!");
Intent mIntent = new Intent();
mIntent.putExtras(bundle);
setResult(RESULT_OK, mIntent);
finish();
```

2、隐式调用：即 B 的名字不必知道，被调用的 Activity 由系统来确定。比如，我们在手机中常会遇到上传微博头像的操作，在我们点击选择头像时，系统会自动给我们选择一个可以浏览图像文件的外部应用程序来供我们选择，如果有多个可以进行选择的应用，则会弹出一个列表供我们使用。选定一个图像文件后，外部应用便会返回被选文件的 URI 供微博应用程序使用。此即为隐式调用。
常用的隐式调用有：

<1>此为调用录音程序，返回录音文件的 URI

```
Intent intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
startActivityForResult(intent, 0);
```

<2>此为调用拍照程序进行录像，返回视频文件的 URI

```
Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
startActivityForResult(intent, 0);
```

<3>此为调用拍照程序进行拍照，返回照片的 URI

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(intent, 0);
```

<4>此为调用系统的图片剪裁程序，返回剪裁后的图像的 URI

```
Intent intent = new Intent(Intent.ACTION_PICK, null);
intent.setDataAndType(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
IMAGE_UNSPECIFIED);
startActivityForResult(intent, 0);
```

<5>浏览图像获取系统某个图像文件的方法，返回所选图片的 URI

```
Intent localIntent1 = new Intent();
localIntent1.setType("image/*");
localIntent1.setAction("android.intent.action.GET_CONTENT");
Intent localIntent2 = Intent.createChooser(localIntent1,"选择图片：");
startActivityForResult(localIntent2, 0);
```

<6>调用系统的语音识别

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Speech recognition demo");
startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
```

其回调函数为：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if (requestCode == VOICE_RECOGNITION_REQUEST_CODE && resultCode ==
RESULT_OK) {
// Fill the list view with the strings the recognizer thought it could have heard
ArrayList matches = data.getStringArrayListExtra(
RecognizerIntent.EXTRA_RESULTS);
mList.setAdapter(new ArrayAdapter(this, android.R.layout.simple_list_item_1,
matches));
}
super.onActivityResult(requestCode, resultCode, data);
}
```

根据不同的 Action，调用不同的外部应用，这里不再一一列举。
需要说明的是：如何通过得到的 URI 获得该文件的绝对路径，
例如：根据返回图像文件的 URI，获得其绝对路径

方法如下：

```
Uri picPath = data.getData();
String absolutePath = getImageAbsolutePath(picPath));

/**此为笔者构造的一个得到图像文件绝对路径的方法*/
protected String getImageAbsolutePath(Uri uri) {
    String[] proj = { MediaStore.Images.Media.DATA };
    Cursor cursor = managedQuery(uri, proj, null, null, null);
    int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
    cursor.moveToFirst();
    return cursor.getString(column_index);
}
**********
```

11、使程序完全退出

1、使用 ActivityManager：

```
ActivityManager am = (ActivityManager)getSystemService (Context.ACTIVITY_SERVICE);
am.restartPackage(getPackageName()); //虽为 restart，但并不是重启
```

系统会将该包下的所有活动、服务全部杀掉，要注意加上权限

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

2、终止进程：（不推荐）

```
finish();
android.os.Process.killProcess(android.os.Process.myPid());
```

注：进程被终止时，查看一下 Log 输出，ActivityManager 会抛出错误，但对用户来说，并无异常

12、列出所有音乐文件

```
String[] projection = new String[] {
    MediaStore.Audio.Media.ALBUM,
    MediaStore.Audio.Media.ARTIST,
    MediaStore.Audio.Media.TITLE
};

Cursor cursor = managedQuery(
    MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, //URI
    projection, //columns to return
    null, //rows to return -    all
    null, //selection arguments - none
    null // sort order - default
);

if (cursor != null) {
```

```
        cursor.moveToFirst();
        while (!cursor.isLast()) {
                Log.d("MediaTest", "Found track:");
                Log.d("MediaTest", "Album " + cursor.getString(0));
                Log.d("MediaTest", "Artist " + cursor.getString(1));
                Log.d("MediaTest", "Track " + cursor.getString(2));
                cursor.moveToNext();
                }

}
```

13、使用 Intent ACTION 调用系统程序

显示网页:

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

显示地图:

```
Uri uri = Uri.parse("geo:38.899533,-77.036476");
Intent it = new Intent(Intent.Action_VIEW,uri);
startActivity(it);
```

路径规划:

```
Uri.parse("http://maps.google.com/maps?f=d&saddr=startLat%20startLng&daddr=endLat%20en
d
Lng&hl=en");
Intent it = new Intent(Intent.ACTION_VIEW,URI);
startActivity(it);
```

拨打电话:

```
案例 1、
Uri uri = Uri.parse("tel:xxxxxx");
Intent it = new Intent(Intent.ACTION_DIAL, uri); //直接呼出
startActivity(it);
案例 2、
Uri uri = Uri.parse("tel.xxxxxx");
```

```
Intent it =new Intent(Intent.ACTION_CALL,uri); //不呼出
startActivity(it);
```
注：要使用这个必须在配置文件中加入
```
<uses-permission id="android.permission.CALL_PHONE" />
```

发送 SMS/MMS

案例 1、
```
Intent it = new Intent(Intent.ACTION_VIEW);
it.putExtra("sms_body", "The SMS text");
it.setType("vnd.android-dir/mms-sms");
startActivity(it);
```
案例 2、
发送短信
```
Uri uri = Uri.parse("smsto:0800000123");
Intent it = new Intent(Intent.ACTION_SENDTO, uri);
it.putExtra("sms_body", "The SMS text");
startActivity(it);
```
案例 3、
发送彩信
```
Uri uri = Uri.parse("content://media/external/images/media/23");
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra("sms_body", "some text");
it.putExtra(Intent.EXTRA_STREAM, uri);
it.setType("image/png");
startActivity(it);
```

发送 Email

案例 1、
```
Uri uri = Uri.parse("mailto:xxx@abc.com");
Intent it = new Intent(Intent.ACTION_SENDTO, uri);
startActivity(it);
```
案例 2、
```
Intent it = new Intent(Intent.ACTION_SEND);
String[] tos={"me@abc.com"};
it.putExtra(Intent.EXTRA_EMAIL, tos); //String 可能无法传递，String[]可以
it.putExtra(Intent.EXTRA_TEXT, "The email body text");
it.setType("text/plain");
startActivity(Intent.createChooser(it, "请选择邮件客户端软件：t"));
```
案例 3、
```
Intent it=new Intent(Intent.ACTION_SEND);
String[] tos={"me@abc.com"};
```

String[] ccs={"you@abc.com"};
it.putExtra(Intent.EXTRA_EMAIL, tos); //String 可能无法传递，String[]可以

it.putExtra(Intent.EXTRA_CC, ccs); //String 可能无法传递，String[]可以
it.putExtra(Intent.EXTRA_TEXT, "The email body text");
it.putExtra(Intent.EXTRA_SUBJECT, "The email subject text");
it.setType("message/rfc822");
startActivity(Intent.createChooser(it, "请选择邮件客户端软件："));
案例 4、
添加附件
Intent it = new Intent(Intent.ACTION_SEND);
it.putExtra(Intent.EXTRA_SUBJECT, "The email subject text");
it.putExtra(Intent.EXTRA_STREAM, "file:///sdcard/mysong.mp3");
sendIntent.setType("audio/mp3");
startActivity(Intent.createChooser(it, "请选择邮件客户端软件："));

为程序添加一个“分享”

1、在 manifest 中添加：

```
<intent-filter>
      <action android:name="android.intent.action.SEND" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:mimeType="image/*" />
  </intent-filter >
```

2、启动代码：

```
Intent intent = new Intent(Intent.ACTION_SEND);
startActivity(Intent.createChooser(intent , "Send options"));
```

打开多种类型的文件：

如果需要打开多种类型的文件，可以自定义返回 Intent 型的不同方法，分别用来打开不同类型的文件。
各种 java 方法如下：

```
// android 获取一个用于打开 HTML 文件的 intent
public static Intent getHtmlFileIntent(String param){
        Uri uri = Uri.parse(param).buildUpon().encodedAuthority("com.android.htmlfileprovider").scheme("content").encodedPath(param).build();
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.setDataAndType(uri, "text/html");
        return intent;
```

```
}

// android 获取一个用于打开图片文件的 intent
public static Intent getImageFileIntent(String param) {
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "image/*");
        return intent;
}

// android 获取一个用于打开 PDF 文件的 intent
public static Intent getPdfFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "application/pdf");
        return intent;
}

// android 获取一个用于打开文本文件的 intent
public static Intent getTextFileIntent(String param, boolean paramBoolean){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        if (paramBoolean) {
                Uri uri1 = Uri.parse(param);
                intent.setDataAndType(uri1, "text/plain");
        } else {
                Uri uri2 = Uri.fromFile(new File(param));
                intent.setDataAndType(uri2, "text/plain");
        }
        return intent;
}

// android 获取一个用于打开音频文件的 intent
public static Intent getAudioFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.putExtra("oneshot", 0);
        intent.putExtra("configchange", 0);
        Uri uri = Uri.fromFile(new File(param));
```

```
        intent.setDataAndType(uri, "audio/*");
        return intent;
    }
    // android 获取一个用于打开视频文件的 intent
    public static Intent getVideoFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.putExtra("oneshot", 0);
        intent.putExtra("configchange", 0);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "video/*");
        return intent;
    }

    // android 获取一个用于打开 CHM 文件的 intent
    public static Intent getChmFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "application/x-chm");
        return intent;
    }
    // android 获取一个用于打开 Word 文件的 intent
    public static Intent getWordFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "application/msword");
        return intent;
    }
    // android 获取一个用于打开 Excel 文件的 intent
    public static Intent getExcelFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "application/vnd.ms-excel");
        return intent;
    }
    // android 获取一个用于打开 PPT 文件的 intent
    public static Intent getPptFileIntent(String param){
        Intent intent = new Intent("android.intent.action.VIEW");
```

```
        intent.addCategory("android.intent.category.DEFAULT");
        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        Uri uri = Uri.fromFile(new File(param));
        intent.setDataAndType(uri, "application/vnd.ms-powerpoint");
        return intent;
    }
```

调用示例：

如：打开 pdf 文件的两种方法：

```
Intent it = getPdfFileIntent("file:///mnt/sdcard/helphelp.pdf");
startActivity( it );
```

或：

```
Intent it = getPdfFileIntent("/mnt/sdcard/helphelp.pdf");//SD 卡主目录
startActivity( it );
```

Uninstall　程序

```
1. Uri uri = Uri.fromParts("package", strPackageName, null);
2. Intent it = new Intent(Intent.ACTION_DELETE, uri);
3. startActivity(it);
```

14、将 Uri 转为绝对路径

方法 1：

```
Uri uri = data.getData():
String[] proj = { MediaStore.Images.Media.DATA };
Cursor actualimagecursor = managedQuery(uri,proj,null,null,null);
int actual_image_column_index = actualimagecursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
actualimagecursor.moveToFirst();
String img_path = actualimagecursor.getString(actual_image_column_index);
```

方法 2：

```
Uri uri = data.getData();
ContentResolver cr = this.getContentResolver();
Cursor cursor = cr.query(uri, null, null, null, null);
cursor.moveToFirst();
for (int i = 0; i < cursor.getColumnCount(); i++) {
    System.out.println(i+"----------"+cursor.getString(i));
}
```

这个可以遍历 cursor，其中_data 字段 就是图片的物理路径

15、Android 支持多种语言

为了能让更多的人用上我们的程序，程序必须有各种语言的版本（英语是必须的）。原来我的想法是先判断 Android 系统当前设置的语言，再加载相应的 layout，或是 String,但是网络上搜索了一下发现，其实 Google 早就已经帮我们考虑这个问题了，我们根本就不需要自己判断机器语言，只要写不同语言的 xml 文件，并放到指定目录，系统会自动识别调用。
Android sdk 从 1.5 开始,不在需要在程序里边单独设置多语言代码,您只需简单的修改基本 res 配置,就可以实现多语言版本.
一般来讲，我们都把 string 先放到 values/strings.xml 文件里，然后在程序里需要显示文字的地方调用，所以我们只要多建几个 strings.xml，对应不同的语言就 OK，下面一步一步来：

具体方法：
打开新建 Android XML File 对话框，
添加列表里的，Region 和 Language,Region 值填写两位地区代码（美国为 US，中国为 ZH，台湾（繁体）为 TW），Language 填写两位语言代码（英语 en,中文 cn）。有一点要注意，简繁体中文并不是由 Language 识别的，不管是简体还是繁体，Language 都填 cn,如果要显示繁体，Region 填为 TW 就可以了

也可以手动创建目录
如：
res/valuse/strings.xml 中如下设置默认语言。
res/values-zh-rCN/string.xml 中设置简体中文。
res/values-zh-rTW/string.xml 中设置繁体中文。
res/values-en-rUS/string.xml 中设置美国英语。

默认情况下会根据系统语言来选择，也可自己配置程序显示的语言，如下：

```
Resources res = getResources();
Configuration config = res.getConfiguration();
config.locale = Locale.SIMPLIFIED_CHINESE;//此为简体中文
DisplayMetrics dm = res.getDisplayMetrics();
res.updateConfiguration(config, dm);
```

16、四种动画的设置属性

1、尺寸伸缩动画效果

属性：interpolator 指定一个动画的插入器

在我试验过程中，使用 android.res.anim 中的资源时候发现有三种动画插入器

accelerate_decelerate_interpolator　加速-减速 动画插入器
accelerate_interpolator　　加速-动画插入器
decelerate_interpolator　　减速- 动画插入器

其他的属于特定的动画效果

浮点型值：

fromXScale 属性为动画起始时 X 坐标上的伸缩尺寸
toXScale　属性为动画结束时 X 坐标上的伸缩尺寸

fromYScale 属性为动画起始时 Y 坐标上的伸缩尺寸
toYScale　属性为动画结束时 Y 坐标上的伸缩尺寸

说明:
以上四种属性值

0.0 表示收缩到没有
1.0 表示正常无伸缩
值小于 1.0 表示收缩
值大于 1.0 表示放大

pivotX　属性为动画相对于物件的 X 坐标的开始位置
pivotY　属性为动画相对于物件的 Y 坐标的开始位置

说明:
以上两个属性值 从 0%-100%中取值
50%为物件的 X 或 Y 方向坐标上的中点位置

长整型值：
duration　属性为动画持续时间
说明:
时间以毫秒为单位

布尔型值:
fillAfter 属性 当设置为 true ，该动画转化在动画结束后被应用

2、translate 位置转移动画效果

整型值:
fromXDelta 属性为动画起始时 X 坐标上的位置
toXDelta 属性为动画结束时 X 坐标上的位置

fromYDelta 属性为动画起始时 Y 坐标上的位置
toYDelta 属性为动画结束时 Y 坐标上的位置
注意:
没有指定 fromXType toXType fromYType toYType 时候，默认是以自己为相对参照物

长整型值：
duration 属性为动画持续时间
说明: 时间以毫秒为单位

3、rotate 旋转动画效果

属性：interpolator 指定一个动画的插入器

在我试验过程中，使用 android.res.anim 中的资源时候发现有三种动画插入器

accelerate_decelerate_interpolator 加速-减速 动画插入器
accelerate_interpolator 加速-动画插入器
decelerate_interpolator 减速- 动画插入器

其他的属于特定的动画效果
浮点数型值:
fromDegrees 属性为动画起始时物件的角度
toDegrees 属性为动画结束时物件旋转的角度 可以大于 360 度

说明:
当角度为负数——表示逆时针旋转
当角度为正数——表示顺时针旋转

(负数 from——to 正数:顺时针旋转)
(负数 from——to 负数:逆时针旋转)
(正数 from——to 正数:顺时针旋转)
(正数 from——to 负数:逆时针旋转)

pivotX　　属性为动画相对于物件的 X 坐标的开始位置
pivotY　　属性为动画相对于物件的 Y 坐标的开始位置

说明:
　　以上两个属性值 从 0%-100%中取值
50%为物件的 X 或 Y 方向坐标上的中点位置

长整型值：
duration　属性为动画持续时间
说明:
　　时间以毫秒为单位

4、透明度控制动画效果 alpha

浮点型值：
fromAlpha 属性为动画起始时透明度
toAlpha　 属性为动画结束时透明度
说明:
0.0 表示完全透明
1.0 表示完全不透明
　　以上值取 0.0-1.0 之间的 float 数据类型的数字

长整型值：
duration　属性为动画持续时间
说明:
　　时间以毫秒为单位

17、横竖屏状态获取

（1）首先在 manifest 文件的 Activity 生明中添加：
　　android:configChanges="keyboardHidden|orientation"
（2）在代码中覆盖如下方法，以获取不同屏幕横竖切换时的动作来进行不同的响应：

```
public void onConfigurationChanged(Configuration newConfig) {
    try {
        super.onConfigurationChanged(newConfig);
        if (this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
            // land
```

```
            } else if (this.getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_PORTRAIT) {
                // port

            }
        } catch (Exception ex) {
        }
    }
```

此方法也可以用来阻止横竖屏切换时 Activity 重启造成的问题诸如：对话框的消失、或 view 的还原。

18、获取手指在屏幕的左右滑动

在代码重写 onTouchEvent 方法：

```
    //记录触摸的坐标
    float from_x = 0;
    float from_y = 0;
    float to_x = 0;
    float to_y = 0;

    /**
     * 获取滑动动作
     */
    public boolean onTouchEvent(MotionEvent event) {
        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            from_x = event.getX();
            from_y = event.getY();
            break;
        case MotionEvent.ACTION_UP:
            to_x = event.getX();
            to_y = event.getY();
            float xdistance = from_x-to_x;
            float ydistance = from_y-to_y;

            Log.e("","Distamce+++++++++++++++" + xdistance + "__" + ydistance);
            if(ydistance-60<0 && xdistance-100>0) {
                Log.e("","-------左滑++++++++++++");

            } else if(ydistance-60<0 && xdistance+100<0) {
                Log.e("","-------右滑++++++++++++++");
```

```
            }
            break;
        case MotionEvent.ACTION_MOVE:

            break;
        }
        return super.onTouchEvent(event);
    }
```

注：这种方法亦有局限性，如果屏幕上布满控件时，则不易获取点击屏幕的动作。

19、解除屏幕锁

当 Android 手机(应该是所有的手机都这样)收到短信或者电话时,会自动点亮屏幕,解开屏幕锁,以方便用户即时操作,下面用代码来实现这一功能:

```
        KeyguardManager km= (KeyguardManager) getSystemService(Context.KEYGUARD_SERVICE);
        //得到键盘锁管理器对象
        KeyguardLock kl = km.newKeyguardLock("unLock");
        //参数是 LogCat 里用的 Tag
        kl.disableKeyguard();
        //解锁
        PowerManager pm=(PowerManager) getSystemService(Context.POWER_SERVICE);
        //获取电源管理器对象
        PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.ACQUIRE_CAUSES_WAKEUP | PowerManager.SCREEN_DIM_WAKE_LOCK, "bright");
        //获取 PowerManager.WakeLock 对象,后面的参数|表示同时传入两个值,最后的是 LogCat 里用的 Tag
        wl.acquire();
        //点亮屏幕
        wl.release();
```

需要在 AndroidManifest.xml 添加权限:

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

20、ViewFlippe 实现循环的动画

```
<ViewFlipper android:id="@+id/flipper"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:flipInterval="2000"> <!—循环时间-- >
<!—放置控件-- >

</ViewFlipper>

    // 启动动态标题
    ViewFlipper flipper = ((ViewFlipper) findViewById(R.id.flipper));
    flipper.startFlipping();
    flipper.setInAnimation(AnimationUtils.loadAnimation(this, R.anim.push_left_in));
    flipper.setOutAnimation(AnimationUtils.loadAnimation(this, R.anim.push_left_out));
```

21、播放 gif 动画

由于 Gif 本身就是动画，所以如果能够直接使用的话，会省去很多的麻烦。

要想播放 gif 动画，首先需要对 gif 动画进行解码，然后将 gif 中的每一帧提取出来，放在一个容器中，然后根据需要绘制每一帧，这样就实现了 gif 动画在手机中直接播放了 GameView.gif

1、

```
package eoe.giftest;


import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.view.View;


public class GameView extends View implements Runnable
{
Context mContext = null;
/* 声明 GifFrame 对象 */
GifFrame mGifFrame = null;
public GameView(Context context)
```

```
{
super(context);
mContext = context;
/* 解析 GIF 动画 */
mGifFrame=GifFrame.CreateGifImage(fileConnect(this.getResources().openRawResource(R.dr
awable.girl)));
/* 开启线程 */
new Thread(this).start();
}
public void onDraw(Canvas canvas)
{
super.onDraw(canvas);
/* 下一帧 */
mGifFrame.nextFrame();
/* 得到当前帧的图片 */
Bitmap b=mGifFrame.getImage();
/* 绘制当前帧的图片 */
if(b!=null)
canvas.drawBitmap(b,10,10,null);
}
/**
* 线程处理
*/


public void run()
{
while (!Thread.currentThread().isInterrupted())
{
try
{
Thread.sleep(100);
}
catch (InterruptedException e)
{
Thread.currentThread().interrupt();
}
//使用 postInvalidate 可以直接在线程中更新界面
postInvalidate();
}
}
/* 读取文件 */
public byte[] fileConnect(InputStream is)
{
```

```
try
{
ByteArrayOutputStream baos = new ByteArrayOutputStream();
int ch = 0;
while( (ch = is.read()) != -1)
{
baos.write(ch);
}
byte[] datas = baos.toByteArray();
baos.close();
baos = null;
is.close();
is = null;
return datas;
}
catch(Exception e)
{
return null;
}
}
}
```

2、

```
package eoe.giftest;


import java.util.Vector;
import android.graphics.Bitmap;
public class GifFrame
{
/* 保存 gif 中所有帧的向量 */
private Vector frames;
/* 当前播放的帧的索引 */
private int index;
public GifFrame()
{
frames = new Vector(1);
index = 0;
}
/* 添加一帧 */
public void addImage(Bitmap image)
{
frames.addElement(image);
}
```

```
/*  返回帧数  */
public int size()
{
return frames.size();
}
/*  得到当前帧的图片  */
public Bitmap getImage()
{
if (size() == 0)
{
return null;
}else{
return (Bitmap) frames.elementAt(index);
}
}
/*  下一帧  */
public void nextFrame()
{
if (index + 1 < size())
{
index++;
}else{
index = 0;
}
}
/*  创建 GifFrame */
public static GifFrame CreateGifImage(byte abyte0[])
{
try
{
GifFrame GF = new GifFrame();
Bitmap image = null;
GifDecoder gifdecoder = new GifDecoder(abyte0);
for (; gifdecoder.moreFrames(); gifdecoder.nextFrame())
{
try
{
image = gifdecoder.decodeImage();
if (GF != null && image != null)
{
GF.addImage(image);
}
continue;
}catch (Exception e){
```

```
e.printStackTrace();
}
break;
}
gifdecoder.clear();
gifdecoder = null;
return GF;
}
catch (Exception e)
{
e.printStackTrace();
return null;
}
}
}
```

3、

```
package eoe.giftest;


import android.app.Activity;
import android.os.Bundle;


public class GifShow extends Activity {
/** Called when the activity is first created. */
GameView myGameView = null;


@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
myGameView = new GameView(this);
setContentView(myGameView);
}
}
```

