# Android目录总结

**android系统的几个比较重要的目录**

/system 存放的是rom的信息；

/system/app 存放rom本身附带的软件即系统软件；

/system/data 存放/system/app 中核心系统软件的数据文件信息。

/data 存放的是用户的软件信息（非自带rom安装的软件）；

/data/app 存放用户安装的软件；

/data/data 存放所有软件（包括/system/app 和 /data/app 和 /mnt/asec中装的软件）的一些lib和xml文件等数据信息；

/data/dalvik-cache 存放程序的缓存文件，这里的文件都是可以删除的。

/mnt 目录，熟悉linux的人都清楚，linux默认挂载外部设备都会挂到这个目录下面去，如将sd卡挂载上去后，会生成一个/mnt/sdcard 目录。

/sdcard 目录，这是一个软链接（相当于windows的文件夹的快捷方式），链接到/mnt/sdcard 目录，即这个目录的内容就是sdcard的内容。

**外部存储external storage和内部存储internal storage**

**1.内部存储：**

注意内部存储不是内存。内部存储位于系统中很特殊的一个位置，如果你想将文件存储于内部存储中，那么文件默认只能被你的应用访问到，且一个应用所创建的所有文件都在和应用包名相同的目录下。也就是说应用创建于内部存储的文件，与这个应用是关联起来的。当一个应用卸载之后，内部存储中的这些文件也被删除。从技术上来讲如果你在创建内部存储文件的时候将文件属性设置成可读，其他app能够访问自己应用的数据，前提是他知道你这个应用的包名，如果一个文件的属性是私有（private），那么即使知道包名其他应用也无法访问。 内部存储空间十分有限，因而显得可贵，另外，它也是系统本身和系统应用程序主要的数据存储所在地，一旦内部存储空间耗尽，手机也就无法使用了。所以对于内部存储空间，我们要尽量避免使用。Shared Preferences和SQLite数据库都是存储在内部存储空间上的。内部存储一般用Context来获取和操作。一般来说，我们不会自己去操作内部存储空间，没有root权限的话，我们也没法操作内部存储空间，事实上内部存储主要是由系统来维护的

**2.外部存储：**

最容易混淆的是外部存储，如果说pc上也要区分出外部存储和内部存储的话，那么自带的硬盘算是内部存储，U盘或者移动硬盘算是外部存储，因此我们很容易带着这样的理解去看待安卓手机，认为机身固有存储是内部存储，而扩展的T卡是外部存储。比如我们任务16GB版本的Nexus 4有16G的内部存储，普通消费者可以这样理解，但是安卓的编程中不能，这16GB仍然是外部存储。

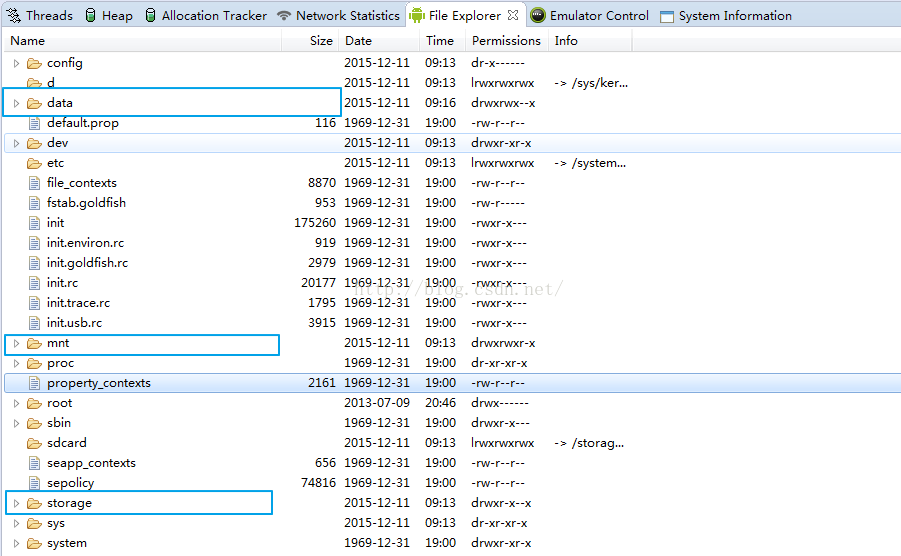
所有的安卓设备都有外部存储和内部存储，这两个名称来源于安卓的早期设备，那个时候的设备内部存储确实是固定的，而外部存储确实是可以像U盘一样移动的。但是在后来的设备中，很多中高端机器都将自己的机身存储扩展到了8G以上，他们将存储在概念上分成了"内部internal" 和"外部external" 两部分，但其实都在手机内部。所以不管安卓手机是否有可移动的sdcard，他们总是有外部存储和内部存储。最关键的是，我们都是通过相同的api来访问可移动的sdcard或者手机自带的存储（外部存储）。

外部存储虽然概念上有点复杂，但也很好区分，你把手机连接电脑，能被电脑识别的部分就一定是外部存储。

api中在介绍Environment.getExternalStorageDirectory()方法的时候说得很清楚：

don't be confused by the word "external" here. This directory can better be thought as media/shared storage. It is a filesystem that can hold a relatively large amount of data and that is shared across all applications (does not enforce permissions). Traditionally this is an SD card, but it may also be implemented as built-in storage in a device that is distinct from the protected internal storage and can be mounted as a filesystem on a computer.