4、调用代码：

```
        // 显示动画图片
        GameView myGifView = null;
        GameView.getGifResource = R.drawable.over_thing;// 设置要显示的 gif 动画
        myGifView = new GameView(this);
        LinearLayout gifLayout = (LinearLayout)
ovetDialogView.findViewById(R.id.gif_show);
```

gifLayout.addView(myGifView);

注：对于帧数比较多的 gif，则需要边解析边播放了！否则等解析完成之后在播放，就会卡、

22、飞行模式转换解析

代码：

```
private void setAirplaneModeOn(boolean enabling) {
mCheckBoxPref.setEnabled(false);
mCheckBoxPref.setSummary(enabling ? R.string.airplane_mode_turning_on
R.string.airplane_mode_turning_off);
// Change the system setting

Settings.System.putInt(mContext.getContentResolver(),
Settings.System.AIRPLANE_MODE_ON,enabling ? 1 : 0);

// Post the intent
Intent intent = new Intent(Intent.ACTION_AIRPLANE_MODE_CHANGED);
intent.putExtra("state", enabling);
mContext.sendBroadcast(intent);
}
```

待机状态下，飞行模式切换流程分析：

飞行模式切换比较复杂，它状态改变时涉及到极大模块状态切换：
GSM 模块，蓝牙模块，wifi 模块。
飞行模式的 enabler 层会发送广播消息：ACTION_AIRPLANE_MODE_CHANGED

因为 GSM ，蓝牙，wifi 模块分别注册了对 ACTION_AIRPLANE_MODE_CHANGED 消息的监测，所以收到

该消息后，模块会进行切换。

BluetoothDeviceService.java

开启蓝牙：enable(false);

关闭蓝牙：disable(false);

PhoneApp.java (packages\apps\phone\src\com\android\phone)

设置 GSM 模块状态 phone.setRadioPower(enabled);

WifiService.java

设置 wifi 状态 setWifiEnabledBlocking(wifiEnabled, false, Process.myUid());

GSM 模块切换过程分析：

phone.setRadioPower(enabled)调用的是：

文件 GSMPhone.java 中的的函数：

public void setRadioPower(boolean power)

mSST.setRadioPower(power);
因为有 ServiceStateTracker mSST;
mSST.setRadioPower 调用的是文件 ServiceStateTracker.java 中的函数：
public void setRadioPower(boolean power)
mDesiredPowerState = power;
setPowerStateToDesired();
cm.setRadioPower(true, null);
或者
cm.setRadioPower(false, null);
因为有：
CommandsInterface cm;
public final class RIL extends BaseCommands implements CommandsInterface
所以 cm.setRadioPower 调用的是文件 RIL.java 中的函数：
public void setRadioPower(boolean on, Message result)
RILRequest rr = RILRequest.obtain(RIL_REQUEST_RADIO_POWER, result);
rr.mp.writeInt(1);
...
send(rr)
通过 send 向 rild 发送 RIL_REQUEST_RADIO_POWER 请求来开启或者关闭 GSM 模块。
rild 数据接收流程：
收到 RIL_REQUEST_RADIO_POWER 执行：
requestRadioPower(data, datalen, t);
然后根据条件往无线模块发送模块开启和关闭请求
主要的 at 命令有：
err = at_send_command("AT+CFUN=0", &p_response);
err = at_send_command("AT+CFUN=1", &p_response);

蓝牙模块切换过程分析：
enable(false);
蓝牙开启调用文件 BluetoothDeviceService.java 中的函数：
public synchronized boolean enable(boolean saveSetting)
setBluetoothState(BluetoothDevice.BLUETOOTH_STATE_TURNING_ON);
mEnableThread = new EnableThread(saveSetting);
mEnableThread.start();

disable(false)
蓝牙关闭调用文件 中的函数：
public synchronized boolean disable(boolean saveSetting)
setBluetoothState(BluetoothDevice.BLUETOOTH_STATE_TURNING_OFF);

wifi 模块切换过程分析：
广播 wifi 状态改变的消息：WIFI_STATE_CHANGED_ACTION
setWifiEnabledState(enable ? WIFI_STATE_ENABLING : WIFI_STATE_DISABLING,

uid);

更新 wifi 状态：
private void updateWifiState()

如果需要使能开启 wifi 那么会发送：
sendEnableMessage(true, false, mLastEnableUid);
sendStartMessage(strongestLockMode == WifiManager.WIFI_MODE_SCAN_ONLY);
mWifiHandler.sendEmptyMessage(MESSAGE_STOP_WIFI);

消息循环中处理命令消息：
public void handleMessage(Message msg)
如果使能 wifi：setWifiEnabledBlocking(true, msg.arg1 == 1, msg.arg2);
开启 wifi： mWifiStateTracker.setScanOnlyMode(msg.arg1 != 0);
setWifiEnabledBlocking(false, msg.arg1 == 1, msg.arg2);
断开 mWifiStateTracker.disconnectAndStop();

开启过程步骤：
1> 装载 wifi 驱动： WifiNative.loadDriver()
2> 启动后退 daemo supplicant： WifiNative.startSupplicant()

关闭过程步骤：
1> 停止后退 daemo supplicant：WifiNative.stopSupplicant()
2> 卸载 wifi 驱动: WifiNative.unloadDriver()
如果 wifi 状态默认为开启那么 WifiService 服务的构造函数：
WifiService(Context context, WifiStateTracker tracker)
boolean wifiEnabled = getPersistedWifiEnabled();
setWifiEnabledBlocking(wifiEnabled, false, Process.myUid());
会开启 wifi 模块。

23、实现按 home 键的效果

```
Intent i= new Intent(Intent.ACTION_MAIN);

i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK); //android123 提示如果是服务里调用，必须加入 new task 标识
i.addCategory(Intent.CATEGORY_HOME);
startActivity(i);
```

24、httpget 与 post

需求：用户登录（name:用户名，pwd:密码）

（一）HttpGet :doGet()方法

```
//doGet():将参数的键值对附加在 url 后面来传递
        public String getResultForHttpGet(String name,String pwd) throws
ClientProtocolException, IOException{
                //服务器 ： 服务器项目 ： servlet 名称
                String path="http://192.168.5.21:8080/test/test";
                String uri=path+"?name="+name+"&pwd="+pwd;
                //name:服务器端的用户名，pwd:服务器端的密码
                //注意字符串连接时不能带空格
                String result="";

                HttpGet httpGet=new HttpGet(uri);
                HttpResponse response=new DefaultHttpClient().execute(httpGet);
                if(response.getStatusLine().getStatusCode()==200){
                        HttpEntity entity=response.getEntity();
                        result=EntityUtils.toString(entity, HTTP.UTF_8);
                }
                return result;
        }
```

（二）HttpPost ： doPost()方法

```
//doPost():将参数打包到 http 报头中传递
        public String getReultForHttpPost(String name,String pwd) throws
ClientProtocolException, IOException{
                //服务器 ： 服务器项目 ： servlet 名称
                String path="http://192.168.5.21:8080/test/test";
                HttpPost httpPost=new HttpPost(path);
                List<NameValuePair>list=new ArrayList<NameValuePair>();
                list.add(new BasicNameValuePair("name", name));
                list.add(new BasicNameValuePair("pwd", pwd));
                httpPost.setEntity(new UrlEncodedFormEntity(list,HTTP.UTF_8));

                String result="";

                HttpResponse response=new DefaultHttpClient().execute(httpPost);
                if(response.getStatusLine().getStatusCode()==200){
                        HttpEntity entity=response.getEntity();
                        result=EntityUtils.toString(entity, HTTP.UTF_8);
                }
                return result;
        }
```

25、异步加载网络图片

在学习"Android 异步加载图像小结"这篇文章时，发现有些地方没写清楚，我就根据我的理解，把这篇文章的代码重写整理了一遍，下面就是我的整理。

下面测试使用的 layout 文件：

简单来说就是 LinearLayout 布局，其下放了 5 个 ImageView。

```
<?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">
    <TextView android:text="图片区域开始" android:id="@+id/textView2"
        android:layout_width="wrap_content"
android:layout_height="wrap_content"></TextView>
    <ImageView android:id="@+id/imageView1"
        android:layout_height="wrap_content" android:src="@drawable/icon"
        android:layout_width="wrap_content"></ImageView>
    <ImageView android:id="@+id/imageView2"
        android:layout_height="wrap_content" android:src="@drawable/icon"
        android:layout_width="wrap_content"></ImageView>
    <ImageView android:id="@+id/imageView3"
        android:layout_height="wrap_content" android:src="@drawable/icon"
        android:layout_width="wrap_content"></ImageView>
    <ImageView android:id="@+id/imageView4"
        android:layout_height="wrap_content" android:src="@drawable/icon"
        android:layout_width="wrap_content"></ImageView>
    <ImageView android:id="@+id/imageView5"
        android:layout_height="wrap_content" android:src="@drawable/icon"
        android:layout_width="wrap_content"></ImageView>
    <TextView android:text="图片区域结束" android:id="@+id/textView1"
        android:layout_width="wrap_content"
android:layout_height="wrap_content"></TextView>
</LinearLayout>
```

我们将演示的逻辑是异步从服务器上下载 5 张不同图片，依次放入这 5 个 ImageView。上下 2 个 TextView 是为了方便我们看是否阻塞了 UI 的显示。

当然 AndroidManifest.xml 文件中要配置好网络访问权限。

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

Handler+Runnable 模式

我们先看一个并不是异步线程加载的例子，使用 Handler+Runnable 模式。

这里为何不是新开线程的原因请参看这篇文章：Android Runnable 运行在那个线程 这里的代码其实是在 UI 主线程中下载图片的，而不是新开线程。

我们运行下面代码时，会发现他其实是阻塞了整个界面的显示，需要所有图片都加载完成后，才能显示界面。

```
package ghj1976.AndroidTest;

import java.io.IOException;
import java.net.URL;
import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.SystemClock;
import android.util.Log;
import android.widget.ImageView;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        loadImage("http://www.baidu.com/img/baidu_logo.gif", R.id.imageView1);
        loadImage(http://www.chinatelecom.com.cn/images/logo_new.gif",
                R.id.imageView2);
        loadImage("http://cache.soso.com/30d/img/web/logo.gif, R.id.imageView3);
        loadImage("http://csdnimg.cn/www/images/csdnindex_logo.gif",
                R.id.imageView4);
        loadImage("http://images.cnblogs.com/logo_small.gif",
                R.id.imageView5);
    }

    private Handler handler = new Handler();

    private void loadImage(final String url, final int id) {
        handler.post(new Runnable() {
            public void run() {
                Drawable drawable = null;
                try {
                    drawable = Drawable.createFromStream(
                            new URL(url).openStream(), "image.gif");
                } catch (IOException e) {
                    Log.d("test", e.getMessage());
                }
                if (drawable == null) {
```

```
                    Log.d("test", "null drawable");
                } else {
                    Log.d("test", "not null drawable");
                }
                                    // 为了测试缓存而模拟的网络延时
SystemClock.sleep(2000);                    ((ImageView) MainActivity.this.findViewById(id))
                        .setImageDrawable(drawable);
            }
        });
    }
}
```

Handler+Thread+Message 模式

这种模式使用了线程，所以可以看到异步加载的效果。

核心代码：

```
package ghj1976.AndroidTest;

import java.io.IOException;
import java.net.URL;
import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.SystemClock;
import android.util.Log;
import android.widget.ImageView;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        loadImage2("http://www.baidu.com/img/baidu_logo.gif", R.id.imageView1);
        loadImage2("http://www.chinatelecom.com.cn/images/logo_new.gif",
                R.id.imageView2);
        loadImage2("http://cache.soso.com/30d/img/web/logo.gif", R.id.imageView3);
        loadImage2("http://csdnimg.cn/www/images/csdnindex_logo.gif",
                R.id.imageView4);
```

```
        loadImage2("http://images.cnblogs.com/logo_small.gif",
                R.id.imageView5);
    }

    final Handler handler2 = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            ((ImageView) MainActivity.this.findViewById(msg.arg1))
                    .setImageDrawable((Drawable) msg.obj);
        }
    };

    // 采用 handler+Thread 模式实现多线程异步加载
    private void loadImage2(final String url, final int id) {
        Thread thread = new Thread() {
            @Override
            public void run() {
                Drawable drawable = null;
                try {
                    drawable = Drawable.createFromStream(
                            new URL(url).openStream(), "image.png");
                } catch (IOException e) {
                    Log.d("test", e.getMessage());
                }

                // 模拟网络延时
                SystemClock.sleep(2000);

                Message message = handler2.obtainMessage();
                message.arg1 = id;
                message.obj = drawable;
                handler2.sendMessage(message);
            }
        };
        thread.start();
        thread = null;
    }

}
```

这时候我们可以看到实现了异步加载， 界面打开时，五个 ImageView 都是没有图的，然后在各自线程下载完后才把图自动更新上去。

Handler+ExecutorService(线程池)+MessageQueue 模式

能开线程的个数毕竟是有限的，我们总不能开很多线程，对于手机更是如此。

这个例子是使用线程池。Android 拥有与 Java 相同的 ExecutorService 实现，我们就来用它。

线程池的基本思想还是一种对象池的思想，开辟一块内存空间，里面存放了众多(未死亡)的线程，池中线程执行调度由池管理器来处理。当有线程任务时，从池中取一个，执行完成后线程对象归池，这样可以避免反复创建线程对象所带来的性能开销，节省了系统的资源。

线程池的信息可以参看这篇文章：Java&Android 的线程池－ExecutorService 下面的演示例子是创建一个可重用固定线程数的线程池。

核心代码

```
package ghj1976.AndroidTest;

import java.io.IOException;
import java.net.URL;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.SystemClock;
import android.util.Log;
import android.widget.ImageView;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        loadImage3("http://www.baidu.com/img/baidu_logo.gif", R.id.imageView1);
        loadImage3("http://www.chinatelecom.com.cn/images/logo_new.gif",
                R.id.imageView2);
        loadImage3("http://cache.soso.com/30d/img/web/logo.gif",
                R.id.imageView3);
        loadImage3("http://csdnimg.cn/www/images/csdnindex_logo.gif",
                R.id.imageView4);
        loadImage3("http://images.cnblogs.com/logo_small.gif",
```

```
                R.id.imageView5);
    }

    private Handler handler = new Handler();

    private ExecutorService executorService = Executors.newFixedThreadPool(5);

    // 引入线程池来管理多线程
    private void loadImage3(final String url, final int id) {
        executorService.submit(new Runnable() {
            public void run() {
                try {
                    final Drawable drawable = Drawable.createFromStream(
                            new URL(url).openStream(), "image.png");
                    // 模拟网络延时
                    SystemClock.sleep(2000);
                    handler.post(new Runnable() {
                        public void run() {
                            ((ImageView) MainActivity.this.findViewById(id))
                                    .setImageDrawable(drawable);
                        }
                    });
                } catch (Exception e) {
                    throw new RuntimeException(e);
                }
            }
        });
    }
```

}这里我们象第一步一样使用了 handler.post(new Runnable() { 更新前段显示当然是在 UI 主线程，我们还有 executorService.submit(new Runnable() { 来确保下载是在线程池的线程中。

Handler+ExecutorService(线程池)+MessageQueue+缓存模式

下面比起前一个做了几个改造:

把整个代码封装在一个类中
为了避免出现同时多次下载同一幅图的问题,使用了本地缓存
封装的类:

```
package ghj1976.AndroidTest;
```

```
import java.lang.ref.SoftReference;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import android.graphics.drawable.Drawable;
import android.os.Handler;
import android.os.SystemClock;

public class AsyncImageLoader3 {
    // 为了加快速度，在内存中开启缓存（主要应用于重复图片较多时，或者同一个图片要多次被访问，比如在 ListView 时来回滚动）
    public Map<String, SoftReference<Drawable>> imageCache = new HashMap<String, SoftReference<Drawable>>();

    private ExecutorService executorService = Executors.newFixedThreadPool(5); // 固定五个线程来执行任务
    private final Handler handler = new Handler();

    /**
     *
     * @param imageUrl
     *                  图像 url 地址
     * @param callback
     *                  回调接口
     * @return 返回内存中缓存的图像，第一次加载返回 null
     */
    public Drawable loadDrawable(final String imageUrl,
            final ImageCallback callback) {
        // 如果缓存过就从缓存中取出数据
        if (imageCache.containsKey(imageUrl)) {
            SoftReference<Drawable> softReference = imageCache.get(imageUrl);
            if (softReference.get() != null) {
                return softReference.get();
            }
        }
        // 缓存中没有图像，则从网络上取出数据，并将取出的数据缓存到内存中
        executorService.submit(new Runnable() {
            public void run() {
                try {
                    final Drawable drawable = loadImageFromUrl(imageUrl);
```

```
                        imageCache.put(imageUrl, new SoftReference<Drawable>(
                                drawable));

                        handler.post(new Runnable() {
                            public void run() {
                                callback.imageLoaded(drawable);
                            }
                        });
                    } catch (Exception e) {
                        throw new RuntimeException(e);
                    }
                }
            });
            return null;
        }

        // 从网络上取数据方法
        protected Drawable loadImageFromUrl(String imageUrl) {
            try {
                // 测试时，模拟网络延时，实际时这行代码不能有
                SystemClock.sleep(2000);

                return Drawable.createFromStream(new URL(imageUrl).openStream(),
                        "image.png");

            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        }

        // 对外界开放的回调接口
        public interface ImageCallback {
            // 注意 此方法是用来设置目标对象的图像资源
            public void imageLoaded(Drawable imageDrawable);
        }
    }
```

说明：

final 参数是指当函数参数为 final 类型时，你可以读取使用该参数，但是无法改变该参数的值。参看：Java 关键字 final、static 使用总结

这里使用 SoftReference 是为了解决内存不足的错误（OutOfMemoryError）的，更详细的可以参看：内存优化的两个类:SoftReference 和 WeakReference

前段调用:

```
package ghj1976.AndroidTest;

import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;

import android.widget.ImageView;

public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        loadImage4("http://www.baidu.com/img/baidu_logo.gif", R.id.imageView1);
        loadImage4("http://www.chinatelecom.com.cn/images/logo_new.gif",
                R.id.imageView2);
        loadImage4("http://cache.soso.com/30d/img/web/logo.gif",
                R.id.imageView3);
        loadImage4("http://csdnimg.cn/www/images/csdnindex_logo.gif",
                R.id.imageView4);
        loadImage4("http://images.cnblogs.com/logo_small.gif",
                R.id.imageView5);
    }

    private AsyncImageLoader3 asyncImageLoader3 = new AsyncImageLoader3();

    // 引入线程池，并引入内存缓存功能,并对外部调用封装了接口，简化调用过程
    private void loadImage4(final String url, final int id) {
        // 如果缓存过就会从缓存中取出图像，ImageCallback 接口中方法也不会被执行
        Drawable cacheImage = asyncImageLoader3.loadDrawable(url,
                new AsyncImageLoader3.ImageCallback() {
                    // 请参见实现：如果第一次加载 url 时下面方法会执行
                    public void imageLoaded(Drawable imageDrawable) {
                        ((ImageView) findViewById(id))
                                .setImageDrawable(imageDrawable);
                    }
                });
        if (cacheImage != null) {
            ((ImageView) findViewById(id)).setImageDrawable(cacheImage);
        }
    }
```

}

25、Bitmap 操作

需求：从服务器下载一张图片，显示在 ImageView 控件上，并将该图片保存在移动设备的 SD 上。
步骤：

获得输入流

```
        //urlPath:服务器路径;
    public InputStream getUrlInputStream(String urlPath) throws IOException{
        URL url=new URL(urlPath);
        HttpURLConnection conn=(HttpURLConnection) url.openConnection();
        InputStream in=conn.getInputStream();
        if(in!=null){
            return in;
        }else{
            Log.i("test", "输入流对象为空");
            return null;
        }
    }
```

将输入流转化为 Bitmap 流

```
public Bitmap getMyBitmap(InputStream in){
        Bitmap bitmap=null;
        if(in!=null){
            bitmap=BitmapFactory.decodeStream(in);
            //BitmapFactory 的作用：create Bitmap objects from various sources,including files,streams and byte-arrays;
            return bitmap;
        }else{
            Log.i("test", "输入流对象 in 为空");
            return null;
        }
    }
```

给 ImageView 对象赋值

```
public void setWidgetImage(Bitmap bitmap){
```

```
                ImageView img=new ImageView(this);
                if(bitmap!=null){
                        img.setImageBitmap(bitmap);
                }
        }
```

获取 SD 卡上的文件存储路径

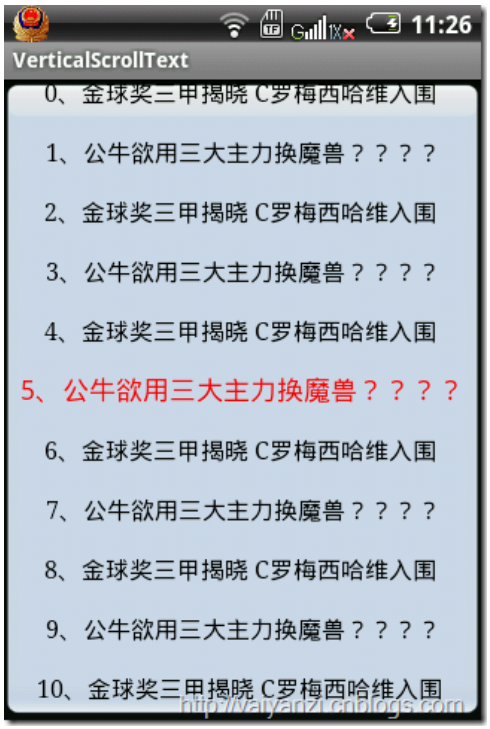
```
 public void createSDFile(){
                File sdroot=Environment.getExternalStorageDirectory();
                File file=new File(sdroot+"/Android/date/包名/文件名");

if(Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState())){
                        /**
                         * 相关操作
                         */
                }
        }
```

注意:SD 卡的权限

将图片保存到 SD 卡上

```
 public boolean readToSDCard(File file,Bitmap bitmap) throws FileNotFoundException{
                FileOutputStream os=new FileOutputStream(file);
                return bitmap.compress(Bitmap.CompressFormat.PNG, 90, os);
                //bitmap.compress()的作用:write a compressed version of the bitmap to the specified outputstream;
                //true:表示操作成功，false:表示操作失败
        }

OutputStream os = new FileOutputStream(fileToWriteTo);


BMPEncodeParam param = new BMPEncodeParam();


ImageEncoder enc = ImageCodec.createImageEncoder("BMP", os, param);


enc.encode(img);


 os.close();
```

26、TextView 垂直滚动



实现图中效果的关键点是：

1、重写 onDrow 方法，计算每次的滚动的距离。

2、计算 view 的 Y 轴的重点，让当前显示的处于高亮显示状态。

3、定时的刷新 View 使其界面不断的刷先，出现滚动的效果。

4、实现数据结构，将数据传给 view。

下面看看主要代码：

1、创建一个类继承 TextView

```
/**
* @author xushilin
*
*  垂直滚动的 TextView Widget
*/
public class VerticalScrollTextView extends TextView
```

2、实现构造函数：

```
public VerticalScrollTextView(Context context) {
                super(context);
                init();
        }
        public VerticalScrollTextView(Context context, AttributeSet attr) {
                super(context, attr);
                init();
        }
        public VerticalScrollTextView(Context context, AttributeSet attr, int i) {
                super(context, attr, i);
                init();
        }
        private void init() {
                setFocusable(true);
                //这里主要处理如果没有传入内容显示的默认值
                if(list==null){
                        list=new ArrayList<Notice>();
                        Notice sen=new Notice(0,"暂时没有通知公告");
                        list.add(0, sen);
                }
                //普通文字的字号，以及画笔颜色的设置
                mPaint = new Paint();
                mPaint.setAntiAlias(true);
                mPaint.setTextSize(16);
                mPaint.setColor(Color.BLACK);
                mPaint.setTypeface(Typeface.SERIF);
                //高亮文字的字号，以及画笔颜色的设置
                mPathPaint = new Paint();
                mPathPaint.setAntiAlias(true);
                mPathPaint.setColor(Color.RED);
                mPathPaint.setTextSize(16);
                mPathPaint.setTypeface(Typeface.SANS_SERIF);
        }
```

3、从写 onDraw 方法，并计算文字的行距，并且将将普通文字和高亮文字，在这个方法中绘制出来

```
protected void onDraw(Canvas canvas) {
                super.onDraw(canvas);
                canvas.drawColor(0xEFeffff);
```

```
                Paint p = mPaint;
                Paint p2 = mPathPaint;
                p.setTextAlign(Paint.Align.CENTER);
                if (index == -1)
                        return;
                p2.setTextAlign(Paint.Align.CENTER);

                canvas.drawText(list.get(index).getName(), mX, middleY, p2);
                float tempY = middleY;

                for (int i = index - 1; i >= 0; i--) {
                        tempY = tempY - DY;
                        if (tempY < 0) {
                                break;
                        }
                        canvas.drawText(list.get(i).getName(), mX, tempY, p);
                }
                tempY = middleY;

                for (int i = index + 1; i < list.size(); i++) {
                        tempY = tempY + DY;
                        if (tempY > mY) {
                                break;
                        }
                        canvas.drawText(list.get(i).getName(), mX, tempY, p);
                }
        }
```

4、计算 Y 轴中值以及更新索引

```
protected void onSizeChanged(int w, int h, int ow, int oh) {
                super.onSizeChanged(w, h, ow, oh);
                mX = w * 0.5f;
                mY = h;
                middleY = h * 0.5f;
        }

        private long updateIndex(int index) {
                if (index == -1)
                        return -1;
                this.index=index;
                return index;
        }
```

5、定时更新 view，并将接口暴露给客户程序调用。

```
public void updateUI(){
		new Thread(new updateThread()).start();
	}

	class updateThread implements Runnable {
		long time = 1000;
		int i=0;
		public void run() {
			while (true) {
				long sleeptime = updateIndex(i);
				time += sleeptime;
				mHandler.post(mUpdateResults);
				if (sleeptime == -1)
					return;
				try {
					Thread.sleep(time);
					i++;
					if(i==getList().size())
						i=0;
				} catch (InterruptedException e) {
					e.printStackTrace();
				}
			}
		}
	}
	Handler mHandler = new Handler();
	Runnable mUpdateResults = new Runnable() {
		public void run() {
			invalidate();
		}
	};
```

6、xml 布局文件中调用：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Demonstrates scrolling with a ScrollView. -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
```

```
        <com.demo.xsl.text.SampleView
            android:id="@+id/sampleView1"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:background="@drawable/selector"
           />

</LinearLayout>
```

７、java 代码中调用，传递数据：

```
package com.demo.xsl.text;

import java.util.ArrayList;
import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;

public class VerticalScrollTextActivity extends Activity {


        SampleView mSampleView;
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                mSampleView = (SampleView) findViewById(R.id.sampleView1);
                List lst=new ArrayList<Sentence>();
                for(int i=0;i<30;i++){
                        if(i%2==0){
                                Sentence sen=new Sentence(i,i+"、金球奖三甲揭晓 C
罗梅西哈维入围 ");
                                lst.add(i, sen);
                        }else{
                                Sentence sen=new Sentence(i,i+"、公牛欲用三大主力换
魔兽？？？？");
                                lst.add(i, sen);
                        }
                }
                //给 View 传递数据
                mSampleView.setList(lst);
                //更新 View
                mSampleView.updateUI();
```

```
        }
    }
```

27、判断某服务是否开启

```
    /**
     * 通过Service的类名来判断是否启动某个服务
     *
     * @param mServiceList
     * @param className
     * @return
     */
    public static boolean serviceIsRunning (Context context, String
serviceName) {

        ActivityManager activityManager = (ActivityManager)
context.getSystemService(Context.ACTIVITY_SERVICE);
        List<ActivityManager.RunningServiceInfo> mServiceList =
activityManager.getRunningServices(100);

        for (int i = 0; i < mServiceList.size(); i++) {
            if
(serviceName.equals(mServiceList.get(i).service.getClassName())) {
                return true;
            }
        }
        return false;
    }
```

28、判断 SD 卡是否已挂载

```
    /**
     * 判断SDcard是否挂载
     */
    public static boolean sdCardCanUsed() {
        return
android.os.Environment.getExternalStorageState().equals(
                android.os.Environment.MEDIA_MOUNTED);
    }
```

29、文件操作类

1、获得文件或目录的大小

```
/**
 * 递归获得文件或目录的大小
 * @param dir
 * @return
 */
public static long getFileSize(File dir) {

    if( !dir.isDirectory() ) {
        size = size + dir.length();
        Log.e("", dir.toString());
    } else {
        for(File file : dir.listFiles()) {
            if(!file.isDirectory()) {
                Log.e("", file.toString());
                size = size + file.length();
            } else {
                getFileSize( file );//递归
            }
        }
    }

    return size;
}
```

2、递归删除目录或文件

```
/**
 * 删除一个文件或目录（可以是非空目录）
 *
 * @param dir
 * @return
 */
public static boolean delDir(File dir) {
    if (dir == null || !dir.exists()) {
        return false;
    }
    if (dir.isFile() || dir.listFiles() == null) {
        dir.delete();
```

```
            return true;
        } else {
            for (File file : dir.listFiles()) {
                if (file.isFile()) {
                    file.delete();
                } else if (file.isDirectory()) {
                    delDir(file);// 递归
                }
            }
            dir.delete();
        }

        return true;
    }
```

3、

30、手动更新所有 Widget

```
    public static void updateAllWidgets(Context context) {
        AppWidgetManager gm = AppWidgetManager.getInstance(context);
        int[] appWidgetIds = gm.getAppWidgetIds(new
ComponentName(context, MyWidgetProvider.class));//注：MyWidgetProvider
是你自己写的继承自AppWidgetProvider的类
        // 更新所有AppWidget
        if (appWidgetIds != null) {
            final int N = appWidgetIds.length;
            for (int i = 0; i < N; i++) {
                DelWidgetProvider.updateAppWidget(context, gm,
appWidgetIds[i]);
            }
        }
    }
```

31、有关 ListView 问题

1、点击、长按、弹出菜单的消息映射

2、注意问题

在长按响应中，若是 return false 结束的话，则执行完长按消息后，紧接着执行点击响应。而 return true 结束，则执行完长按响应后停止响应。

32、旋转屏幕对话框消失的问题

在 androidmanifest 中的 activity 中加入 ：
android:configChanges="keyboardHidden|orientation"

在 Activity 中重写此方法：

```
    // 重写横竖屏切换时加载项方法，防止对话框消失
    public void onConfigurationChanged(Configuration newConfig) {
        try {
            super.onConfigurationChanged(newConfig);
            if  (this.getResources().getConfiguration().orientation  ==
Configuration.ORIENTATION_LANDSCAPE) {
                // land

            }                           else                           if
(this.getResources().getConfiguration().orientation                 ==
Configuration.ORIENTATION_PORTRAIT) {
                // port

            }
        } catch (Exception ex) {
        }

    }
```

32、在手机上打开文件的方法

```
private void openFile(File f) {
    Intent intent = new Intent();
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.setAction(android.content.Intent.ACTION_VIEW);
 startActivity(intent);
 }
```

33、使用系统自带的 TabHost 的问题

Your content must have a TabHost whose id attribute is 'android.R.id.tabhost'问题的解决办法 2011-11-19 22:02 问题 1. 运行 Activity 的时候出现 Your content must have a TabHost whose id attribute is 'android.R.id.tabhost'

添加 Layout 的时候，xml 跟元素选择 TabHost， 但是 ADT 没有添加 id 属性， 运行的时候，会提示 Your content must have a TabHost whose id attribute is 'android.R.id.tabhost'错误， 需要添加 android:id="@android:id/tabhost"， 这样就可以了。

问题 2. 运行 Activity 的时候出现 Your TabHost must have a TabWidget whose id attribute is 'android.R.id.tabcontent'

解决方法： 修改 FrameLayout 添加 id 属性， ADT 自动生成的 xml 文件中 Id 是 android:id="@+id/FrameLayout01 " ， 需要修改成下面的格式 android:id="@android:id/tabcontent "，这个估计会困扰一大批初学者，谁会想到会修改这个地方，看到错误很容易修改成 tabcontent，但是前缀不容易想到。 而且在 ADT 可视化编辑这个文件的时候， 界面上显示 NullPointerException，这个是 ADT 的一个 BUG。

修改后的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost
   android:id="@android:id/tabhost"
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent" android:layout_height="fill_parent">
      <LinearLayout android:id="@+id/LinearLayout01" android:layout_height="fill_parent" android:layout_width="fill_parent" android:orientation="vertical">

            <TabWidget android:id="@android:id/tabs" android:layout_height="wrap_content" android:layout_width="fill_parent">
            </TabWidget>
            <FrameLayout android:id="@android:id/tabcontent" android:layout_width="fill_parent" android:layout_height="fill_parent">
            </FrameLayout>
</LinearLayout>
</TabHost>
```

注意： 如果用 TabHost 的话， 上面标红的三处必须是一样， 这个是 Google 的约定。 而且一个工程中只能有一个 TabHost。

(3) 可以删除自己配置文件，不用指定配置文件。

34、弹出菜单

效果如图：

代码如下：

```
/**
   *  构造菜单 Adapter
   *
   * @param menuNameArray
   *                  名称
   * @param imageResourceArray
   *                  图片
   * @return SimpleAdapter
   */
  private SimpleAdapter getMenuAdapter(String[] menuNameArray,
      int[] imageResourceArray) {
    ArrayList<HashMap<String, Object>> data = new ArrayList<HashMap<String, Object>>();
    for (int i = 0; i < menuNameArray.length; i++) {
      HashMap<String, Object> map = new HashMap<String, Object>();
      map.put("itemImage", imageResourceArray[i]);
      map.put("itemText", menuNameArray[i]);
      data.add(map);
    }
    SimpleAdapter simperAdapter = new SimpleAdapter(this, data,
        R.layout.item_menu, new String[] { "itemImage", "itemText" },
```

```
        new int[] { R.id.item_image, R.id.item_text });
    return simperAdapter;
  }
```

35、Toast 重叠显示时延迟解决

可能大部分童鞋在用 Toast 的时候都是直接一行代码搞定吧？就像下面的代码段：

Toast.makeText(this,”这样有延时啊!!!”,Toast.LENGTH_SHORT).show();

但是这样写得话就是如果在 Toast 还没有消失的时候又点击显示 Toast，则显示的 Toast 会先将之前显示的消失掉后才能显示后面的，所以有延时啊！

分开写就不会有延迟了：

```
Toast toast ;
if(toast == null){
    toast = Toast.makeText(this,"",Toast.LENGTH_SHORT) ;
}
toast.setText("这样木有延时！！！") ;
toast.show() ;
```

来自：无延时显示Toast的方法 http://www.apkstory.com/development/toast-show-no-delay.html | APK故事汇

36、ADT 新特性：ImageView 的定义

今天使用了下 ADT 16.0 在定义一个 ImageVIew 的时候 总是提示这个[Accessibility] Missing contentDescription attribute on image 警告，虽说可以不理 但总是感觉怪怪的，在网上一搜 发现原来这是 ADT 16.0 的新特性，在一些没有文本显示的控件里，如 imageView 和 imageButton 等，ADT 会提示你定义一个 android:contentDescription 属性，用来描述这个控件的作用。英文原文如下，如有翻译的不对的地方，敬请批评指正。

Resolved this warning by setting attribute android:contentDescription for my ImageView

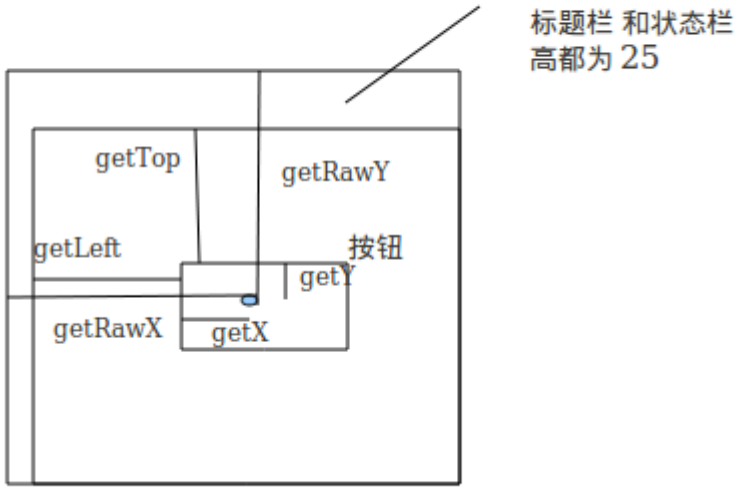
android:contentDescription="@string/desc"

Android Lint support in ADT 16 throws this warning to ensure that image widgets provide a contentDescription

This defines text that briefly describes content of the view. This property is used primarily for accessibility. Since some views do not have textual representation this attribute can be used for providing such.

Non-textual widgets like ImageViews and ImageButtons should use the contentDescription attribute to specify a textual description of the widget such that screen readers and other accessibility tools can adequately describe the user interface.

37、MotionEvent 中获取坐标的问题



getRowX：触摸点相对于屏幕的坐标
getX： 触摸点相对于按钮的坐标
getTop： 按钮左上角相对于父 view（LinerLayout）的 y 坐标
getLeft： 按钮左上角相对于父 view（LinerLayout）的 x 坐标

可以想象 getRight()等同于下面的计算：getLeft()+getWidth()。

38、添加多个 Widget 样式

思想：添加多个接收器，为每一个接收机分配一个 Widget 样式，这样就会有多个可选 widget 样式了。

1、添加多个接收器

```
<receiver android:name="com.jftt.widget.MyWidgetProvider1" android:label="天气  4 x 1">
    <meta-data android:name="android.appwidget.provider"
            android:resource="@xml/my_widget1" />
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
```

```
        </intent-filter>
    </receiver>
    <receiver android:name="com.jftt.widget.MyWidgetProvider" android:label="全部  4 x 2">
        <meta-data android:name="android.appwidget.provider"
                android:resource="@xml/my_widget" />
        <intent-filter>
            <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
        </intent-filter>
    </receiver>
    <receiver android:name="com.jftt.widget.MyWidgetProvider2" android:label="步数  4 x 1">
        <meta-data android:name="android.appwidget.provider"
                android:resource="@xml/my_widget2" />
        <intent-filter>
            <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
        </intent-filter>
    </receiver>
```

2、CreateWidget.java 代码

```
package com.jftt.activity;

import android.app.Activity;
import android.appwidget.AppWidgetManager;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;

import com.jftt.widget.R;

public class CreateWidget extends Activity {
    private static final String TAG = "CreateWidget";
    int mAppWidgetId;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i(TAG, " on WidgetConf ... ");

        setResult(RESULT_CANCELED);
        // Find the widget id from the intent.
        Intent intent = getIntent();
        Bundle extras = intent.getExtras();
        if (extras != null) {
            mAppWidgetId   =   extras.getInt(AppWidgetManager.EXTRA_APPWIDGET_ID,
```

```
AppWidgetManager.INVALID_APPWIDGET_ID);
            }

            // If they gave us an intent without the widget id, just bail.
            if (mAppWidgetId == AppWidgetManager.INVALID_APPWIDGET_ID) {
                finish();
            }

            // return OK
            Intent resultValue = new Intent();
            resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, mAppWidgetId);

            setResult(RESULT_OK, resultValue);
            finish();
        }
}
```

3、MyWidgetProvider.java

```
package com.jftt.widget;

import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.util.Log;
import android.view.View;
import android.widget.RemoteViews;

import com.jftt.activity.WidgetSetting;

public class MyWidgetProvider extends AppWidgetProvider {
    private static final String TAG = "MyWidgetProvider";

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[]
appWidgetIds) {
            super.onUpdate(context, appWidgetManager, appWidgetIds);
            Log.i(TAG, "onUpdate");
            final int N = appWidgetIds.length;
            for (int i = 0; i < N; i++) {
                int appWidgetId = appWidgetIds[i];
```

```
            Log.i(TAG, "this is [" + appWidgetId + "] onUpdate!");

            Intent intent = new Intent(context, WidgetSetting.class);
            PendingIntent pending = PendingIntent.getActivity(context, 0, intent, 0);

            Intent intent2 = new Intent(context, MyWidgetProvider.class);
            intent2.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId);

            Log.i(TAG, "appWidgetId "+appWidgetId);
            PendingIntent doubleClick = PendingIntent.getBroadcast(context, appWidgetId,
intent2, PendingIntent.FLAG_UPDATE_CURRENT);

            RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.widget);
            remoteViews.setOnClickPendingIntent(R.id.pic5, pending);
            remoteViews.setOnClickPendingIntent(R.id.widget, doubleClick);
            appWidgetManager.updateAppWidget(appWidgetId, remoteViews);
        }
    }

    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        super.onDeleted(context, appWidgetIds);
        Log.i(TAG, "onDeleted");
        final int N = appWidgetIds.length;
        for (int i = 0; i < N; i++) {
            int appWidgetId = appWidgetIds[i];
            Log.i(TAG, "this is [" + appWidgetId + "] onDelete!");
        }
    }

    @Override
    public void onDisabled(Context context) {
        super.onDisabled(context);
        Log.i(TAG, "onDisabled");
        PackageManager pm = context.getPackageManager();
        pm.setComponentEnabledSetting(new ComponentName("com.jftt.widget",
"com.jftt.widget.MyWidgetProvider"),
PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
PackageManager.DONT_KILL_APP);
    }

    @Override
    public void onEnabled(Context context) {
```

```
        super.onEnabled(context);
        Log.i(TAG, "onEnabled");
        PackageManager pm = context.getPackageManager();
        pm.setComponentEnabledSetting(new ComponentName("com.jftt.widget",
"com.jftt.widget.MyWidgetProvider"),
PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
PackageManager.DONT_KILL_APP);
    }

    static long start = 0;

    @Override
    public void onReceive(Context context, Intent intent) {
        super.onReceive(context, intent);
        if ((System.currentTimeMillis() - start) < 500) {
            AppWidgetManager appWidgetManager =
AppWidgetManager.getInstance(context);
            RemoteViews remoteViews = new RemoteViews(context.getPackageName(),
R.layout.widget);
            remoteViews.setViewVisibility(R.id.setting, View.VISIBLE);
            int appWidgetId =
intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, 0);
            appWidgetManager.updateAppWidget(appWidgetId, remoteViews);
        } else {
            start = System.currentTimeMillis();
        }
    }
}
```

39、为 Activity 添加快捷方式

1、实现代码：

```
public static void createShortCut(Context context) {
    final Intent addIntent = new Intent(
            "com.android.launcher.action.INSTALL_SHORTCUT");
    final Parcelable icon = Intent.ShortcutIconResource.fromContext(
            context, R.drawable.icon); // 获取快捷键的图标
    addIntent.putExtra("duplicate", false);
    final Intent myIntent = new Intent(context,XX.class);
    addIntent.putExtra(Intent.EXTRA_SHORTCUT_NAME,
            context.getString(R.string.mobile_site_shortcut_name));// 快捷方式的标题
    addIntent.putExtra(Intent.EXTRA_SHORTCUT_ICON_RESOURCE, icon);// 快捷方式的
图标
    addIntent.putExtra(Intent.EXTRA_SHORTCUT_INTENT, myIntent);// 快捷方式的动作
```

```
        context.sendBroadcast(addIntent);
}
```
2、添加权限
```
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT" />
```
3、给指定的 Activity 创建快捷方式，添加过滤器
```
<intent-filter>
        <action android:name="android.intent.action.CREATE_SHORTCUT" />
</intent-filter>
```

40、点击 widget 获取 ID

在更新 Widget 时，我们常常要绑定一些响应事件，比如绑定到 Activity、Service 等；存在多个 Widget 时，如果我们需要获得在点击的 Widget 的 ID，那么为了实现此功能，我们可以使用 Bundle 传值的方法，使与之绑定的 Activity 或 Service 获得其 ID；
举例代码如下：
在 WidgetProvider 中我们可以重写 onUpdate()方法，并在其中添加绑定操作：

```
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager,
            int[] appWidgetIds) {
        final int N = appWidgetIds.length;
        for (int i = 0; i < N; i++) {
            int appWidgetId = appWidgetIds[i];
            RemoteViews views = new RemoteViews(context.getPackageName(),
                        R.layout.layout_widget);//此为 widget 的布局文件
            views.setTextViewText(R.id.text_widget, title);//布局文件中的一个 TextView
            Intent intent = new Intent(context, ExActivity.class);
            Bundle bundle = new Bundle();
            bundle.putInt(TakePicOpers.TAG_WIDGETID, appWidgetId);
            intent.putExtras(bundle);
            //此处切记！ 不要把 requestCode 设置为一个定值，否则，bundle 传送的值总是最后添加的 widget 的 ID
            PendingIntent pendingIntent = PendingIntent.getActivity(context, appWidgetId,
                        intent, PendingIntent.FLAG_UPDATE_CURRENT);
            views.setOnClickPendingIntent(R.id.img_widget, pendingIntent); //绑定到该布局中的 ImageView 上
            appWidgetManager.updateAppWidget(appWidgetId, views);
        }
}
```

另外也可以自己手动添加一个更新方法，来实现不同功能。

41、ViewFlipper 小动画

如果实现某个或某几个小部件循环不停的做同样的运动，那么这个 ViewFlipper 将很方便，实现如下：

（此例为一个小图片做上上下下的运动）

1、开启动画的 Java 代码：

```
ViewFlipper flipper = ((ViewFlipper) findViewById(R.id.flipper));
flipper.startFlipping();
flipper.setInAnimation(AnimationUtils.loadAnimation(this, R.anim. anim_out));
flipper.setOutAnimation(AnimationUtils.loadAnimation(this, R.anim. anim_alpha));
```

2、ViewFlipper 布局文件：

```
<ViewFlipper android:id="@+id/flipper"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:flipInterval="3000"><!—此为循环间隔时间--!
        <ImageView
            android:id="@+id/img_donate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/kkk"/>
        <!—还可以继续添加更多部件--!>
</ViewFlipper>
```

3、动画效果文件：

1、anim_out.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromYDelta="0"
        android:toYDelta="100%p"
        android:duration="300"/>
</set>
```

2、anim_alpha.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<alpha
android:fromAlpha="0.1"
android:toAlpha="1.0"
android:duration="2500"
/>
</set>
```

42、setTextColor 的问题

若直接这么设置如：
text_sound.setTextColor(R.color.color_red);
是无效的，应该这么设置：
方法 1:通过 rgb 值的方式
tv.setTextColor(Color.rgb(255, 255, 255));
方案二:通过资源文件

```
Resources resource = (Resources) getBaseContext().getResources();
ColorStateList csl = (ColorStateList) resource.getColorStateList(R.color.my_color);
if (csl != null) {
        tv.setTextColor(csl);
}
```

43、获取程序信息并 kill

```
package com.pig.message.info;

import java.util.List;

import android.app.ActivityManager;
import android.app.ActivityManager.MemoryInfo;
import android.app.ActivityManager.RunningAppProcessInfo;
import android.app.ActivityManager.RunningServiceInfo;
import android.app.ActivityManager.RunningTaskInfo;
import android.content.Context;
import android.os.Debug;

public class AppInformation {
  private ActivityManager activityManager = null;

  public AppInformation(Context context){
    this.activityManager                                                    =
(ActivityManager)context.getSystemService(Context.ACTIVITY_SERVICE);
  }

  public boolean isAppRunning(String packagename){
    System.out.println("appIsRunning()...");

    return false;
  }

  public int killProcessByName(String killProcessName){
```

```
    System.out.println("killProcessByName()...");

    int killNum = 0;
    List appProcessList = activityManager.getRunningAppProcesses();

    for(int i=0; i<appProcessList.size(); i++){
      RunningAppProcessInfo appProcessInfo = (RunningAppProcessInfo) appProcessList.get(i);
      //进程 ID
      int pid = appProcessInfo.pid;
      //用户 ID，类似于 Linux 的权限不同，ID 也就不同， 比如 root
      int uid = appProcessInfo.uid;
      //进程名，默认是包名或者由属性 android:process=""指定
      String processName = appProcessInfo.processName;
      //获得该进程占用的内存
      int[] memPid = new int[]{ pid };
      //此 MemoryInfo 位于 android.os.Debug.MemoryInfo 包中，用来统计进程的内存信息
      Debug.MemoryInfo[] memoryInfo = activityManager.getProcessMemoryInfo(memPid);
      //获取进程占内存用信息 kb 单位
      int memSize = memoryInfo[0].dalvikPrivateDirty;

      System.out.println("process name: " + processName + " pid: " + pid + " uid: " + uid + "
memory size is -->" + memSize + "kb");

      //获得每个进程里运行的应用程序(包)，即每个应用程序的包名
      String[] packageList = appProcessInfo.pkgList;
      for(String pkg : packageList){
        System.out.println("package name " + pkg + " in process id is -->" + pid);
      }

      if(killProcessName.equals(processName)){
        System.out.println("================killProcess pid-->" + pid);
        android.os.Process.killProcess(pid);
        killNum++;
      }
    }
    return killNum;
  }

  public long getSystemAvaialbeMemorySize(){
    MemoryInfo memoryInfo = new MemoryInfo();
    activityManager.getMemoryInfo(memoryInfo);
    long memSize = memoryInfo.availMem;

    System.out.println("getSystemAvaialbeMemorySize()...memory size: " + memSize);
```

```
    return memSize;

    //调用系统函数，字符串转换 long -String KB/MB
    //return Formatter.formatFileSize(context, memSize);
  }

  public List getRunningAppProcessInfo(){
    System.out.println("getRunningAppProcessInfo()...");

    List appProcessList = activityManager.getRunningAppProcesses();

    for(int i=0; i<appProcessList.size(); i++){
      RunningAppProcessInfo appProcessInfo = (RunningAppProcessInfo) appProcessList.get(i);
      //进程 ID
      int pid = appProcessInfo.pid;
      //用户 ID，类似于 Linux 的权限不同，ID 也就不同， 比如 root
      int uid = appProcessInfo.uid;
      //进程名，默认是包名或者由属性 android:process=""指定
      String processName = appProcessInfo.processName;
      //获得该进程占用的内存
      int[] memPid = new int[]{ pid };
      //此 MemoryInfo 位于 android.os.Debug.MemoryInfo 包中，用来统计进程的内存信息
      Debug.MemoryInfo[] memoryInfo = activityManager.getProcessMemoryInfo(memPid);
      //获取进程占内存用信息 kb 单位
      int memSize = memoryInfo[0].dalvikPrivateDirty;

      System.out.println("process name: " + processName + " pid: " + pid + " uid: " + uid + "
memory size is -->" + memSize + "kb");

      //获得每个进程里运行的应用程序(包)，即每个应用程序的包名
      String[] packageList = appProcessInfo.pkgList;
      for(String pkg : packageList){
        System.out.println("package name " + pkg + " in process id is -->" + pid);
      }

    }

    return appProcessList;
  }

  public List getRunningServiceInfo(){
    System.out.println("getRunningServiceInfo()...");
```

```
  List serviceList = activityManager.getRunningServices(30);

  for(int i=0; i<serviceList.size(); i++){
    RunningServiceInfo serviceInfo = (RunningServiceInfo) serviceList.get(i);
    //进程 ID
    int pid = serviceInfo.pid;
    //用户 ID，类似于 Linux 的权限不同，ID 也就不同， 比如 root
    int uid = serviceInfo.uid;
    //进程名，默认是包名或者由属性 android:process=""指定
    String processName = serviceInfo.process;

    String serviceStr = serviceInfo.toString();
    System.out.println("serviceStr: " + serviceStr);

    //获得该进程占用的内存
    int[] memPid = new int[]{ pid };
    //此 MemoryInfo 位于 android.os.Debug.MemoryInfo 包中，用来统计进程的内存信息
    Debug.MemoryInfo[] memoryInfo = activityManager.getProcessMemoryInfo(memPid);
    //获取进程占内存用信息 kb 单位
    int memSize = memoryInfo[0].dalvikPrivateDirty;

    System.out.println("The name of the process this service runs in: " + processName + " pid: " +
pid + " uid: " + uid + " memory size is -->" + memSize + "kb");

  }

  return serviceList;
 }

 public List getRunningTaskInfo(){
  System.out.println("getRunningServiceInfo()...");

  List taskList = activityManager.getRunningTasks(30);

  for(int i=0; i<taskList.size(); i++){
    RunningTaskInfo taskInfo = (RunningTaskInfo) taskList.get(i);
    String packageName = taskInfo.baseActivity.getPackageName();

    System.out.println("package name: " + packageName);

  }

  return taskList;
 }
```

}

44、mediaPlayer 与 soundPool

前者用于播放一般音频；后者更适用于播放短促的提示音！
具体原因：
在 Android 开发中我们经常使用 MediaPlayer 来播放音频文件，但是 MediaPlayer 存在一些不足，例如：资源占用量较高、延迟时间较长、不支持多个音频同时播放等。这些缺点决定了 MediaPlayer 在某些场合的使用情况不会很理想，例如在对时间精准度要求相对较高的游戏开发中。

在游戏开发中我们经常需要播放一些游戏音效（比如：子弹爆炸，物体撞击等），这些音效的共同特点是短促、密集、延迟程度小。在这样的场景下，我们可以使用 SoundPool 代替 MediaPlayer 来播放这些音效。

SoundPool（android.media.SoundPool），顾名思义是声音池的意思，主要用于播放一些较短的声音片段，支持从程序的资源或文件系统加载。与 MediaPlayer 相比，SoundPool 的优势在于 CPU 资源占用量低和反应延迟小。另外，SoundPool 还支持自行设置声音的品质、音量、播放比率等参数，支持通过 ID 对多个音频流进行管理。

SoundPool 存在的缺陷

1.SoundPool 最大只能申请 1M 的内存空间，这就意味着我们只能用一些很短的声音片段，而不是用它来播放歌曲或者做游戏背景音乐。

2.SoundPool 提供了 pause 和 stop 方法，但这些方法建议最好不要轻易使用，因为有些时候它们可能会使你的程序莫名其妙的终止。建议使用这两个方法的时候尽可能多做测试工作，还有些朋友反映它们不会立即中止播放声音，而是把缓冲区里的数据播放完才会停下来，也许会多播放一秒钟。

3.SoundPool 的效率问题。其实 SoundPool 的效率在这些播放类中算是很好的了，但是有的朋友在 G1 中测试它还是有 100ms 左右的延迟，这可能会影响用户体验。也许这不能管 SoundPool 本身，因为到了性能比较好的 Droid 中这个延迟就可以让人接受了。

在现阶段 SoundPool 有这些缺陷，但也有着它不可替代的优点，基于这些我们建议大在如下情况中多使用 SoundPool：1.应用程序中的声效(按键提示音，消息等)2.游戏中密集而短暂的声音(如多个飞船同时爆炸)

1、 MediaPlayer

一般用法：

```
MediaPlayer mediaPlayer = new MediaPlayer();
if (mediaPlayer.isPlaying()) {
    mediaPlayer.reset();//重置为初始状态
```

```
}
mediaPlayer.setDataSource("/mnt/sdcard/god.mp3");
mediaPlayer.prepare();//缓冲
mediaPlayer.start();//开始或恢复播放
mediaPlayer.pause();//暂停播放
mediaPlayer.start();//恢复播放
mediaPlayer.stop();//停止播放
mediaPlayer.release();//释放资源
mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {//播出完毕事件
            @Override public void onCompletion(MediaPlayer arg0) {
        mediaPlayer.release();
            }
});
mediaPlayer.setOnErrorListener(new MediaPlayer.OnErrorListener() {//  错误处理事件
             @Override public boolean onError(MediaPlayer player, int arg1, int arg2) {
   mediaPlayer.release();
   return false;
             }
});
```

2、 SoundPool

一般用法：

```
public class AudioActivity extends Activity     {
      private SoundPool pool;
      @Override
      public void onCreate(Bundle savedInstanceState)     {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            //指定声音池的最大音频流数目为 10，声音品质为 5
            pool = new SoundPool(10, AudioManager.STREAM_SYSTEM, 5);
            //载入音频流，返回在池中的 id
            final int sourceid = pool.load(this, R.raw.pj, 0);
            Button button = (Button)this.findViewById(R.id.button);
            button.setOnClickListener(new View.OnClickListener()     {
                  public void onClick(View v)     {
//播放音频，第二个参数为左声道音量;第三个参数为右声道音量;第四个参数为优先级；第五个参数为循环次数，0 不循环，-1 循环;第六个参数为速率，速率       最低 0.5 最高为 2，1 代表正常速度
                        pool.play(sourceid, 1, 1, 0, -1, 1);
                  }
            });
      }
}
```

45、标题栏添加图标

方法 1:java 代码实现

```
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    requestWindowFeature(Window.FEATURE_LEFT_ICON); //申请设置个性化小图标
    setContentView(R.layout.custom_dialog_activity);
    //以下是 设置小图标
    Window win = getWindow();
    win. setFeatureDrawableResource(Window.FEATURE_LEFT_ICON, R.drawable.icon);
} //注意：requestFeature... 必须位于 setContentView...之前
```

方法 2: 为应用加上图标。

```
<application android:name="ApiDemosApplication"
android:label="@string/activity_sample_code"
android:icon="@drawable/app_sample_code" >
```

46、 URI

data 匹配规则
首先务必认识到，data 是一个相对复杂的要素。 data 由 URI 来描述和定位，URI 由三部分组成:
scheme://host:port/path 模式://主机:端口/路径

例如我们截获的挂载 SD 卡事件，它的 data 的 URI 是 file:///mnt/sdcard
其中模式部分是 file ， 主机:端口部分是空的， path 部分是 mnt/sdcard
此外在事件中，还可以设置 data 的 MIME 类型，作为事件的 datatype 属性。

首先明确一个 URI 的匹配原则：
部分匹配原则，即只要 filter 中声明的部分匹配成功，就认为整个 URI 匹配成功。

如：content://com.silenceburn.SdCardTester:1000/mydata/private/
若 filter 定义为 content://com.silenceburn.SdCardTester:1000/
则它们是可以匹配的。filter 中没有定义 path 部分，但是依然可以匹配成功，因为 filter 不声明的部分不进行比较。即无论 path 是什么，都可以匹配成功。

data 的 URI 的匹配规则如下：
1. 如果 data 的 URI 和 datatype 为空，则 filter 的 URI 和 type 也必须为空，才能匹配成功
2. 如果 data 的 URI 不为空，但是 datatype 为空，则 filter 必须定义 URI 并匹配成功，且 type 为空，才能匹配成功
3. 如果 data 的 URI 为空，但是 datatype 不为空，则 filter 必须 URI 为空，定义 type 并匹配成功，才能匹配成功

4. 如果 data 的 URI 和 data 都不为空，则 filter 的 URI 和 type 都必须定义并匹配成功，才能匹配成功。对于 URI 部分，有一个特殊处理，就是即使 filter 没有定义 URI，content 和 file 两种 URI 也作为既存的 URI 存在。（举个例子，对于 content 和 file 两种模式的 data，只要 filter 定义的 datatype 可以和事件匹配，就认为匹配成功，filter 不需要显式的增加 content 和 file 两种模式，这两种模式内置支持 ）

案例分析：SD 卡插拔事件的匹配

SD 卡插拔是一个隐式事件，而且 action 和 category 部分和我们的 filter 均可以匹配成功。其 data 部分的 URI 为 file:///mnt/sdcard ，datatype 为 null ，因此应用如下规则：
“如果 data 的 URI 不为空，但是 datatype 为空，则 filter 必须定义 URI 并匹配成功，且 type 为空，才能匹配成功”
我们的 filter 中没有使用 addtype 方法 ，因此 filter 的 type 为空， datatype 部分匹配成功。data 的 URI 为 file:///mnt/sdcard ，我们使用：
filter.addDataScheme("file");
定义 schema 为 file，根据部分匹配规则，data 匹配成功。
至此，整个事件匹配成功，因此必须添加 filter.addDataScheme("file"); 语句才能收到事件！

我们可以尝试把 filter.addDataScheme("file"); 后面增加语句
filter.addDataPath("mnt/sdcard", PatternMatcher.PATTERN_LITERAL);
依然可以匹配成功，收到 SD 卡插拔事件。因为这样就组成了一个 URI 的完全匹配。

我们可以尝试把给 filter 增加 datatype 属性，

```
try {
	filter.addDataType("text/*");
} catch (MalformedMimeTypeException e) {
	e.printStackTrace();
}
	try {
		filter.addDataType("text/*");
	} catch (MalformedMimeTypeException e) {
		e.printStackTrace();
	}
```

这样就无法匹配成功了。因为 SD 卡插拔事件的 datatype 是 null，
而我们定义的 filter 的 datatype 是 MIME"text/*"

47、BroadcastReceiver 旧事重提

系统接收器很常用，比如监听短信的收发、电话的呼入呼出，实现开机启动等。
要注意注册 Intent.actoin 方能接收到，如果是 java 代码实现的注册，记得在不需接收后取消注册，如果在 MainFest 文件中注册的话那就不必这么繁琐了。

话说若接收存储卡拔插的消息时，我们需要注意：
添加这么一句：
方法 1、在 Manifest 文件中添加：

```
<intent-filter>
    <action android:name="android.intent.action.MEDIA_MOUNTED"/>
    <action android:name="android.intent.action.MEDIA_EJECT" />
    <data android:scheme="file"/>
</intent-filter>
```

方法 2、若是用 java 代码注册的接收器，则如下：

```
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(Intent.ACTION_MEDIA_SCANNER_STARTED);
intentFilter.addAction(Intent.ACTION_MEDIA_SCANNER_FINISHED);
intentFilter.addDataScheme("file");
MyReceiver scanReceiver = new MyReceiver();//MyReceiver 是继承自 BroadcastReceiver 类的接收器
registerReceiver(scanReceiver, intentFilter);
//记得要在程序结束时取消注册：
unregisterReceiver(scanReceiver);
```

言归正传：
之所以重提 BroadcastReceiver 的原因是：
我们在程序中删除了存储卡中的文件——特别是媒体文件后，系统是不会自动更新媒体库的。
常见的问题是：我们在程序中删除了某张图片，再获取系统既存的媒体库时，该照片的记录仍然存在，但实际上它已经被删除了。
那么如何在删除媒体文件后让其自动更新媒体库，是我们要考虑的。
通常借助 BroadcastReceiver 可以方便的实现：

```
//扫描整个存储卡，更新媒体库。
String sdcardPath = Environment.getExternalStorageDirectory().toString();
sendBroadcast(new Intent(Intent.ACTION_MEDIA_MOUNTED, Uri.parse("file://" + sdcardPath)));
```

全盘扫描速度会比较慢，因此可以选择特定目录或文件进行扫描：

```
/这样只对刚刚删除或添加的图片进行扫描就可以了，
String filePath = Environment.getExternalStorageDirectory().toString() + “\craining.jpg”;
sendBroadcast(new    Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, "file://" + filePath)));
```

对于更新过程的显示我们可以接收此消息：

```
Intent.ACTION_MEDIA_SCANNER_STARTED//开开始扫描
Intent.ACTION_MEDIA_SCANNER_FINISHED//结束扫描
```

48、从 CalendarProvider 得到数据的方法：

```
ContentResolver contentResolver = context.getContentResolver();
Cursor cursor = contentResolver.query(Uri.parse("content://calendar/events"), null, null, null, null);
```

```
while(cursor.moveToNext()) {
    String eventTitle = cursor.getString(cursor.getColumnIndex("title"));
    Date eventStart = new Date(cursor.getLong(cursor.getColumnIndex("dtstart")));    // etc.
}
```

49、为程序添加一个“分享”发送到某个地方

50、屏幕关闭，不睡眠

```
WindowManager.LayoutParams params = getWindow().getAttributes();
params.flags |= WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON;
params.screenBrightness = 0;
getWindow().setAttributes(params);
```

51、Android 与 Linux 休眠

Linux 休眠过程：

1、冻结用户态进程和内核态任务

2、调用注册的设备的 suspend 的回调函数。顺序是按照注册顺序

3、休眠核心设备和使 CPU 进入休眠态，冻结进程是内核把进程列表中所有的进程的状态都设置为停止，并且保存下所有进程的上下文。 当这些进程被解冻的时候，他们是不知道自己被冻结过的，只是简单的继续执行。如何让 Linux 进入休眠呢？用户可以通过读写 sys 文件/sys /power/state 是实现控制系统进入休眠。.

```
# echo standby > /sys/power/state
```

命令系统进入休眠. 也可以使用

```
# cat /sys/power/state
```

来得到内核支持哪几种休眠方式

Linux Suspend 的流程相关的文件

你可以通过访问 Linux 内核网站 来得到源代码,下面是文件的路径：

- linux_soruce/kernel/power/main.c
- linux_source/kernel/arch/xxx/mach-xxx/pm.c
- linux_source/driver/base/power/main.c

接下来让我们详细的看一下 Linux 是怎么休眠/唤醒的

用户对于/sys/power/state 的读写会调用到 main.c 中的 state_store()，用户可以写入 const char * const pm_state[] 中定义的字符串， 比如"mem", "standby"。.然后 state_store()会调用 enter_state()，它首先会检查一些状态参数，然后同步文件系统。下面是代码：

准备，冻结进程

当进入到 suspend_prepare()中以后，它会给 suspend 分配一个虚拟终端来输出信 息，然后广播一个系统要进入 suspend 的 Notify，关闭掉用户态的 helper 进程，然 后一次调用

suspend_freeze_processes()冻结所有的进程，这里会保存所有进程 当前的状态，也许有一些进程会拒绝进入冻结状态，当有这样的进程存在的时候，会导致冻结失败,此函数就会放弃冻结进程,并且解冻刚才冻结的所有进程.

让外设进入休眠

现在，所有的进程(也包括 workqueue/kthread) 都已经停止了，内核态人物有 可能在停止的时候握有一些信号量，所以如果这时候在外设里面去解锁这个信号 量有可能会发生死锁，所以在外设的 suspend()函数里面作 lock/unlock 锁要非常 小心,这里建议设计的时候就不要在 suspend()里面等待锁. 而且因为 suspend 的时候,有一些 Log 是无法输出的,所以一旦出现问题,非常难调试.

然后 kernel 在这里会尝试释放一些内存.

最后会调用 suspend_devices_and_enter()来把所有的外设休眠，在这个函数中，如果平台注册了 suspend_pos(通常是在板级定义中定义和注册)，这里就会调用 suspend_ops->begin()，然后 driver/base/power/main.c 中的 device_suspend()->dpm_suspend() 会被调用,他们会依次调用驱动的 suspend() 回调来休眠掉所有的设备.

当所有的设备休眠以后，suspend_ops->prepare()会被调用，这个函数通常会作 一些准备工作来让板机进入休眠. 接下来 Linux,在多核的 CPU 中的非启动 CPU 会被关掉，通过注释看到是避免这些其他的 CPU 造成 race condion,接下来的以后只有一个 CPU 在运行了.
suspend_ops 是板级的电源管理操作，通常注册在文件 arch/xxx/mach-xxx/pm.c 中.

接下来，suspend_enter()会被调用，这个函数会关闭 arch irq，调用 device_power_down()，它会调用 suspend_late()函数，这个函数是系统真正进入 休眠最后调用的函数，通常会在这个函数中作最后的检查. 如果检查没问题，接 下来休眠所有的系统设备和总线，并且调用 suspend_pos->enter() 来使 CPU 进入 省电状态. 这时候,就已经休眠了.代码的执行也就停在这里了.

Resume

如果在休眠中系统被中断或者其他事件唤醒，接下来的代码就会开始执行，这个 唤醒的顺序是和休眠的循序相反的,所以系统设备和总线会首先唤醒,使能系统中 断，使能休眠时候停止掉的非启动 CPU，以及调用 suspend_ops->finish()，而且 在 suspend_devices_and_enter()函数中也会继续唤醒每个设备,使能虚拟终端，最后调用 suspend_ops->end().

在返回到 enter_state()函数中的，当 suspend_devices_and_enter() 返回以后，外设已经唤醒了,但是进程和任务都还是冻结状态，这里会调用 suspend_finish()来解冻这些进程和任务，而且发出 Notify 来表示系统已经从 suspend 状态退出，唤醒终端.
到这里，所有的休眠和唤醒就已经完毕了，系统继续运行了.

Android 休眠(suspend)

在一个打过 android 补丁的内核中，state_store()函数会走另外一条路,会进 入到 request_suspend_state()中，这个文件在 earlysuspend.c 中. 这些功能都 是 android 系统加的,后面会对 earlysuspend 和 late resume 进行介绍.

涉及到的文件:
linux_source/kernel/power/main.c
linux_source/kernel/power/earlysuspend.c
linux_source/kernel/power/wakelock.c
特性介绍
Early Suspend
Early suspend 是 android 引进的一种机制, 这种机制在上游备受争议,这里 不做评论. 这个机制作用在关闭显示的时候, 在这个时候, 一些和显示有关的 设备, 比如 LCD 背光, 比如重力感应器, 触摸屏, 这些设备都会关掉, 但是系 统可能还是在运行状态(这时候还有 wake lock)进行任务的处理, 例如在扫描 SD 卡上的文件等. 在嵌入式设备中, 背光是一个很大的电源消耗,所以 android 会加入这样一种机制.

Late Resume
Late Resume 是和 suspend 配套的一种机制, 是在内核唤醒完毕开始执行的. 主要就是唤醒在 Early Suspend 的时候休眠的设备.
Wake Lock
Wake Lock 在 Android 的电源管理系统中扮演一个核心的角色. Wake Lock 是一种锁的机制, 只要有人拿着这个锁, 系统就无法进入休眠, 可以被用户态程序和内核获得. 这个锁可以是有超时的或者是没有超时的, 超时的锁会在时间过去以后自动解锁. 如果没有锁了或者超时了, 内核就会启动休眠的那套机制来进入休眠.
Android Suspend
当用户写入 mem 或者 standby 到 /sys/power/state 中的时候, state_store()会被调用, 然后 Android 会在这里调用 request_suspend_state() 而标准的 Linux 会在这里进入 enter_state()这个函数. 如果请求的是休眠, 那么 early_suspend 这个 workqueue 就会被调用,并且进入 early_suspend 状态.
Early Suspend
在 early_suspend()函数中, 首先会检查现在请求的状态还是否是 suspend, 来 防止 suspend 的请求会在这个时候取消掉(因为这个时候用户进程还在运行),如 果需要退出, 就简单的退出了. 如果没有, 这个函数就会把 early suspend 中 注册的一系列的回调都调用一次, 然后同步文件系统, 然后放弃掉 main_wake_lock, 这个 wake lock 是一个没有超时的锁,如果这个锁不释放, 那 么系统就无法进入休眠.
Late Resume
当所有的唤醒已经结束以后, 用户进程都已经开始运行了, 唤醒通常会是以下的几种原因:

- 来电

如果是来电, 那么 Modem 会通过发送命令给 rild 来让 rild 通知 WindowManager 有来电响应, 这样就会远程调用 PowerManagerService 来写"on" 到/sys/power/state 来执行 late resume 的设备, 比如点亮屏幕等

- 用户按键用户按键事件会送到 WindowManager 中, WindowManager 会处理这些 按键事件,按键分为几种情况, 如果案件不是唤醒键(能够唤醒系统的按键) 那么 WindowManager 会主动放弃 wakeLock 来使系统进入再次休眠, 如果按键 是唤醒键,那么 WindowManger 就会调用 PowerManagerService 中的接口来执行 Late Resume.

- Late Resume 会依次唤醒前面调用了 Early Suspend 的设备.

Wake Lock

我们接下来看一看 wake lock 的机制是怎么运行和起作用的，主要关注 wakelock.c 文件就可以了.

wake lock 有加锁和解锁两种状态，加锁的方式有两种，一种是永久的锁住，这样的锁除非显示的放开，是不会解锁的，所以这种锁的使用是非常小心的. 第二种是超时锁，这种锁会锁定系统唤醒一段时间，如果这个时间过去了，这个锁会自动解除.

锁有两种类型:

1、WAKE_LOCK_SUSPEND 这种锁会防止系统进入睡眠

2、WAKE_LOCK_IDLE 这种锁不会影响系统的休眠，作用我不是很清楚. 在 wake lock 中，会有 3 个地方让系统直接开始 suspend()，分别是:

a、在 wake_unlock()中，如果发现解锁以后没有任何其他的 wake lock 了，就开始休眠

b、在定时器都到时间以后，定时器的回调函数会查看是否有其他的 wake lock，如果没有，就在这里让系统进入睡眠.

c、在 wake_lock() 中，对一个 wake lock 加锁以后，会再次检查一下有没有锁，我想这里的检查是没有必要的，更好的方法是使加锁的这个操作原子化，而 不是繁冗的检查. 而且这样的检查也有可能漏掉

Suspend

当 wake_lock 运行 suspend()以后，在 wakelock.c 的 suspend()函数会被调用,这 个函数首先 sync 文件系统,然后调用 pm_suspend(request_suspend_state),接 下来 pm_suspend()就会调用 enter_state()来进入 Linux 的休眠流程.

```
static void suspend(struct work_struct *work){
        int ret;
        int entry_event_num;

        if(has_wake_lock(WAKE_LOCK_SUSPEND)) {
                if (debug_mask &DEBUG_SUSPEND)
                        pr_info("suspend: abort suspend\n");
                return;
        }

        entry_event_num =current_event_num;
        sys_sync();
        if (debug_mask &DEBUG_SUSPEND)
                pr_info("suspend:enter suspend\n");
        ret =pm_suspend(requested_suspend_state);
        if (current_event_num ==entry_event_num) {
                wake_lock_timeout(&unknown_wakeup, HZ / 2);
        }
```

}
Android 于标准 Linux 休眠的区别
● pm_suspend() 虽然会调用 enter_state()来进入标准的 Linux 休眠流程,但是还 是有一些区别:

当进入冻结进程的时候, android 首先会检查有没有 wake lock,如果没有, 才会停止这些进程,因为在开始 suspend 和冻结进程期间有可能有人申请了 wake lock,如果是这样, 冻结进程会被中断.

● 在 suspend_late()中, 会最后检查一次有没有 wake lock, 这有可能是某种快速申请 wake lock,并且快速释放这个锁的进程导致的,如果有这种情况, 这里会返回错误, 整个 suspend 就会全部放弃.如果 pm_suspend()成功了,LOG 的输出可以通过在 kernel cmd 里面增加 "no_console_suspend" 来看到 suspend 和 resume 过程中的 log 输出

52、防止系统、屏幕休眠（避免服务停止等问题）

大家需要注意，在手机连接电脑时，可以防止手机休眠，但是在断开与电脑连接时，手机会进入休眠状态，有屏幕休眠和 CPU 休眠，若 CPU 休眠了，您的后台服务将会停止，等待唤醒，为此，我们若需要服务时刻运行的话请大家从下面的内容里找答案吧！
1、Activity 启动时防止屏幕关闭
getWindow().setFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON,
WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);//在 SetContentView 前

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

2、唤醒屏幕与解锁：

WakeLock screenLock = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK | PowerManager.ACQUIRE_CAUSES_WAKEUP | PowerManager.ON_AFTER_RELEASE, TAG);

KeyguardManager km = (KeyguardManager) getSystemService(Context.KEYGUARD_SERVICE);
KeyguardLock keyguardLock = km.newKeyguardLock(TAG);
if (km.inKeyguardRestrictedInputMode()) {
    keyguardLock.disableKeyguard(); // 解开屏幕锁
}
keyguardLock.reenableKeyguard(); // 屏幕锁生效
3、防止系统休眠
获取锁：
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE);
WakeLock wakeLock = pm.newWakeLock(PowerManager.SCREEN_DIM_WAKE_LOCK, this.getClass().getCanonicalName());//此参数是使屏幕不自动休眠，特别说明的是，我想让自己的后台服务在屏幕自动休眠之后仍然能够运行，因此使用此参数：PARTIAL_WAKE_LOCK，具体看下面的参数

说明，或查看 androidAPI 源码 doc
wakeLock.acquire();

释放锁：
if(wakeLock !=null && wakeLock.isHeld()) {
wakeLock.release();
wakeLock=null;
}
权限：<uses-permission android:name="android.permission.WAKE_LOCK"/>
参数说明：
PARTIAL_WAKE_LOCK:保持 CPU 运转，屏幕和键盘灯有可能是关闭的。
SCREEN_DIM_WAKE_LOCK：保持 CPU 运转，允许保持屏幕显示但有可能是灰的，允许关闭键盘灯
SCREEN_BRIGHT_WAKE_LOCK：保持 CPU 运转，允许保持屏幕高亮显示，允许关闭键盘灯
FULL_WAKE_LOCK：保持 CPU 运转，保持屏幕高亮显示，键盘灯也保持亮度
ACQUIRE_CAUSES_WAKEUP：
ON_AFTER_RELEASE：wakelock 释放后 Activity 启动

3、 点亮屏幕
PowerManager pm = (PowerManager) getSystemService(Context.POWER_SERVICE); // 获取电源管理器对象
PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.ACQUIRE_CAUSES_WAKEUP | PowerManager.SCREEN_DIM_WAKE_LOCK, "bright");
// 获取 PowerManager.WakeLock 对象,后面的参数|表示同时传入两个值,最后的是 LogCat 里用的 Tag
wl.acquire();// 点亮屏幕
wl.release();// 释放
4、 使屏幕不锁定，即解锁，但是此时屏幕还未点亮
KeyguardManager km = (KeyguardManager) getSystemService(Context.KEYGUARD_SERVICE); // 得到键盘锁管理器对象
KeyguardLock kl = km.newKeyguardLock("unLock");// 参数是 LogCat 里用的 Tag
kl.disableKeyguard();
5、 真正的锁屏

Android 2.2 SDK 提供了一个可管理和操作设备的 API 叫 DevicePolicyManager，使用这个 API 你可以接管手机的应用权限，对手机做出很多大胆的操作，比如锁屏，恢复出厂设置（这么和谐的东西要是在中国是不大可能提供给你的），还有设置密码、强制清除密码，修改密码、设置屏幕灯光渐暗时间间隔等操作。这个 API 可谓是直接可以将你做的应用程序变成系统的老大哥了。虽说是这样，但应用程序可做老大只是对于你本身应用程序有效，别人也可以做类似的应用程序，这个与别人的权限是不起冲突的。

具体的编写代码的流程：

1.因为这个 API 是用的 2.2 提供的 API，所以必须将 sdkVersion 设置为 8，像这样<uses-sdk android:minSdkVersion="8" />，这是必须的。

2.注册一个广播服务类，用以监听权限的变化：

```
<receiver android:name=".deviceAdminReceiver" android:label="@string/app_name"
    android:description="@string/description"
    android:permission="android.permission.BIND_DEVICE_ADMIN">
    <meta-data android:name="android.app.device_admin"
        android:resource="@xml/device_admin" />
    <intent-filter>
        <action android:name="android.app.action.DEVICE_ADMIN_ENABLED" />
    </intent-filter>
</receiver>
```

android:permission 表示此功能需要的权限
android:name="android.app.action.DEVICE_ADMIN_ENABLED" 表示此动作的跳转界面

```
<meta-data android:name="android.app.device_admin"
android:resource="@xml/device_admin" />
```

表示这个应用可以管理的权限清单，xml 清单如下：
device_admin.xml 文件：

```
<?xml version="1.0" encoding="utf-8"?>
<device-admin xmlns:android="http://schemas.android.com/apk/res/android">
<uses-policies>
<limit-password />
<watch-login />
<reset-password />
<force-lock />
<wipe-data />
</uses-policies>
</device-admin>
```

DeviceAdminReceiver 是扩展于 BroadcastReceiver。

1、先开启设备管理：

```
Intent intent = new Intent(
DevicePolicyManager.ACTION_ADD_DEVICE_ADMIN);
intent.putExtra(DevicePolicyManager.EXTRA_DEVICE_ADMIN,
mDeviceComponentName);
intent.putExtra(DevicePolicyManager.EXTRA_ADD_EXPLANATION,
```

"请开启设备管理");
startActivityForResult(intent, RESULT_ENABLE);

1.1、 解除设备管理：

锁屏操作，由于是模拟器不能做到真正错屏，只能停到初始模拟器进来需要解锁的状态，屏幕不会变暗。锁屏代码：

2、锁屏

```
if (mAM.isUserAMonkey()) {
    AlertDialog.Builder builder = new AlertDialog.Builder(
    deviceActivity.this);
    builder.setMessage("你不能对此屏幕进行操作，因为你不是管理员");
    builder.setPositiveButton("I admit defeat", null);
    builder.show();
} else {
    boolean active = mDPM.isAdminActive(mDeviceComponentName);
    if (active) {
        mDPM.lockNow();
    }
}
```

3、屏幕在设置相应时间后灯光变暗效果

```
if (mAM.isUserAMonkey()) {
    AlertDialog.Builder builder = new AlertDialog.Builder(
    deviceActivity.this);
    builder.setMessage("你不能对我的屏幕进行操作，因为你不是管理员");
    builder.setPositiveButton("I admit defeat", null);
    builder.show();
} else {
    boolean active = mDPM.isAdminActive(mDeviceComponentName);
    if (active) {
        long timeout = 1000L * Long.parseLong(et.getText().toString());
        mDPM.setMaximumTimeToLock(mDeviceComponentName, timeout);
    }
}
```

由于是模拟器，恢复出厂设置清除数据后，将无法重新开机，必须重新启动机子，在真机上是没有问题的，测试的时候必须小心，以免将你的数据清除掉。恢复出厂设置代码：
4、恢复出厂设置

```
if (mAM.isUserAMonkey()) {
                AlertDialog.Builder builder = new AlertDialog.Builder(deviceActivity.this);
                builder.setMessage("You can't wipe my data because you are a monkey!");
                builder.setPositiveButton("I admit defeat", null);
                builder.show();
            } else {
                AlertDialog.Builder builder = new Builder(deviceActivity.this);
                builder.setMessage("将重置数据，你确定此操作吗？");
                builder.setPositiveButton(android.R.string.ok,                                    new
DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        // TODO Auto-generated method stub
                        AlertDialog.Builder aler = new AlertDialog.Builder(deviceActivity.this);
                        aler.setMessage("删除数据后，系统将会重新启动.确定吗？");
                        aler.setPositiveButton(android.R.string.ok,                               new
DialogInterface.OnClickListener() {

                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                // TODO Auto-generated method stub
                                boolean                    active                    =
mDPM.isAdminActive(mDeviceComponentName);
                                if (active) {
                                    mDPM.wipeData(0);
                                }
                            }
                        });
                        aler.setNeutralButton(android.R.string.cancel, null);
                        aler.show();
                    }
                });
                builder.setNeutralButton(android.R.string.cancel, null);
                builder.show();
            }
```

Android 锁屏 API-DevicePolicyManager 介绍
作者: Android 开发网原创 时间: 2010-09-25
从 Android 2.2 开始，加入了一个新的锁屏 API 位于

android.app.admin.DevicePolicyManager 包，DevicePolicyManager 类的 lockNow 方法可以锁住屏幕，查看 Android 源代码发现其实是从 IDevicePolicyManager 实现的，整个 AIDL 接口调用代码为:

```
        private final IDevicePolicyManager mService;

    mService = IDevicePolicyManager.Stub.asInterface(
                    ServiceManager.getService(Context.DEVICE_POLICY_SERVICE));

  if (mService != null) {
                try {
                    mService.lockNow();
                } catch (RemoteException e) {
                    Log.w(TAG, "Failed talking with device policy service", e);
                }
            }
```

这里提示大家传统的方法加入 <uses-permission android:name="android.permission.DISABLE_KEYGUARD"></uses-permission>权限，使用下面代码可以锁住键盘，但屏幕不行

```
    KeyguardManager km = (KeyguardManager)getSystemService(Context.KEYGUARD_SERVICE);
    KeyguardLock kl= km.newKeyguardLock(KEYGUARD_SERVICE);
    kl.reenableKeyguard();
```

53、读取 office 文件

1、读取 doc 文件：

解析 doc，要 tm-extractors-0.4.jar 这个包

读取 doc 文件方法：

```
public static String readDOC(String path) {
// path 形如：  Environment.getExternalStorageDirectory().getAbsolutePath()+ "/aa.doc")
//  创建输入流读取 doc 文件
      FileInputStream in;
      String text = null;
      try {
            in = new FileInputStream(new File(path));
            int a = in.available();
            WordExtractor extractor = null;
            //  创建 WordExtractor
```

```
            extractor = new WordExtractor();
            // 对 doc 文件进行提取
            text = extractor.extractText(in);
            System.out.println("解析得到的东西" + text);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (text == null) {
            text = "解析文件出现问题";
        }
        return text;
    }
```

读取 docx 文件方法如下：

```
    public static String readDOCX(String path) {
        String river = "";
        try {
            ZipFile xlsxFile = new ZipFile(new File(path));
            ZipEntry sharedStringXML = xlsxFile.getEntry("word/document.xml");
            InputStream inputStream = xlsxFile.getInputStream(sharedStringXML);
            XmlPullParser xmlParser = Xml.newPullParser();
            xmlParser.setInput(inputStream, "utf-8");
            int evtType = xmlParser.getEventType();
            while (evtType != XmlPullParser.END_DOCUMENT) {
                switch (evtType) {
                case XmlPullParser.START_TAG:
                    String tag = xmlParser.getName();
                    System.out.println(tag);
                    if (tag.equalsIgnoreCase("t")) {
                        river += xmlParser.nextText() + "\n";
                    }
                    break;
                case XmlPullParser.END_TAG:
                    break;
                default:
                    break;
                }
                evtType = xmlParser.next();
            }
        } catch (ZipException e) {
            e.printStackTrace();
        } catch (IOException e) {
```

```
            e.printStackTrace();
        } catch (XmlPullParserException e) {
            e.printStackTrace();
        }
        if (river == null) {
            river = "解析文件出现问题";
        }
        return river;
    }
```

2、 读取 xls 文件：

解析 xls，要 jxl.jar 这个包

读取 xls 方法如下：

```
    public static String readXLS(String path) {
        String str = "";
        try {
            Workbook workbook = null;
            workbook = Workbook.getWorkbook(new File(path));
            Sheet sheet = workbook.getSheet(0);
            Cell cell = null;
            int columnCount = sheet.getColumns();
            int rowCount = sheet.getRows();
            for (int i = 0; i < rowCount; i++) {
                for (int j = 0; j < columnCount; j++) {
                    cell = sheet.getCell(j, i);
                    String temp2 = "";
                    if (cell.getType() == CellType.NUMBER) {
                        temp2 = ((NumberCell) cell).getValue() + "";
                    } else if (cell.getType() == CellType.DATE) {
                        temp2 = "" + ((DateCell) cell).getDate();
                    } else {
                        temp2 = "" + cell.getContents();
                    }
                    str = str + "    " + temp2;
                }
                str += "\n";
            }
            workbook.close();
        } catch (Exception e) {
        }
        if (str == null) {
            str = "解析文件出现问题";
        }
```

```
        return str;
    }
```

读取 xlsx 方法如下：

```
    public static String readXLSX(String path) {
        String str = "";
        String v = null;
        boolean flat = false;
        List<String> ls = new ArrayList<String>();
        try {
            ZipFile xlsxFile = new ZipFile(new File(path));
            ZipEntry sharedStringXML = xlsxFile.getEntry("xl/sharedStrings.xml");
            InputStream inputStream = xlsxFile.getInputStream(sharedStringXML);
            XmlPullParser xmlParser = Xml.newPullParser();
            xmlParser.setInput(inputStream, "utf-8");
            int evtType = xmlParser.getEventType();
            while (evtType != XmlPullParser.END_DOCUMENT) {
                switch (evtType) {
                case XmlPullParser.START_TAG:
                    String tag = xmlParser.getName();
                    if (tag.equalsIgnoreCase("t")) {
                        ls.add(xmlParser.nextText());
                    }
                    break;
                case XmlPullParser.END_TAG:
                    break;
                default:
                    break;
                }
                evtType = xmlParser.next();
            }
            ZipEntry sheetXML = xlsxFile.getEntry("xl/worksheets/sheet1.xml");
            InputStream inputStreamsheet = xlsxFile.getInputStream(sheetXML);
            XmlPullParser xmlParsersheet = Xml.newPullParser();
            xmlParsersheet.setInput(inputStreamsheet, "utf-8");
            int evtTypesheet = xmlParsersheet.getEventType();
            while (evtTypesheet != XmlPullParser.END_DOCUMENT) {
                switch (evtTypesheet) {
                case XmlPullParser.START_TAG:
                    String tag = xmlParsersheet.getName();
                    if (tag.equalsIgnoreCase("row")) {
                    } else if (tag.equalsIgnoreCase("c")) {
                        String t = xmlParsersheet.getAttributeValue(null, "t");
                        if (t != null) {
```

```
                        flat = true;
                        System.out.println(flat + "有");
                    } else {
                        System.out.println(flat + "没有");
                        flat = false;
                    }
                } else if (tag.equalsIgnoreCase("v")) {
                    v = xmlParsersheet.nextText();
                    if (v != null) {
                        if (flat) {
                            str += ls.get(Integer.parseInt(v)) + "    ";
                        } else {
                            str += v + "    ";
                        }
                    }
                }
                break;
            case XmlPullParser.END_TAG:
                if (xmlParsersheet.getName().equalsIgnoreCase("row")
                && v != null) {
                    str += "\n";
                }
                break;
            }
            evtTypesheet = xmlParsersheet.next();
        }
        System.out.println(str);
    } catch (ZipException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (XmlPullParserException e) {
        e.printStackTrace();
    }
    if (str == null) {
        str = "解析文件出现问题";
    }
    return str;
}
```

54、设置 ListView 滚动条属性

哈哈，让我发现了（不过 2.0 后才支持），fading 就是淡去的意思

void android.view.View.setScrollbarFadingEnabled(boolean fadeScrollbars)

public void setScrollbarFadingEnabled (boolean fadeScrollbars)

Since: API Level 5
Define whether scrollbars will fade when the view is not scrolling.

Parameters
fadeScrollbars　wheter to enable fading

所以 setScrollbarFadingEnabled(true);就可以啦！！！！
当然，很明显不仅仅是 ListView 可以这样设置，继承 View 类的都可以啊！

55、获取 Array.xml 文件中的值

比如在 arrays.xml 里:
<!--leo added for KYLIN-496-->
<string-array name="reboot_item">
<item>Reboot</item>
<item>Recovery</item>
<item>BootLoader</item>
</string-array>

在代码里获取:
String item0 =this.getResources().getStringArray(R.array.reboot_item)[0];
String item1 = this.getResources().getStringArray(R.array.reboot_item)[1];
CharSequence[] items　＝　this.getResources().getStringArray(R.array.reboot_item);

56、获取系统媒体声音文件

获取文件的方法：

```
    private void getSystemRing() {
        ContentResolver cr = this.getContentResolver();
        String[] cols = new String[] { MediaStore.Audio.Media._ID,
MediaStore.Audio.Media.DATA, MediaStore.Audio.Media.DISPLAY_NAME };
        Cursor cursor = cr.query(MediaStore.Audio.Media.INTERNAL_CONTENT_URI, cols,
null, null, null);
        if (cursor.moveToFirst()) {
            do {
                Log.e("", cursor.getString(0) + ":" + cursor.getString(1) + ":" +
cursor.getString(2));
            } while (cursor.moveToNext());
        }

    }
```

从以下代码中找出区分媒体文件类型的方法：

```
Uri uri = MediaStore.Audio.Media.getContentUriForPath(file.getAbsolutePath());
                ContentValues values = new ContentValues();
                values.put(MediaStore.MediaColumns.TITLE, files[index].title);
                values.put(MediaStore.MediaColumns.MIME_TYPE, "audio/mp3");
                values.put(MediaStore.MediaColumns.DATA, file.getAbsolutePath());
                values.put(MediaStore.Audio.Media.DATA, file.getAbsolutePath());
                values.put(MediaStore.Audio.Media.ARTIST, Options.TITLE);
                values.put(MediaStore.Audio.Media.IS_RINGTONE, type == RingtoneManager.TYPE_RINGTONE);
                values.put(MediaStore.Audio.Media.IS_NOTIFICATION, type == RingtoneManager.TYPE_NOTIFICATION);
                values.put(MediaStore.Audio.Media.IS_ALARM, type == RingtoneManager.TYPE_ALARM);
                values.put(MediaStore.Audio.Media.IS_MUSIC, false);
                getContentResolver().delete(uri, MediaStore.MediaColumns.DATA + "=\"" + file.getAbsolutePath() + "\"", null);
                Uri newUri = getContentResolver().insert(uri, values);
```

57、自定义 Adapter

用于更新显示 listview 中自定义 View，
listview 中若存在 imageview 或 textview 等常用控件时，直接使用个 simpleAdapter 即可将要显示的内容传递给它们，若 listview 存在的是自己定义的一个 View，而此 view 需要进行不同效果的显示，则可通过自定义一个 Adapter 来传值给 listview 中自定义的 view，进行不同的显示效果，方法如下：

58、记住 listview 滚动位置

1. 记住最后一次点击的 id: int id = lv.getSelectedItemPosition();
2. ActivitynCreate() 最后加上次语句：lv.setSelection(id);

59、更改系统超时休眠的时间

默认情况下，Android 系统在超过 N 分钟没操作，会自动关屏并进入休眠状态。
实际上，有些项目要求超时不休眠，如果只是针对单个应用程序，我们可以通过电源管理设置状态来实现，
而如果要设置所有应用的超时时间，则可以参考以下方法：

方法一、调整代码：
Settings.System.putInt(getContentResolver(),android.provider.Settings.System.SCREEN_OFF_TIMEOUT,-1);
权限：<uses-permission android:name="android.permission.WRITE_SETTINGS" />

方法二、调整数据库：
android 的这些设置都是存放在 sql 数据库里的，也就是说可以直接通过修改数据库来不让 android 睡眠。
sqlite3 /data/data/com.android.providers.settings/databases/settings.db
具体 sql：
UPDATE system SET value = '-1' WHERE name = 'screen_off_timeout' ;

60、更改对话框大小

1.在 manifest.xml 中 activity 添加
android:theme="@android:style/Theme.Dialog"
2.WindowManager m = getWindowManager();
Display d = m.getDefaultDisplay(); //为获取屏幕宽、高

LayoutParams p = getWindow().getAttributes(); //获取对话框当前的参数值
p.height = (int) (d.getHeight() * 0.6); //高度设置为屏幕的 0.6
p.width = (int) (d.getWidth() * 0.95); //宽度设置为屏幕的 0.95

getWindow().setAttributes(p); //设置生效

61、json 数据格式解析

普通形式的：
服务器端返回的 json 数据格式如下：

{"userbean":{"Uid":"100196","Showname":"\u75af\u72c2\u7684\u7334\u5b50","Avtar":null,"State":1}}

分析代码如下：

```
// TODO  状态处理  500 200
                int res = 0;
                res = httpClient.execute(httpPost).getStatusLine().getStatusCode();
                if (res == 200) {
                    /*
                     * 当返回码为 200 时，做处理
                     * 得到服务器端返回 json 数据，并做处理
                     * */
```

```
                        HttpResponse httpResponse = httpClient.execute(httpPost);
                        StringBuilder builder = new StringBuilder();
                        BufferedReader bufferedReader2 = new BufferedReader(
                                new
InputStreamReader(httpResponse.getEntity().getContent()));
                        String str2 = "";
                        for (String s = bufferedReader2.readLine(); s != null; s =
bufferedReader2
                                .readLine()) {
                            builder.append(s);
                        }
                        Log.i("cat", ">>>>>>" + builder.toString());

JSONObject jsonObject = new JSONObject(builder.toString())
                                .getJSONObject("userbean");

                        String Uid;
                        String Showname;
                        String Avtar;
                        String State;

                        Uid = jsonObject.getString("Uid");
                        Showname = jsonObject.getString("Showname");
                        Avtar = jsonObject.getString("Avtar");
                        State = jsonObject.getString("State");
```

带数组形式的：

服务器端返回的数据格式为：

```
{"calendar":
    {"calendarlist":
            [

{"calendar_id":"1705","title":"(\u4eb2\u5b50)ddssd","category_name":"\u9ed8\u8ba4\u5206\u7c7
b","showtime":"1288927800","endshowtime":"1288931400","allDay":false},

{"calendar_id":"1706","title":"(\u65c5\u884c)","category_name":"\u9ed8\u8ba4\u5206\u7c7b","s
howtime":"1288933200","endshowtime":"1288936800","allDay":false}
            ]
    }
}
```

分析代码如下：

```
// TODO  状态处理  500 200
```

```
                int res = 0;
                res = httpClient.execute(httpPost).getStatusLine().getStatusCode();
                if (res == 200) {
                    /*
                     * 当返回码为 200 时，做处理
                     * 得到服务器端返回 json 数据，并做处理
                     * */
                    HttpResponse httpResponse = httpClient.execute(httpPost);
                    StringBuilder builder = new StringBuilder();
                    BufferedReader bufferedReader2 = new BufferedReader(
                            new
InputStreamReader(httpResponse.getEntity().getContent()));
                    String str2 = "";
                    for (String s = bufferedReader2.readLine(); s != null; s =
bufferedReader2
                            .readLine()) {
                        builder.append(s);
                    }
                    Log.i("cat", ">>>>>>" + builder.toString());
                    /**
                     * 这里需要分析服务器回传的 json 格式数据，
                     */
                    JSONObject jsonObject = new JSONObject(builder.toString())
                            .getJSONObject("calendar");
                    JSONArray jsonArray = jsonObject.getJSONArray("calendarlist");
                    for(int i=0;i<jsonArray.length();i++){
                        JSONObject jsonObject2 = (JSONObject)jsonArray.opt(i);
                        CalendarInfo calendarInfo = new CalendarInfo();
                        calendarInfo.setCalendar_id(jsonObject2.getString("calendar_id"));
                        calendarInfo.setTitle(jsonObject2.getString("title"));

calendarInfo.setCategory_name(jsonObject2.getString("category_name"));
                        calendarInfo.setShowtime(jsonObject2.getString("showtime"));
                        calendarInfo.setEndtime(jsonObject2.getString("endshowtime"));
                        calendarInfo.setAllDay(jsonObject2.getBoolean("allDay"));
                        calendarInfos.add(calendarInfo);
                    }
```

总结，普通形式的只需用 JSONObject ，带数组形式的需要使用 JSONArray 将其变成一个 list。

62、两种 Toast

1、带图片

Toast toast = new Toast（this）;
ImageView view = new ImageView（this）;
view.setImageResource（R.drawable.icon）;
toast.setView（view）;
toast.show（）;

3、 使用文字对话框

Toast toast = Toast.makeText（this， "lalalal"， Toast.LENGTH_LONG）;
View textView = toast.getView（）;
LinearLayout lay = new LinearLayout（this）;
lay.setOrientation（LinearLayout.HORIZONTAL）;
ImageView view = new ImageView（this）;
view.setImageResource（R.drawable.icon）;
lay.addView（view）;
lay.addView（textView）;
toast.setView（lay）;
toast.show（）;

63、控件抖动的实现

动画效果 anim/cycle.xml 文件内容

```
<?xml version="1.0" encoding="utf-8"?>
<cycleInterpolator xmlns：
android="http：//schemas.android.com/apk/res/android"
android：cycles="20" />
```

1、X 轴抖动

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns：
android="http：//schemas.android.com/apk/res/android"
android：fromXDelta="0"
android：toXDelta="10"
android：duration="1000"
android：interpolator="@anim/cycle" />
```

2、Y 轴抖动

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns：
android="http：//schemas.android.com/apk/res/android"
android：duration="1000"
android：fromYDelta="0"
android：interpolator="@anim/cycle"
android：toYDelta="10" >
</translate>
```

64、判断媒体文件类型

```
package com.ekangcn.healthtangnb.util；
import java.util.HashMap；
import java.util.Iterator；
/**
* MediaScanner helper class.
*
* {@hide}
*/
public class MediaFile {
// comma separated list of all file extensions supported by the media scanner
public static String sFileExtensions；
// Audio file types
public static final int FILE_TYPE_MP3 = 1；
public static final int FILE_TYPE_M4A = 2；
public static final int FILE_TYPE_WAV = 3；
public static final int FILE_TYPE_AMR = 4；
public static final int FILE_TYPE_AWB = 5；
public static final int FILE_TYPE_WMA = 6；
public static final int FILE_TYPE_OGG = 7；
private static final int FIRST_AUDIO_FILE_TYPE = FILE_TYPE_MP3；
private static final int LAST_AUDIO_FILE_TYPE = FILE_TYPE_OGG；
// MIDI file types
public static final int FILE_TYPE_MID = 11；
public static final int FILE_TYPE_SMF = 12；
public static final int FILE_TYPE_IMY = 13；
private static final int FIRST_MIDI_FILE_TYPE = FILE_TYPE_MID；
private static final int LAST_MIDI_FILE_TYPE = FILE_TYPE_IMY；
// Video file types
public static final int FILE_TYPE_MP4 = 21；
public static final int FILE_TYPE_M4V = 22；
public static final int FILE_TYPE_3GPP = 23；
public static final int FILE_TYPE_3GPP2 = 24；
public static final int FILE_TYPE_WMV = 25；
private static final int FIRST_VIDEO_FILE_TYPE = FILE_TYPE_MP4；
private static final int LAST_VIDEO_FILE_TYPE = FILE_TYPE_WMV；
// Image file types
public static final int FILE_TYPE_JPEG = 31；
public static final int FILE_TYPE_GIF = 32；
public static final int FILE_TYPE_PNG = 33；
public static final int FILE_TYPE_BMP = 34；
public static final int FILE_TYPE_WBMP = 35；
```

```
private static final int FIRST_IMAGE_FILE_TYPE = FILE_TYPE_JPEG；
private static final int LAST_IMAGE_FILE_TYPE = FILE_TYPE_WBMP；
// Playlist file types
public static final int FILE_TYPE_M3U = 41；
public static final int FILE_TYPE_PLS = 42；
public static final int FILE_TYPE_WPL = 43；
private static final int FIRST_PLAYLIST_FILE_TYPE = FILE_TYPE_M3U；
private static final int LAST_PLAYLIST_FILE_TYPE = FILE_TYPE_WPL；
static class MediaFileType {
int fileType；
String mimeType；
MediaFileType（int fileType， String mimeType） {
this.fileType = fileType；
this.mimeType = mimeType；
}
}
private static HashMap<String， MediaFileType> sFileTypeMap
= new HashMap<String， MediaFileType>（）；
private static HashMap<String， Integer> sMimeTypeMap
= new HashMap<String， Integer>（）；
static void addFileType（String extension， int fileType， String mimeType） {
sFileTypeMap.put（extension， new MediaFileType（fileType， mimeType））；
sMimeTypeMap.put（mimeType， new Integer（fileType））；
}
static {
addFileType（"MP3"， FILE_TYPE_MP3， "audio/mpeg"）；
addFileType（"M4A"， FILE_TYPE_M4A， "audio/mp4"）；
addFileType（"WAV"， FILE_TYPE_WAV， "audio/x-wav"）；
addFileType（"AMR"， FILE_TYPE_AMR， "audio/amr"）；
addFileType（"AWB"， FILE_TYPE_AWB， "audio/amr-wb"）；
addFileType（"WMA"， FILE_TYPE_WMA， "audio/x-ms-wma"）；
addFileType（"OGG"， FILE_TYPE_OGG， "application/ogg"）；
addFileType（"MID"， FILE_TYPE_MID， "audio/midi"）；
addFileType（"XMF"， FILE_TYPE_MID， "audio/midi"）；
addFileType（"RTTTL"， FILE_TYPE_MID， "audio/midi"）；
addFileType（"SMF"， FILE_TYPE_SMF， "audio/sp-midi"）；
addFileType（"IMY"， FILE_TYPE_IMY， "audio/imelody"）；
addFileType（"MP4"， FILE_TYPE_MP4， "video/mp4"）；
addFileType（"M4V"， FILE_TYPE_M4V， "video/mp4"）；
addFileType（"3GP"， FILE_TYPE_3GPP， "video/3gpp"）；
addFileType（"3GPP"， FILE_TYPE_3GPP， "video/3gpp"）；
addFileType（"3G2"， FILE_TYPE_3GPP2， "video/3gpp2"）；
addFileType（"3GPP2"， FILE_TYPE_3GPP2， "video/3gpp2"）；
addFileType（"WMV"， FILE_TYPE_WMV， "video/x-ms-wmv"）；
```

```
addFileType（"JPG"， FILE_TYPE_JPEG， "image/jpeg"）；
addFileType（"JPEG"， FILE_TYPE_JPEG， "image/jpeg"）；
addFileType（"GIF"， FILE_TYPE_GIF， "image/gif"）；
addFileType（"PNG"， FILE_TYPE_PNG， "image/png"）；
addFileType（"BMP"， FILE_TYPE_BMP， "image/x-ms-bmp"）；
addFileType（"WBMP"， FILE_TYPE_WBMP， "image/vnd.wap.wbmp"）；
addFileType（"M3U"， FILE_TYPE_M3U， "audio/x-mpegurl"）；
addFileType（"PLS"， FILE_TYPE_PLS， "audio/x-scpls"）；
addFileType（"WPL"， FILE_TYPE_WPL， "application/vnd.ms-wpl"）；
// compute file extensions list for native Media Scanner
StringBuilder builder = new StringBuilder（）；
Iterator<String> iterator = sFileTypeMap.keySet（）.iterator（）；
while （iterator.hasNext（））｛
if （builder.length（） > 0）｛
builder.append（'，'）；
}
builder.append（iterator.next（））；
}
sFileExtensions = builder.toString（）；
}
public static final String UNKNOWN_STRING = "<unknown>"；
public static boolean isAudioFileType（int fileType）｛
return （（fileType >= FIRST_AUDIO_FILE_TYPE &&
fileType <= LAST_AUDIO_FILE_TYPE） ||
（fileType >= FIRST_MIDI_FILE_TYPE &&
fileType <= LAST_MIDI_FILE_TYPE））；
}
public static boolean isVideoFileType（int fileType）｛
return （fileType >= FIRST_VIDEO_FILE_TYPE &&
fileType <= LAST_VIDEO_FILE_TYPE）；
}
public static boolean isImageFileType（int fileType）｛
return （fileType >= FIRST_IMAGE_FILE_TYPE &&
fileType <= LAST_IMAGE_FILE_TYPE）；
}
public static boolean isPlayListFileType（int fileType）｛
return （fileType >= FIRST_PLAYLIST_FILE_TYPE &&
fileType <= LAST_PLAYLIST_FILE_TYPE）；
}
public static MediaFileType getFileType（String path）｛
int lastDot = path.lastIndexOf（"."）；
if （lastDot < 0）
return null；
return sFileTypeMap.get（path.substring（lastDot + 1）.toUpperCase（））；
```

```
}
//重载方法，根据文件路径识别音频文件
public static boolean isAudioFileType（String path） {
MediaFileType type = getFileType（path）;
if（null ！= type） {
return isAudioFileType（type.fileType）;
}
return false;
}
public static int getFileTypeForMimeType（String mimeType） {
Integer value = sMimeTypeMap.get（mimeType）;
return （value == null ？ 0 ： value.intValue（））;
}
}
```

65、编写使用 root 权限的应用

如图：


```
public static boolean runRootCommand（String command） {
Process process = null;
DataOutputStream os = null;
try {
process = Runtime.getRuntime（）.exec（"su"）;
os = new DataOutputStream（process.getOutputStream（））;
os.writeBytes（command+"\n"）;
os.writeBytes（"exit\n"）;
os.flush（）;
process.waitFor（）;
} catch （Exception e） {
Log.d（TAG， "the device is not rooted， error message： " + e.getMessage（））;
return false;
```

```
} finally {
try {
if （os ！= null） {
os.close（）;
}
if（process ！= null） {
process.destroy（）;
}
} catch （Exception e） {
e.printStackTrace（）;
}
}
return true;
}
```

66、获取所有安装了的 App 的信息

```
class PInfo {
private String appname = "";
private String pname = "";
private String versionName = "";
private int versionCode = 0;
private Drawable icon;
private void prettyPrint（） {
log（appname + "\t" + pname + "\t" + versionName + "\t" + versionCode + "\t"）;
}
}
```

```
private void listPackages（） {
ArrayList<PInfo> apps = getInstalledApps（false）;  /* false = no system packages */
final int max = apps..size（）;
for （int i=0;  i<max;  i++） {
apps.get（i）.prettyPrint（）;
}
}
private ArrayList<PInfo> getInstalledApps（boolean getSysPackages） {
ArrayList<PInfo> res = new ArrayList<PInfo>（）;
List<PackageInfo> packs = getPackageManager（）.getInstalledPackages（0）;
for（int i=0; i<packs.size（）; i++） {
PackageInfo p = packs.get（i）;
if （（！ getSysPackages） && （p.versionName == null）） {
continue ;
}
```

```
PInfo newInfo = new PInfo（）;
newInfo.appname = p.applicationInfo.loadLabel（getPackageManager（））.toString（）;
newInfo.pname = p.packageName;
newInfo.versionName = p.versionName;
newInfo.versionCode = p.versionCode;
newInfo.icon = p.applicationInfo.loadIcon（getPackageManager（））;
res.add（newInfo）;
}
return res;
}
```

67、帧动画

1、Java 代码：

```
package eoe.demo;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
/**
* @description android 中的逐帧动画.
* 逐帧动画的原理很简单，跟电影的播放一样，一张张类似的图片不停的切换，当切换速度达到一定值时，
* 我们的视觉就会出现残影，残影的出现保证了视觉上变化的连续性，这时候图片的切换看在我们眼中就跟真实的一样了。
* 想使用逐帧动画：
* 第一步：需要在 res/drawable 文件夹下新建一个 xml 文件，该文件详细定义了动画播放时所用的图片、切换每张图片
* 所用的时间、是否为连续播放等等。(有些文章说，在 res/anim 文件夹下放置该文件，事实证明，会出错哦)
* 第二步：在代码中，将该动画布局文件，赋值给特定的图片展示控件，如本例子中的 ImageView。
* 第三步：通过 imageView.getBackGround()获取相应的 AnimationDrawable 对象，然后通过该对象的方法进行控制动画
*
*/
public class Animation1Activity extends Activity {
ImageView imageView ;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.animation1);
imageView = (ImageView) findViewById(R.id.imageView_animation1);
```

```
imageView.setBackgroundResource(R.drawable.animation1_drawable);
}
public void myClickHandler(View targetButton){
// 获取 AnimationDrawable 对象
AnimationDrawable animationDrawable = (AnimationDrawable)imageView.getBackground();
// 动画是否正在运行
if(animationDrawable.isRunning()){
//停止动画播放
animationDrawable.stop();
}
else{
//开始或者继续动画播放
animationDrawable.start();
}
}
}
```

2、animation1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">
<Button
android:id="@+id/button_animation1"
android:text="动画开始"
android:layout_gravity="center_horizontal"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="myClickHandler">
</Button>
<ImageView
android:id="@+id/imageView_animation1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_weight="1">
</ImageView>
</LinearLayout>
```

3、 animation1_drawable.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
根标签为 animation-list，其中 oneshot 代表着是否只展示一遍，设置为 false 会不停的循环播放动画
根标签下，通过 item 标签对动画中的每一个图片进行声明
```

```
android:duration 表示展示所用的该图片的时间长度
-->
<animation-list
xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="false"
>
<item android:drawable="@drawable/a1" android:duration="50"></item>
<item android:drawable="@drawable/a2" android:duration="50"></item>
<item android:drawable="@drawable/a3" android:duration="50"></item>
<item android:drawable="@drawable/a4" android:duration="50"></item>
<item android:drawable="@drawable/a5" android:duration="50"></item>
<item android:drawable="@drawable/a6" android:duration="50"></item>
</animation-list
```

除此之外：在 AnimationDrawable 中，我们还可以看到如下的几个重要方法：

setOneShot(boolean flag) 和在配置文件中进行配置一样，可以设置动画是否播放一次，false 为连续播放；

addFrame (Drawable frame, int duration) 动态的添加一个图片进入该动画中

68、scrollview

1、横向反弹效果

```
public class MyScrollView extends HorizontalScrollView {
private View inner;
private float x;
private Rect normal = new Rect();
@Override
protected void onFinishInflate() {
if (getChildCount() > 0) {
inner = getChildAt(0);
}
System.out.println("getChildCount():" + getChildCount());
}
public MyScrollView(Context context, AttributeSet attrs) {
super(context, attrs);
}
@Override
public boolean onTouchEvent(MotionEvent ev) {
if (inner == null) {
return super.onTouchEvent(ev);
} else {
commOnTouchEvent(ev);
```

```
}
return super.onTouchEvent(ev);
}
public void commOnTouchEvent(MotionEvent ev) {
int action = ev.getAction();
switch (action) {
case MotionEvent.ACTION_DOWN:
x = ev.getX();
break;
case MotionEvent.ACTION_UP:
if (isNeedAnimation()) {
animation();
}
break;
case MotionEvent.ACTION_MOVE:
final float preX = x;
float nowX = ev.getX();
int deltaX = (int) (preX - nowX);
// 滚动
scrollBy(0, deltaX);
x = nowX;
// 当滚动到最左或者最右时就不会再滚动，这时移动布局
if (isNeedMove()) {
if (normal.isEmpty()) {
// 保存正常的布局位置
normal.set(inner.getLeft(), inner.getTop(), inner.getRight(), inner.getBottom());
}
// 移动布局
 inner.layout(inner.getLeft() - deltaX/2, inner.getTop() , inner.getRight()- deltaX/2,
inner.getBottom() );
}
break;
default:
break;
}
}
// 开启动画移动
public void animation() {
// 开启移动动画
TranslateAnimation ta = new TranslateAnimation(0, 0, inner.getTop(), normal.top);
ta.setDuration(200);
inner.startAnimation(ta);
// 设置回到正常的布局位置
inner.layout(normal.left, normal.top, normal.right, normal.bottom);
```

```
normal.setEmpty();
}
// 是否需要开启动画
public boolean isNeedAnimation() {
return !normal.isEmpty();
}
// 是否需要移动布局
public boolean isNeedMove() {
int offset = inner.getMeasuredWidth() - getWidth();
int scrollX = getScrollX();
if (scrollX == 0 || scrollX == offset) {
return true;
}
return false;
}
}
```

2、整个屏幕横向滚动

```
import java.lang.reflect.Field;

import android.content.Context;
import android.util.AttributeSet;
import android.view.animation.AnimationUtils;
import android.view.animation.Interpolator;
import android.view.animation.OvershootInterpolator;
import android.widget.HorizontalScrollView;
import android.widget.OverScroller;
import android.widget.Scroller;

public class HorizontalScrollViewEx extends HorizontalScrollView {
    static final int ANIMATED_SCROLL_GAP = 250;
    private long mLastScroll;

    private Field mScrollerField;
    ScrollerEx scrollerEx = null;

    public HorizontalScrollViewEx(Context context) {
        this(context, null);

    }

    public HorizontalScrollViewEx(Context context, AttributeSet attrs) {
        super(context, attrs);
```

```
        this.setSmoothScrollingEnabled(false);
        initScroller();
    }

    public HorizontalScrollViewEx(Context context, AttributeSet attrs, int defStyle) {

        super(context, attrs, defStyle);
        this.setSmoothScrollingEnabled(false);
        initScroller();
    }

    private void initScroller() {
        try {
            mScrollerField = HorizontalScrollView.class.getDeclaredField("mScroller");
            mScrollerField.setAccessible(true);
            String type = mScrollerField.getType().getSimpleName();

            if ("OverScroller".equals(type)) {
                scrollerEx = new ScrollerEx() {
                    private OverScroller mScroller = null;

                    public void startScroll(int startX, int startY, int dx, int dy, int
duration) {
                        mScroller.startScroll(startX, startY, dx, dy, duration);
                    }

                    public boolean isFinished() {
                        return mScroller.isFinished();
                    }

                    public Object getScroller() {
                        return mScroller;
                    }

                    public void create(Context context, Interpolator interpolator) {
                        mScroller = new OverScroller(context, interpolator);
                    }

                    public void abortAnimation() {
                        if (mScroller != null) {
                            mScroller.abortAnimation();
                        }
                    }
                };
```

```
                } else {
                    scrollerEx = new ScrollerEx() {
                        private Scroller mScroller = null;

                        public void startScroll(int startX, int startY, int dx, int dy, int
duration) {
                            mScroller.startScroll(startX, startY, dx, dy, duration);
                        }

                        public boolean isFinished() {
                            return mScroller.isFinished();
                        }

                        public Object getScroller() {
                            return mScroller;
                        }

                        public void create(Context context, Interpolator interpolator) {
                            mScroller = new Scroller(context, interpolator);
                        }

                        public void abortAnimation() {
                            if (mScroller != null) {
                                mScroller.abortAnimation();
                            }
                        }
                    };
                }

            } catch (Exception ex) {
            }
        }

        public final void smoothScrollBy(int dx, int dy, int addDuration) {

            float tension = 0f;

            scrollerEx.abortAnimation();

            Interpolator ip = new OvershootInterpolator(tension);
            scrollerEx.create(getContext(), ip);

            try {
                mScrollerField.set(this, scrollerEx.getScroller());
```

```
            } catch (Exception e) {
            }

            long duration = AnimationUtils.currentAnimationTimeMillis() - mLastScroll;
            if (duration > ANIMATED_SCROLL_GAP) {
                scrollerEx.startScroll(getScrollX(), getScrollY(), dx, dy, addDuration);

                awakenScrollBars();

                // awakenScrollBars(mScroller.getDuration());

                invalidate();
            } else {
                if (!scrollerEx.isFinished()) {
                    scrollerEx.abortAnimation();
                }
                scrollBy(dx, dy);
            }
            mLastScroll = AnimationUtils.currentAnimationTimeMillis();
        }
        public final void smoothScrollTo(int x, int y, int duration) {
            smoothScrollBy(x - getScrollX(), y - getScrollY(), duration);
        }
        private interface ScrollerEx {
            void create(Context context, Interpolator interpolator);
            Object getScroller();
            void abortAnimation();
            void startScroll(int startX, int startY, int dx, int dy, int duration);
            boolean isFinished();
        }
    }
```

69、内存泄露分析

1、内存检测

判断是否会有内存泄露

内存监测工具 DDMS --> Heap

无论怎么小心，想完全避免 bad code 是不可能的，此时就需要一些工具来帮助我们检查代码中是否存在会造成内存泄漏的地方。Android tools 中的 DDMS 就带有一个很不错的内存监测工具 Heap(这里我使用 eclipse 的 ADT 插件，并以真机为例，在模拟器中的情况类似)。用 Heap 监测应用进程使用内存情况的步骤如下：

1. 启动 eclipse 后，切换到 DDMS 透视图，并确认 Devices 视图、Heap 视图都是打开的；

2. 将手机通过 USB 链接至电脑，链接时需要确认手机是处于“USB 调试”模式，而不是作为“Mass Storage”；
3. 链接成功后，在 DDMS 的 Devices 视图中将会显示手机设备的序列号，以及设备中正在运行的部分进程信息；
4. 点击选中想要监测的进程，比如 system_process 进程；
5. 点击选中 Devices 视图界面中最上方一排图标中的“Update Heap”图标；
6. 点击 Heap 视图中的“Cause GC”按钮；
7. 此时在 Heap 视图中就会看到当前选中的进程的内存使用量的详细情况。
说明：
a) 点击“Cause GC”按钮相当于向虚拟机请求了一次 gc 操作；
b) 当内存使用信息第一次显示以后，无须再不断的点击“Cause GC”，Heap 视图界面会定时刷新，在对应用的不断的操作过程中就可以看到内存使用的变化；
c) 内存使用信息的各项参数根据名称即可知道其意思，在此不再赘述。
如何才能知道我们的程序是否有内存泄漏的可能性呢。这里需要注意一个值：Heap 视图中部有一个 Type 叫做 data object，即数据对象，也就是我们的程序中大量存在的类类型的对象。在 data object 一行中有一列是“Total Size”，其值就是当前进程中所有 Java 数据对象的内存总量，一般情况下，这个值的大小决定了是否会有内存泄漏。可以这样判断：
a) 不断的操作当前应用，同时注意观察 data object 的 Total Size 值；
b) 正常情况下 Total Size 值都会稳定在一个有限的范围内，也就是说由于程序中的的代码良好，没有造成对象不被垃圾回收的情况，所以说虽然我们不断的操作会不断的生成很多对象，而在虚拟机不断的进行 GC 的过程中，这些对象都被回收了，内存占用量会会落到一个稳定的水平；
c) 反之如果代码中存在没有释放对象引用的情况，则 data object 的 Total Size 值在每次 GC 后不会有明显的回落，随着操作次数的增多 Total Size 的值会越来越大，
直到到达一个上限后导致进程被 kill 掉。
d) 此处已 system_process 进程为例，在我的测试环境中 system_process 进程所占用的内存的 data object 的 Total Size 正常情况下会稳定在 2.2~2.8 之间，而当其值超过 3.55 后进程就会被 kill。

总之，使用 DDMS 的 Heap 视图工具可以很方便的确认我们的程序是否存在内存泄漏的可能性。

2、内存分析

如何用 MAT 来分析，前提是 Android 开发和测试的工具安装完整，SDK，Eclipse：

1.打开 Eclipse

2.选择 Help->Install New Software;

3.在 Work with 中添加站点：http://download.eclipse.org/mat/1.0/update-site/(这个地址可能会变化，但是新的地址可以在官方网站上找到：http://www.eclipse.org/mat/downloads.php)

4.生成.hprof 文件:

插入 SD 卡(Android 机器很多程序都需要插入 SD 卡)，并将设备连接到 PC，在 Eclipse 中的 DDMS 中选择要测试的进程，然后点击 Update Heap 和 Dump HPROF file 两个 Button。

.hprof 文件会自动保存在 SD 卡上，把 .hprof 文件拷贝到 PC 上的\ android-sdk-windows\tools 目录下。这个由 DDMS 生成的文件不能直接在 MAT 打开，需要转换。

运行 cmd 打开命令行，cd 到\ android-sdk-windows\tools 所在目录，并输入命令 hprof-conv xxxxx.hprof yyyyy.hprof，其中 xxxxx.hprof 为原始文件，yyyyy.hprof 为转换过后的文件。转换过后的文件自动放在 android-sdk-windows\tools 目录下。

OK，到此为止，.hprof 文件处理完毕，可以用来分析内存泄露情况了。

5.打开 MAT：

在 Eclipse 中点击 Windows->Open Perspective->Other->Memory Analysis

6.导入.hprof 文件

在 MAT 中点击 File->Open File,浏览到刚刚转换而得到的.hprof 文件，并 Cancel 掉自动生成报告,点击 Dominator Tree，并按 Package 分组，选择自己所定义的 Package 类点右键，在弹出菜单中选择 List objects->With incoming references。

这时会列出所有可疑类，右键点击某一项，并选择 Path to GC Roots->exclude weak/soft references,会进一步筛选出跟程序相关的所有有内存泄露的类。据此，可以追踪到代码中的某一个产生泄露的类。

70、避免内存泄露

1.资源对象没关闭造成的内存泄漏

描述：

资源性对象比如（Cursor，File 文件等）往往都用了一些缓冲，我们在不使用的时候，应该及时关闭它们，以便它们的缓冲及时回收内存。它们的缓冲不仅存在于 java 虚拟机内，还存在于 java 虚拟机外。如果我们仅仅是把它的引用设置为 null,而不关闭它们，往往会造成内存泄漏。因为有些资源性对象，比如 SQLiteCursor（在析构函数 finalize(),如果我们没有关闭它，它自己会调 close()关闭），如果我们没有关闭它，系统在回收它时也会关闭它，但是这样的效率太低了。因此对于资源性对象在不使用的时候，应该调用它的 close()函数，将其关闭掉，然后才置为 null.在我们的程序退出时一定要确保我们的资源性对象已经关闭。

程序中经常会进行查询数据库的操作，但是经常会有使用完毕 Cursor 后没有关闭的情况。如果我们的查询结果集比较小，对内存的消耗不容易被发现，只有在常时间大量操作的情况下才会复现内存问题，这样就会给以后的测试和问题排查带来困难和风险。

示例代码：

```
Cursor cursor = getContentResolver().query(uri...);
if (cursor.moveToNext()) {
   ... ...
}
```

修正为:

```
Cursor cursor = null;
```

```
try {
   cursor = getContentResolver().query(uri...);
   if (cursor != null &&cursor.moveToNext()) {
         ... ...
   }
} finally {
   if (cursor != null) {
         try {
               cursor.close();
         } catch (Exception e) {
               //ignore this
         }
    }
}
```

2.构造 Adapter 时，没有使用缓存的 convertView

描述：

以构造 ListView 的 BaseAdapter 为例，在 BaseAdapter 中提供了方法：

public View getView(int position, ViewconvertView, ViewGroup parent)

来向 ListView 提供每一个 item 所需要的 view 对象。初始时 ListView 会从 BaseAdapter 中根据当前的屏幕布局实例化一定数量的 view 对象，同时 ListView 会将这些 view 对象缓存起来。当向上滚动 ListView 时，原先位于最上面的 list item 的 view 对象会被回收，然后被用来构造新出现的最下面的 list item。这个构造过程就是由 getView()方法完成的，getView()的第二个形参 View convertView 就是被缓存起来的 list item 的 view 对象(初始化时缓存中没有 view 对象则 convertView 是 null)。由此可以看出，如果我们不去使用 convertView，而是每次都在 getView()中重新实例化一个 View 对象的话，即浪费资源也浪费时间，也会使得内存占用越来越大。ListView 回收 list item 的 view 对象的过程可以查看:

android.widget.AbsListView.java --> voidaddScrapView(View scrap) 方法。

示例代码：

```
public View getView(int position, ViewconvertView, ViewGroup parent) {
   View view = new Xxx(...);
   ... ...
   return view;
}
```

修正为：

```
public View getView(int position, ViewconvertView, ViewGroup parent) {
   View view = null;
   if (convertView != null) {
   view = convertView;
   populate(view, getItem(position));
   ...
   } else {
   view = new Xxx(...);
   ...
```

```
    }
    return view;
}
```

3.Bitmap 对象没有在使用结束时调用 recycle()释放内存

描述：

有时我们会手工的操作 Bitmap 对象，如果一个 Bitmap 对象比较占内存，当它不在被使用的时候，可以调用 Bitmap.recycle()方法回收此对象的像素所占用的内存，但这不是必须的，视情况而定。可以看一下代码中的注释：

```
/**
    * Free up the memory associated with this bitmap's pixels, and mark the
    * bitmap as "dead", meaning it will throw an exception if getPixels() or
    * setPixels() is called, and will draw nothing. This operation cannot be
    * reversed, so it should only be called if you are sure there are no
    * further uses for the bitmap. This is an advanced call, and normally need
    * not be called, since the normal GCprocess will free up this memory when
    * there are no more references to this bitmap.
    */
```

4.试着使用关于 application 的 context 来替代和 activity 相关的 context

这是一个很隐晦的内存泄漏的情况。有一种简单的方法来避免 context 相关的内存泄漏。最显著地一个是避免 context 逃出他自己的范围之外。使用 Application context。这个 context 的生存周期和你的应用的生存周期一样长，而不是取决于 activity 的生存周期。如果你想保持一个长期生存的对象，并且这个对象需要一个 context,记得使用 application 对象。你可以通过调用 Context.getApplicationContext() or Activity.getApplication()来获得。更多的请看这篇文章如何避免 Android 内存泄漏。

参加如下详细分析：

至少在 T-Mobile G1 上 Android 应用在堆上分配的内存大小被限制 16MB 以内。对于手机来说，这是个不小的内存，但是这仍然远远不能满足一些开发者的需求。但是，即使你不打算使用所有的内存空间，你也应该尽可能地少用内存，从而使得其他应用能够运行而不是被杀掉。因为 Android 能够在内存中保持的应用越多，那么用户切换应用的速度就会越快。作为我工作的一部分，我在做 android 应用开发的时候也会陷入内存泄漏的问题中，大多数时候内存的泄漏都是由于犯了相同的错误：长期持有了一个 Context 的引用。

Android 上 ，Context 可以用于很多操作，但是大部分时候是用来加载以及使用资源。这就是为什么所有的 widgets 在他们的构造函数中接受一个 Context 参数。在一般的 android 应用中，你通常有两种 Context：分别是 Activity 和 Application。通常的，当我们的类和方法需要使用到 context 时，我们传递的是 Activity 这个 context：

```
@Override
protected void onCreate(Bundle state) {
    super.onCreate(state);

    TextView label = new TextView(this);
```

```
    label.setText("Leaks are bad");

    setContentView(label);
}
```

这意味着 views 拥有一个对整个 activity 的引用，也就是引用了你的 activity 所拥有的一切；通常的，这指的是完整的视图层级结构以及所有它的资源。因此，如果你泄露了一个 Context(“ 泄漏 ”意味着你保持着它的一个引用，从而使它不能被垃圾回收机制回收)，就意味着你泄漏了很多的内存。如果你不小心， 泄漏一 个 activity 的所有资源真的非常容易。

当屏幕的方向发生改变的时候，系统默认将会销毁当前的 activity 并且创建一个新的 activity 同时保持着原有的状态。在做这个的时候，Android 会从资源重新加载应用的 UI。现在，想象一下你写了一个应用，这个应用有一张很大的 bitmap。你不想再每一次旋转的时候重新加载它。最简单的方法让 bitmap 持续作用而不随每一个方向而重新加载 ，就是把它放进一个静态域：

```
private static Drawable sBackground;

@Override
protected void onCreate(Bundle state) {
    super.onCreate(state);

    TextView label = new TextView(this);
    label.setText("Leaks are bad");

    if (sBackground == null) {
        sBackground = getDrawable(R.drawable.large_bitmap);
    }
    label.setBackgroundDrawable(sBackground);

    setContentView(label);
}
```

这段代码很快，但是错误也很严重：它泄漏了第一个 activity，这个在第一次屏幕改变时被创建的 activity。当一个 Drawable 被关联到一个 view 上，这个 view 就相当于在 drawable 上设置的一个回调。在上面的代码片段中，这表示 drawable 有一个 TextView 的引用，而这个 TextView 又拥有一个 activity 的引用(Context)，activity 依次引用了几乎所有的东西(取决于你的代码)。

这个例子展示了一个最简单的 Context 泄漏的情况，你可以在 Home screen 的源码中看到我们是如何解决这个问题的(查找 unbindDrawables() 方法) ，这就是当 activity 被销毁的时候将 drawables 的回调设为 null 。有趣的是，你可能创造出一系列 context 泄漏的情况有很多，这非常糟糕。他们会是你很快内存溢出。

有两种简单的方法来避免 context 相关的内存泄漏。最显著地一个是避免 context 逃出他自己的范围之外。上面的例子就展示了使用静态引用的情况，而内部类和他们对外部类的

的隐式引用也是同样危险的。第二种方法是使用 Application context 。这个 context 的生存周期和你的应用的生存周期一样长，而不是取决于 activity 的生存周期。如果你想保持一个长期生存的对象，并且这个对象需要一个 context ，记得使用 application 对象。你可以通过调用 Context.getApplicationContext() or Activity.getApplication() 来获得。

总而言之，想要避免 context 相关的内存泄漏 ，记住以下几点：

- 不要对 activity 的 context 长期引用(一个 activity 的引用的生存周期应该和 activity 的生命周期相同)
- 试着使用关于 application 的 context 来替代和 activity 相关的 context
- 如果一个 acitivity 的非静态内部类的生命周期不受控制，那么避免使用它；使用一个静态的内部类并且对其中的 activity 使用一个弱引用。解决这个问题的方法是使用一个静态的内部类，并且对它的外部类有一 WeakReference，就像在 ViewRoot 中内部类 W 所做的就是这么个例子。
- 垃圾回收器不能处理内存泄漏的保障。

5. 释放对象的引用

示例 A：

假设有如下操作

```
public class DemoActivity extends Activity {
   ... ...
   private Handler mHandler = ...
   private Object obj;
   public void operation() {
   obj = initObj();
   ...
   [Mark]
   mHandler.post(new Runnable() {
   public void run() {
   useObj(obj);
   }
   });
   }
}
```

我们有一个成员变量 obj，在 operation()中我们希望能够将处理 obj 实例的操作 post 到某个线程的 MessageQueue 中。在以上的代码中，即便是 mHandler 所在的线程使用完了 obj 所引用的对象，但这个对象仍然不会被垃圾回收掉，因为 DemoActivity.obj 还保有这个对象的引用。 所以如果在 DemoActivity 中不再使用这个对象了，可以在[Mark]的位置释放对象的引用，而代码可以修改为：

```
... ...
public void operation() {
   obj = initObj();
   ...
   final Object o = obj;
```

```
        obj = null;
        mHandler.post(new Runnable() {
        public void run() {
        useObj(o);
        }
        }
    }
```

再如注册没取消造成的内存泄漏
一些 Android 程序可能引用我们的 Android 程序的对象（比如注册机制）。即使我们的 Android 程序已经结束了，但是别的引用程序仍然还有对我们的 Android 程序的某个对象的引用，泄漏的内存依然不能被垃圾回收。调用 registerReceiver 后未调用 unregisterReceiver。(会出现程序异常终止)

比如:假设我们希望在锁屏界面(LockScreen)中，监听系统中的电话服务以获取一些信息(如信号强度等)，则可以在 LockScreen 中定义一个 PhoneStateListener 的对象，同时将它注册到 TelephonyManager 服务中。对于 LockScreen 对象，当需要显示锁屏界面的时候就会创建一个 LockScreen 对象，而当锁屏界面消失的时候 LockScreen 对象就会被释放掉。

但是如果在释放 LockScreen 对象的时候忘记取消我们之前注册的 PhoneStateListener 对象，则会导致 LockScreen 无法被垃圾回收。如果不断的使锁屏界面显示和消失，则最终会由于大量的 LockScreen 对象没有办法被回收而引起 OutOfMemory,使得 system_process 进程挂掉。

虽然有些系统程序，它本身好像是可以自动取消注册的（当然不及时），但是我们还是应该在我们的程序中明确的取消注册，程序结束时应该把所有的注册都取消掉。

6.集合中对象没清理造成的内存泄漏

我们通常把一些对象的引用加入到了集合中，当我们不需要该对象时，并没有把它的引用从集合中清理掉，这样这个集合就会越来越大。如果这个集合是 static 的话，那情况就更严重了。

71、屏蔽 Home 键

```
public void onAttachedToWindow() {
this.getWindow().setType(WindowManager.LayoutParams.TYPE_KEYGUARD_DIALOG);
        super.onAttachedToWindow();
}
```

72、onTouch 和 onClick 事件

Button 的 onTouch，onClick，onLongClick 事件发生先后顺序和关联：

1、onTouch 返回 false
首先是 onTouch 事件的 down 事件发生，此时，如果长按，触发 onLongClick 事件；
然后是 onTouch 事件的 up 事件发生，up 完毕，最后触发 onClick 事件。

2、onTouch 返回 true
首先是 onTouch 事件的 down 事件发生，然后是 onTouch 事件的 up 事件发生；期间不触发 onClick 和 onLongClick 事件

4、 onTouch：down 返回 true，up 返回 false：
结果同二。
机制分析：
onTouch 事件中：down 事件返回值标记此次事件是否为点击事件（返回 false，是点击事件；返回 true，不记为点击事件），而 up 事件标记此次事件结束时间，也就是判断是否为长按。
只要当 down 返回 true 时候，系统将不把本次事件记录为点击事件，也就不会触发 onClick 或者 onLongClick 事件了。因此尽管当 up 的时候返回 false，系统也不会继续触发 onClick 事件了。

5、onTouch：down 返回 false，up 返回 true：
首先是 onTouch 事件的 down 事件发生，此时：
短按，触发 onLongClick 事件，然后是 onTouch 事件的 up 事件发生，完毕。
长按，先触发 onTouch 的 up 事件， 到一定时间后，自动触发 onLongClick 事件。
机制分析：
onTouch 事件中：down 事件返回值标记此次事件是否为点击事件（返回 false，是点击事件；返回 true，不记为点击事件），而 up 事件标记此次事件结束时间，也就是判断是否为长按。
当 down 返回 false，标记此次事件为点击事件，而 up 返回了 true，则表示此次事件一直没有结束，也就是一直长按下去了，达到长按临界时间后，自然触发长按事件，而 onClick 事件没有触发到

73、监听某个数据表

定义 ContentObserver，监听某个数据表
定义 ContentObserver，监听某个数据表

```
private ContentObserver mDownloadsObserver = new DownloadsChangeObserver(Downloads.CONTENT_URI);private class DownloadsChangeObserver extends ContentObserver {
        public DownloadsChangeObserver(Uri uri) {
            super(new Handler());
         }
         @Override
        public void onChange(boolean selfChange) {
        }
```

74、设置闹钟，响应闹钟

1、建立 Intent 和 Pending 来调用目标组件
Intent intent = new Intent(this, AlarmReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(this, 0, intent, 0);
2、获得闹钟管理实例
AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
3、设置单词闹钟：
alarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + (5*1000), pendingIntent);
4、 设置重复闹钟：
alarmManager.setRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + (10*1000), (24*60*60*1000), pendingIntent);//第二个参数是延迟时间，第三个参数是间隔时间

5、响应闹钟的接收器，记得在 manifest 中列出
public static class AlarmReceiver extends BroadcastReceiver {
@Override
public void onReceive(Context context, Intent intent) {
Toast.makeText(context, "闹钟提示：时间到！", Toast.LENGTH_LONG).show();
}
}

5、 Alarm 的取消：
Intent intent =new Intent(Main.this, alarmreceiver.class);
intent.setAction("repeating");
PendingIntent sender=PendingIntent.getBroadcast(Main.this, 0, intent, 0);
AlarmManager alarm=(AlarmManager)getSystemService(ALARM_SERVICE);
alarm.cancel(sender);
其中需要注意的是取消的 Intent 必须与启动 Intent 保持绝对一致才能支持取消 AlarmManager

74、IP 地址

1、获得 IP

方法 1、wifi 连接的情况下
WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
WifiInfo wifiInfo = wifiManager.getConnectionInfo();
int ipAddress = wifiInfo.getIpAddress();
String ip = intToIp(ipAddress);

```
public String intToIp(int i) {
    return ((i >> 24 ) & 0xFF ) + "." +
                  ((i >> 16 ) & 0xFF) + "." +
                  ((i >> 8 ) & 0xFF) + "." +
                  ( i & 0xFF) ;
}
```

方法 2、wifi 或 3G

```
public String getLocalIpAddress() {
   try {
      for (Enumeration<NetworkInterface> en = NetworkInterface.getNetworkInterfaces();
en.hasMoreElements();) {
         NetworkInterface intf = en.nextElement();
         for (Enumeration<InetAddress> enumIpAddr = intf.getInetAddresses();
enumIpAddr.hasMoreElements();) {
            InetAddress inetAddress = enumIpAddr.nextElement();
            if (!inetAddress.isLoopbackAddress()) {
            return inetAddress.getHostAddress().toString();
         }
      }
   }
   } catch (SocketException ex) {
         Log.e(LOG_TAG, ex.toString());
   }
   return null;
}
```

2、设置 IP

```
    private void setIp() {

        android.provider.Settings.System.putString(getContentResolver(),
                android.provider.Settings.System.WIFI_USE_STATIC_IP, "0");
        android.provider.Settings.System.putString(getContentResolver(),
                android.provider.Settings.System.WIFI_STATIC_IP,
                "192.168.1.111");

        android.provider.Settings.System.putString(getContentResolver(),
                android.provider.Settings.System.WIFI_STATIC_DNS1,
                "192.168.0.2");
        android.provider.Settings.System.putString(getContentResolver(),
                android.provider.Settings.System.WIFI_STATIC_DNS2,
                "192.168.0.3");
```

```
            android.provider.Settings.System.putString(getContentResolver(),
                    android.provider.Settings.System.WIFI_STATIC_GATEWAY,
                    "192.168.0.1");
            android.provider.Settings.System.putString(getContentResolver(),
                    android.provider.Settings.System.WIFI_STATIC_NETMASK,
                    "255.255.255.0");
            android.provider.Settings.System.putString(getContentResolver(),
                    android.provider.Settings.System.WIFI_STATIC_IP, "1");
        }
```

75、判断 Intent 是否可用

```
public static boolean isIntentAvailable(Context context, Intent intent) {
        final PackageManager packageManager = context.getPackageManager();
        List<ResolveInfo> list = packageManager.queryIntentActivities(intent,
                PackageManager.GET_ACTIVITIES);
        return list.size() > 0;
}
```

76、软件更换皮肤

在实现程序功能的同时，如果能让程序界面更加美观，有锦上添花之妙.
先说思路:
1、皮肤也就是相关的资源文件单独放置在某个工程中,一种皮肤一个工程文件.一个工程包括 N 多的资源文件,多个工程间资源的关系是,文件名,资源 ID 等完全一样.不同的可能是图片资源,style 等的设置不一样.
2)如:皮肤工程在 AndroidManifest.xml 中配置 android:sharedUserId="com.eric.skinmain".
表明允许 com.eric.skinmain 访问本工程中的资源文件. com.eric.skinmain 是主项目的包名
3、主项目通过
this.createPackageContext("com.eric.blackskin",Context.CONTEXT_IGNORE_SECURITY);
获取到 com.eric.blackskin 对应的 Context,然后通过返回的 context 对象就可以访问到 com.eric.blackskin 中的任何资源,如同访问自身的资源一样.

源码如下:

```
public class main extends Activity {
        /** Called when the activity is first created. */
        private LinearLayout showBg;
        private Button btn;
        private Context green_skin_Context = null;
```

```
    private Context black_skin_Context = null;
    int flag = 0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        showBg = (LinearLayout) findViewById(R.id.linear_layout_1);
        try {
            green_skin_Context = this.createPackageContext(
                    "com.eric.greenskin", Context.CONTEXT_IGNORE_SECURITY);
        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }
        try {
            black_skin_Context = this.createPackageContext(
                    "com.eric.blackskin", Context.CONTEXT_IGNORE_SECURITY);
        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }
        btn = (Button) findViewById(R.id.btn_change_skin);
        btn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                if (flag == 0) {
                    showBg.setBackgroundDrawable(green_skin_Context
                            .getResources().getDrawable(R.drawable.bg));
                    btn.setBackgroundDrawable(green_skin_Context.getResources()
                            .getDrawable(R.drawable.btn_normal));
                    flag = 1;
                } else if (flag == 1) {
                    showBg.setBackgroundDrawable(black_skin_Context
                            .getResources().getDrawable(R.drawable.bg));
                    btn.setBackgroundDrawable(black_skin_Context.getResources()
                            .getDrawable(R.drawable.btn_normal));
                    flag = 0;
                }
            }
        });
    }
}
```

77、禁止软件盘自动弹出

在 manifest 中对应的 Activity 里添加：
android:windowSoftInputMode="adjustUnspecified|stateHidden"
此方法在 edittext 被点击获得焦点后，仍会弹起键盘。

78、EditText 设置最大宽度

EditText 可以通过 Android:maxLength 属性来限制输入的长度,但这是按照 UNICODE 来算的,当中英文混合时,想要限制输入长度为 N 个字符时就要通过 InputFilter 来实现了.

```
InputFilter inputFilter = new InputFilter() {
@Override
public CharSequence filter(CharSequence source, int start, int end, Spanned dest,
int dstart, int dend) {
// TODO Auto-generated method stub
try {
//转换成中文字符集的长度
int destLen = dest.toString().getBytes(“GB18030″).length;
int sourceLen = source.toString().getBytes(“GB18030″).length;
Log.e(“filter”, String.valueOf(destLen + sourceLen));
//如果超过 100 个字符
if (destLen + sourceLen > 100) {
return “”;
}
//如果按回退键
if (source.length() < 1 && (dend － dstart >= 1)) {
return dest.subSequence(dstart, dend － 1);
}

//其他情况直接返回输入的内容
return source;
} catch (UnsupportedEncodingException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
return “”;
}
};
editText.setFilters(new InputFilters[] {inputFilter});
```

79、搭建流媒体服务器

最近开发，要用到流媒体服务器进行测试，虽然搜了一些网上提供的流媒体数据链接，但是总是不好用或者达不到自己的要求，又考虑到在服务器上搭建的成本过高，于是想在自己的 pc 机上搭建一个简单的能符合自己要求的容易操作的能满足自己项目需求的流媒体服务器。还好搭建成功，于是和兄弟们来分享下。

步骤：

1 http://dss.macosforge.org/downloads/DarwinStreamingSrvr5.5.5-Windows.exe 下载

2 下载后解压，会看到一个 Install.bat 的文件，直接运行它就会安装到 自己的目录。

3 安装 Perl 解释器， http://www.perl.org/get.html 这里下载

4 在 perl 目录下有一个 readme 文件，按照指示安装 perl

5 在 dos 下根据提示创建 WebAdmin 的账号和密码

C:\Program Files\Darwin Streaming Server>perl WinPasswdAssistant.pl

6 运行 WebAdmin 管理器(以后每次使用服务器的时候都需要用下边的命令来启动)

C:\Program Files\Darwin Streaming Server>perl streamingadminserver.pl

7 http://127.0.0.1:1220/ 来对流媒体服务器进行管理

在 DSS WebAdmin 里面修改 General Settings -> Media Directory ，将它改为你的 Media 目录，当然也可以使用默认的，把自己的视频文件放到 C:\Program Files\Darwin Streaming Server\Movies 里面

好了，基本上搭建完成了。

现在你可以使用自己写的 videoView 的程序或 VLC 打开 rtsp://127.0.0.1:554/sample_300kbit.mp4 来测试流媒体服务器的效果了。

在 DarwinStreamingSrvr5.5.5 的目录下有好几个格式的视频文件，这几个文件拷贝到你在第七步骤配置的目录里，就可以用了。

80、Bundle 传值重申

```
Bundle b = new Bundle();
b.putString(“”, );
```

80、获得 LayoutInflater 实例的三种方式

1.LayoutInflater inflater = getLayoutInflater(); //调用 Activity 的 getLayoutInflater()

2.LayoutInflater localinflater = (LayoutInflater) context.getSystemService (Context.LAYOUT_INFLATER_SERVICE);

3. LayoutInflater inflater = LayoutInflater.from(context);

其实，这三种方式本质是相同的，从源码中可以看出：

getLayoutInflater()：

Activity 的 getLayoutInflater() 方法是调用 PhoneWindow 的 getLayoutInflater()方法，看一下该源代码：

public PhoneWindow(Context context) { super(context); mLayoutInflater = LayoutInflater.from(context); }

可以看出它其实是调用 LayoutInflater.from(context)。

LayoutInflater.from(context)：

publicstatic LayoutInflater from(Context context) {

LayoutInflater LayoutInflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

if (LayoutInflater == null) {

thrownew AssertionError("LayoutInflater not found.");

}

return LayoutInflater;

}

可以看出它其实调用 context.getSystemService()。

81、获得屏幕像素的两种方法

其实是获得 DisplayMetrics 实例的两种方法：

方法 1：

```
DisplayMetrics metric = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metric);
```

方法 2：

```
DisplayMetrics metric = con.getResources().getDisplayMetrics();
```

获得宽高：（注意区分横竖屏状态）

```
displayWidth = metric.widthPixels;
displayHeight = metric.heightPixels;
```

82、ShowDialog（int id）;

利用 onCreateDialog 显示对话框的方式

1.重写 onCreateDialog（int id）

```
@Override
 protected Dialog onCreateDialog(int id) {
  Builder builder = null;
  switch (id) {
  case 1:
   builder = new AlertDialog.Builder(this)
              .setIcon(R.drawable.d_message)
              .setTitle("Message")
              .setMessage(R.string.runeEncrypt_message)
              .setNeutralButton("知道了", new DialogInterface.OnClickListener() {

              @Override
              public void onClick(DialogInterface dialog, int which) {
               // TODO Auto-generated method stub
               ShowView.back(RunEncoding.this);
              }
             });
   break;
  case 2:
   builder = new AlertDialog.Builder(RunEncoding.this)
    .setIcon(R.drawable.d_message)
   .setTitle("取消")
   .setMessage("您确实要取消吗")
   .setPositiveButton(R.string.positive, new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
     // TODO Auto-generated method stub
     cancelFlag = false;
    }
   }).
   setNegativeButton(R.string.negative, new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
     // TODO Auto-generated method stub

    }
   });
   break;
  default:
   break;
```

```
    }
    return builder.create();
  }
```

2，调用显示对话框 showDialog（id）；该方法会调用 onCreateDialog 方法来显示对话框

83、透明效果的实现

1、用 android 系统的透明效果

```
android:background="@android:color/transparent"
android:background="@android:color/transparent"
```

2、用 ARGB 来控制

```
半透明<Button android:background="#e0000000" />
透明<Button android:background="#00000000" />
半透明<Button android:background="#e0000000" />
透明<Button android:background="#00000000" />
```

3、设置 alpha

```
View v = findViewById(R.id.content);//找到你要设透明背景的 layout 的 id
v.getBackground().setAlpha(100);//透明度 0~255 透明度值 ， 值越小越透明
```

84、根据网络或 GPS 获得经纬度

1、所需权限：

```
<!-- 连接互联网 Internet 权限 -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- GPS 定位权限 -->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

2、代码：

```
    private void getLocation() {
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);

        // 从 GPS 获得经纬度
```

```
        if (locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
            Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if (location != null) {
                latitude = location.getLatitude();
                longitude = location.getLongitude();
                Log.e("Map", "Location GPS : Lat: " + latitude + " Lng: " + longitude);
            }
        } else {
            //注册监听
            LocationListener locationListener = new LocationListener() {
                // Provider 的状态在可用、暂时不可用和无服务三个状态直接切换时触发此函数
                @Override
                public void onStatusChanged(String provider, int status, Bundle extras) {

                }
                // Provider 被 enable 时触发此函数，比如 GPS 被打开
                @Override
                public void onProviderEnabled(String provider) {

                }
                // Provider 被 disable 时触发此函数，比如 GPS 被关闭
                @Override
                public void onProviderDisabled(String provider) {

                }
                // 当坐标改变时触发此函数，如果 Provider 传进相同的坐标，它就不会被触发
                @Override
                public void onLocationChanged(Location location) {
                    if (location != null) {
                        Log.e("Map", "Location changed : Lat: " + location.getLatitude() + " Lng: " + location.getLongitude());
                    }
                }
            };

    locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 1000, 0, locationListener);
            // 从网络获得经纬度
            Location location = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
```

```
            if (location != null) {
                latitude = location.getLatitude(); //  经度
                longitude = location.getLongitude(); //  纬度
                Log.e("Map", "Location NET : Lat: " + latitude + " Lng: " + longitude);
            }
        }
    }
```

85、TextView

在 xml 文件中使用 android:textStyle="bold" 可以将英文设置成粗体，但是不能将中文设置成粗体，将中文设置成粗体的方法是：

```
TextView tv = (TextView)findViewById(R.id.TextView01);
TextPaint tp = tv.getPaint();
tp.setFakeBoldText(true);
```

其他还有：

```
textView.setTextSize(TypedValue.COMPLEX_UNIT_SP, 24f);//设置成 24sp
textView.setTypeface(Typeface.defaultFromStyle(Typeface.BOLD));//可能中文加粗无效
textView.setTypeface(Typeface.defaultFromStyle(Typeface.ITALIC));//可能中文无效
textView.setText(Html.fromHtml("<u>"+texts+"</u>"));//下划线
textView.setTypeface(Typeface.MONOSPACE,Typeface.ITALIC);//斜体，中文有效
textView.getPaint().setFlags(Paint. STRIKE_THRU_TEXT_FLAG );//中间加横线
textView.getPaint().setFlags(Paint. UNDERLINE_TEXT_FLAG );//底部加横线
```

90、获取存储卡和手机内部存储空间

该代码片段可以让我们获取 internal 和 external 的存储空间大小。

```
import java.io.File;
import android.os.Environment;
import android.os.StatFs;

public class StorageUtil {
    private static final int ERROR = -1;

    //存储卡是否已经挂载
    public static boolean externalMemoryAvailable() {
        return android.os.Environment.getExternalStorageState().equals(
                android.os.Environment.MEDIA_MOUNTED);
    }
    //内部存储的可用空间 B
    public static long getAvailableInternalMemorySize() {
```

```
            File path = Environment.getDataDirectory();
            StatFs stat = new StatFs(path.getPath());
            long blockSize = stat.getBlockSize();
            long availableBlocks = stat.getAvailableBlocks();
            return availableBlocks * blockSize;
        }
        //内部存储空间总大小 B
        public static long getTotalInternalMemorySize() {
            File path = Environment.getDataDirectory();
            StatFs stat = new StatFs(path.getPath());
            long blockSize = stat.getBlockSize();
            long totalBlocks = stat.getBlockCount();
            return totalBlocks * blockSize;
        }
        //存储卡可用空间大小 B
        public static long getAvailableExternalMemorySize() {
            if (externalMemoryAvailable()) {
                File path = Environment.getExternalStorageDirectory();
                StatFs stat = new StatFs(path.getPath());
                long blockSize = stat.getBlockSize();
                long availableBlocks = stat.getAvailableBlocks();
                return availableBlocks * blockSize;
            } else {
                return ERROR;
            }
        }
        //存储卡空间总大小 B
        public static long getTotalExternalMemorySize() {
            if (externalMemoryAvailable()) {
                File path = Environment.getExternalStorageDirectory();
                StatFs stat = new StatFs(path.getPath());
                long blockSize = stat.getBlockSize();
                long totalBlocks = stat.getBlockCount();
                return totalBlocks * blockSize;
            } else {
                return ERROR;
            }
        }
    }
```

附录：

1、各种权限的说明

```
<uses-permission
android:name="android.permission.ACCESS_CHECKIN_PROPERTIES" />
```
允许读写访问"properties"表在checkin数据库中，改值可以修改上传

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```
允许一个程序访问CellID或WiFi热点来获取粗略的位置

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
```
允许一个程序访问精良位置(如GPS)

```
<uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
```
允许应用程序访问额外的位置提供命令

```
<uses-permission
android:name="android.permission.ACCESS_MOCK_LOCATION" />
```
允许程序创建模拟位置提供用于测试

```
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
```
允许程序访问有关GSM网络信息

```
<uses-permission
android:name="android.permission.ACCESS_SURFACE_FLINGER" />
```
允许程序使用SurfaceFlinger底层特性

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
/>
```
允许程序访问Wi-Fi网络状态信息

```
<uses-permission android:name="android.permission.ADD_SYSTEM_SERVICE"
/>
```
允许程序发布系统级服务

```
<uses-permission android:name="android.permission.BATTERY_STATS" />
```
允许程序更新手机电池统计信息

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```
允许程序连接到已配对的蓝牙设备

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```
允许程序发现和配对蓝牙设备

```
<uses-permission android:name="android.permission.BRICK" />
```
请求能够禁用设备

```
<uses-permission
android:name="android.permission.BROADCAST_PACKAGE_REMOVED" />
```
允许程序广播一个提示消息在一个应用程序包已经移除后

```
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
```
允许一个程序广播常用intents

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```
允许一个程序初始化一个电话拨号不需通过拨号用户界面需要用户确认

```
<uses-permission android:name="android.permission.CALL_PRIVILEGED" />
```
允许一个程序拨打任何号码，包含紧急号码无需通过拨号用户界面需要用户确认

```
<uses-permission android:name="android.permission.CAMERA" />
```
请求访问使用照相设备

```
<uses-permission
android:name="android.permission.CHANGE_COMPONENT_ENABLED_STATE" />
```
允许一个程序是否改变一个组件或其他的启用或禁用

```
<uses-permission
android:name="android.permission.CHANGE_CONFIGURATION" />
```
允许一个程序修改当前设置，如本地化

```
<uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
```
允许程序改变网络连接状态

```
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"
/>
```
允许程序改变Wi-Fi连接状态

```
<uses-permission android:name="android.permission.CLEAR_APP_CACHE" />
```
允许一个程序清楚缓存从所有安装的程序在设备中

```
<uses-permission
android:name="android.permission.CLEAR_APP_USER_DATA" />
```
允许一个程序清除用户设置

```
<uses-permission
android:name="android.permission.CONTROL_LOCATION_UPDATES" />
```
允许启用禁止位置更新提示从无线模块

```
<uses-permission android:name="android.permission.DELETE_CACHE_FILES"
/>
```
允许程序删除缓存文件

```
<uses-permission android:name="android.permission.DELETE_PACKAGES" />
```
允许一个程序删除包

```
<uses-permission android:name="android.permission.DEVICE_POWER" />
```
允许访问底层电源管理

```
<uses-permission android:name="android.permission.DIAGNOSTIC" />
```
允许程序RW诊断资源

```
<uses-permission android:name="android.permission.DISABLE_KEYGUARD" />
```
允许程序禁用键盘锁

```
<uses-permission android:name="android.permission.DUMP" />
```
允许程序返回状态抓取信息从系统服务

```
<uses-permission
android:name="android.permission.EXPAND_STATUS_BAR"/>
```
允许一个程序扩展收缩状态栏

```
<uses-permission android:name="android.permission.FACTORY_TEST"/>
```
作为一个工厂测试程序，运行在root用户

```
<uses-permission android:name="android.permission.FLASHLIGHT"/>
```
访问闪光灯

```
<uses-permission android:name="android.permission.FORCE_BACK"/>
```
允许程序强行一个后退操作是否在顶层activities

```
<uses-permission android:name="android.permission.FOTA_UPDATE"/>
```
一个预留权限

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

访问一个帐户列表在Accounts Service中

<uses-permission android:name="android.permission.GET_PACKAGE_SIZE"/>
允许一个程序获取任何package占用空间容量

<uses-permission android:name="android.permission.GET_TASKS"/>
允许一个程序获取信息有关当前或最近运行的任务，一个缩略的任务状态，是否活动等等

<uses-permission android:name="android.permission.HARDWARE_TEST"/>
允许访问硬件

<uses-permission android:name="android.permission.INJECT_EVENTS"/>
允许一个程序截获用户事件如按键、触摸、轨迹球等等到一个时间流

<uses-permission android:name="android.permission.INSTALL_PACKAGES"/>
允许一个程序安装packages

<uses-permission
android:name="android.permission.INTERNAL_SYSTEM_WINDOW"/>
允许打开窗口使用系统用户界面

<uses-permission android:name="android.permission.INTERNET"/>
允许程序打开网络套接字

<uses-permission
android:name="android.permission.MANAGE_APP_TOKENS"/>
允许程序管理(创建、催后、z- order默认向z轴推移)程序引用在窗口管理器中

<uses-permission android:name="android.permission.MASTER_CLEAR"/>
恢复出厂设置权限，清除一切用户数据

<uses-permission
android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
允许程序修改全局音频设置

<uses-permission
android:name="android.permission.MODIFY_PHONE_STATE"/>
允许修改话机状态，如电源，人机接口等

<uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
允许挂载和反挂载文件系统可移动存储

<uses-permission

```
android:name="android.permission.PERSISTENT_ACTIVITY"/>
```
允许一个程序设置他的activities显示

```
<uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS"/>
```
允许程序监视、修改有关播出电话

```
<uses-permission android:name="android.permission.READ_CALENDAR"/>
```
允许程序读取用户日历数据

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```
允许程序读取用户联系人数据

```
<uses-permission
android:name="android.permission.READ_FRAME_BUFFER"/>
```
允许程序屏幕波或和更多常规的访问帧缓冲数据

```
<uses-permission android:name="android.permission.READ_INPUT_STATE"/>
```
允许程序读取底层系统日志文件

```
<uses-permission android:name="android.permission.READ_OWNER_DATA"/>
```
允许程序读取所有者数据

```
<uses-permission android:name="android.permission.READ_SMS"/>
```
允许程序读取短信息

```
<uses-permission
android:name="android.permission.READ_SYNC_SETTINGS"/>
```
允许程序读取同步设置

```
<uses-permission android:name="android.permission.READ_SYNC_STATS"/>
```
允许程序读取同步状态

```
<uses-permission android:name="android.permission.REBOOT"/>
```
请求能够重新启动设备

```
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```
允许一个程序接收到 ACTION_BOOT_COMPLETED广播在系统完成启动

```
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
```
允许一个程序监控将收到MMS彩信,记录或处理

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

允许程序监控一个将收到短信息，记录或处理

<uses-permission android:name="android.permission.RECEIVE_WAP_PUSH"/>
允许程序监控将收到WAP PUSH信息

<uses-permission android:name="android.permission.RECORD_AUDIO"/>
允许程序录制音频

<uses-permission android:name="android.permission.REORDER_TASKS"/>
允许程序改变Z轴排列任务

<uses-permission android:name="android.permission.RESTART_PACKAGES"/>
允许程序重新启动其他程序

<uses-permission android:name="android.permission.SEND_SMS"/>
允许程序发送SMS短信

<uses-permission
android:name="android.permission.SET_ACTIVITY_WATCHER"/>
允许程序监控或控制activities已经启动全局系统中

<uses-permission
android:name="android.permission.SET_ALWAYS_FINISH"/>
允许程序控制是否活动间接完成在处于后台时

<uses-permission
android:name="android.permission.SET_ANIMATION_SCALE"/>
修改全局信息比例

<uses-permission android:name="android.permission.SET_DEBUG_APP"/>
配置一个程序用于调试

<uses-permission android:name="android.permission.SET_ORIENTATION"/>
允许底层访问设置屏幕方向和实际旋转

<uses-permission
android:name="android.permission.SET_PREFERRED_APPLICATIONS"/>
允许一个程序修改列表参数PackageManager.addPackageToPreferred() 和
PackageManager.removePackageFromPreferred()方法

<uses-permission
android:name="android.permission.SET_PROCESS_FOREGROUND"/>
允许程序当前运行程序强行到前台

```
<uses-permission
android:name="android.permission.SET_PROCESS_LIMIT"/>
```
允许设置最大的运行进程数量

```
<uses-permission android:name="android.permission.SET_TIME_ZONE"/>
```
允许程序设置时间区域

```
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
```
允许程序设置壁纸

```
<uses-permission
android:name="android.permission.SET_WALLPAPER_HINTS"/>
```
允许程序设置壁纸hits

```
<uses-permission
android:name="android.permission.SIGNAL_PERSISTENT_PROCESSES"/>
```
允许程序请求发送信号到所有显示的进程中

```
<uses-permission android:name="android.permission.STATUS_BAR"/>
```
允许程序打开、关闭或禁用状态栏及图标

```
<uses-permission
android:name="android.permission.SUBSCRIBED_FEEDS_READ"/>
```
允许一个程序访问订阅RSS Feed内容提供

```
<uses-permission
android:name="android.permission.SUBSCRIBED_FEEDS_WRITE"/>
```
系统暂时保留改设置

```
<uses-permission
android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
```
允许一个程序打开窗口使用 TYPE_SYSTEM_ALERT，显示在其他所有程序的顶层

```
<uses-permission android:name="android.permission.VIBRATE"/>
```
允许访问振动设备

```
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```
允许使用PowerManager的 WakeLocks保持进程在休眠时从屏幕消失

```
<uses-permission
android:name="android.permission.WRITE_APN_SETTINGS"/>
```
允许程序写入APN设置 (access point name)

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

允许一个程序写入但不读取用户日历数据

```
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
```
允许程序写入但不读取用户联系人数据

```
<uses-permission android:name="android.permission.WRITE_GSERVICES"/>
```
允许程序修改Google服务地图

```
<uses-permission android:name="android.permission.WRITE_OWNER_DATA"/>
```
允许一个程序写入但不读取所有者数据

```
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
```
允许程序读取或写入系统设置

```
<uses-permission android:name="android.permission.WRITE_SMS"/>
```
允许程序写短信

```
<uses-permission
android:name="android.permission.WRITE_SYNC_SETTINGS"/>
```
允许程序写入同步设置