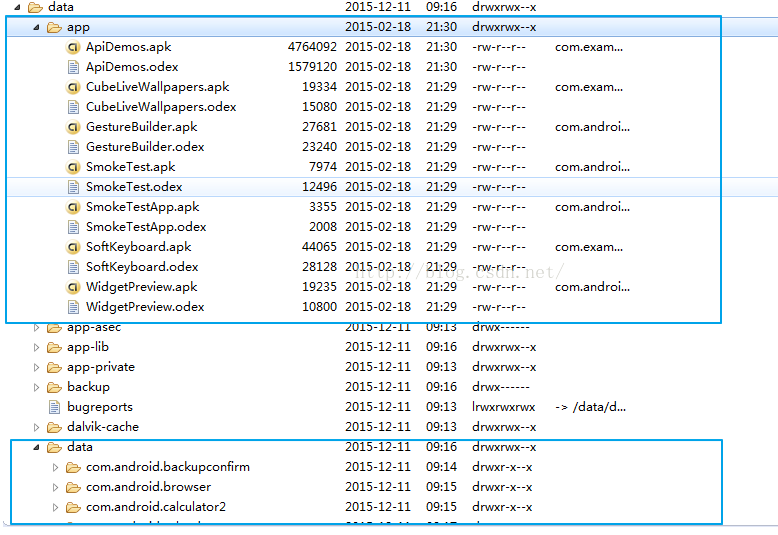
我们打开DDMS，有一个File Explorer，如下：

****

这里有三个文件夹需要我们重视，一个是data，一个是mnt，一个是storage，我们下面就详细说说这三个文件夹。

1.内部存储

data文件夹就是我们常说的内部存储，当我们打开data文件夹之后（没有root的手机不能打开该文件夹），里边有两个文件夹值得我们关注，如下：



一个文件夹是app文件夹，还有一个文件夹就是data文件夹，app文件夹里存放着我们所有安装的app的apk文件，其实，当我们调试一个app的时候，可以看到控制台输出的内容，有一项是uploading .....就是上传我们的apk到这个文件夹，上传成功之后才开始安装。另一个重要的文件夹就是data文件夹了，这个文件夹里边都是一些包名，打开这些包名之后我们会看到这样的一些文件：

1.data/data/包名/shared\_prefs

2.data/data/包名/databases

3.data/data/包名/files

4.data/data/包名/cache

如果打开过data文件，应该都知道这些文件夹是干什么用的，我们在使用sharedPreferenced的时候，将数据持久化存储于本地，其实就是存在这个文件中的xml文件里，我们App里边的数据库文件就存储于databases文件夹中，还有我们的普通数据存储在files中，缓存文件存储在cache文件夹中，存储在这里的文件我们都称之为内部存储。

2.外部存储

外部存储才是我们平时操作最多的，外部存储一般就是我们上面看到的storage文件夹，当然也有可能是mnt文件夹，这个不同厂家有可能不一样。

一般来说，在storage文件夹中有一个sdcard文件夹，这个文件夹中的文件又分为两类，一类是公有目录，还有一类是私有目录，其中的公有目录有九大类，比如DCIM、DOWNLOAD等这种系统为我们创建的文件夹，私有目录就是android这个文件夹，这个文件夹打开之后里边有一个data文件夹，打开这个data文件夹，里边有许多包名组成的文件夹。

使用Environment.getExternalStorageDirectory()获得sdcard根路径。

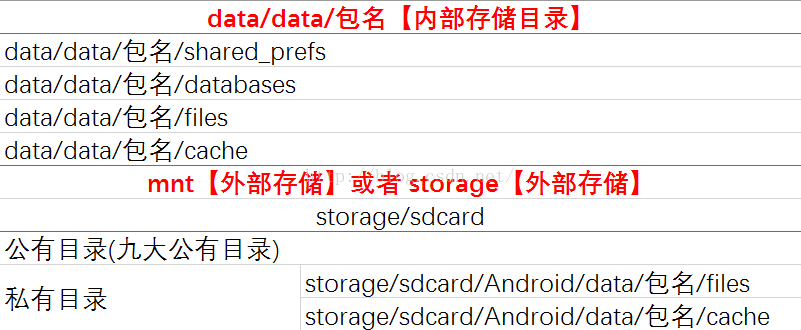
在使用外部存储之前，你必须要先检查外部存储的当前状态，以判断是否可用。

说到这里，我想大家应该已经可以分清楚什么是内部存储什么是外部存储了吧？好，分清楚之后我们就要看看怎么来操作内部存储和外部存储了。

3.操作存储空间

首先，经过上面的分析，大家已经明白了，什么是内部存储，什么是外部存储，以及这两种存储方式分别存储在什么位置，一般来说，我们不会自己去操作内部存储空间，没有root权限的话，我们也没法操作内部存储空间，事实上内部存储主要是由系统来维护的。不过在代码中我们是可以访问到这个文件夹的。由于内部存储空间有限，在开发中我们一般都是操作外部存储空间，Google官方建议我们App的数据应该存储在外部存储的私有目录中该App的包名下，这样当用户卸载掉App之后，相关的数据会一并删除，如果你直接在/storage/sdcard目录下创建了一个应用的文件夹，那么当你删除应用的时候，这个文件夹就不会被删除。

经过以上的介绍，我们可以总结出下面一个表格：



如果按照路径的特征，我们又可以将文件存储的路径分为两大类，一类是路径中含有包名的，一类是路径中不含有包名的，含有包名的路径，因为和某个App有关，所以对这些文件夹的访问都是调用Context里边的方法，而不含有包名的路径，和某一个App无关，我们可以通过Environment中的方法来访问。如下图：

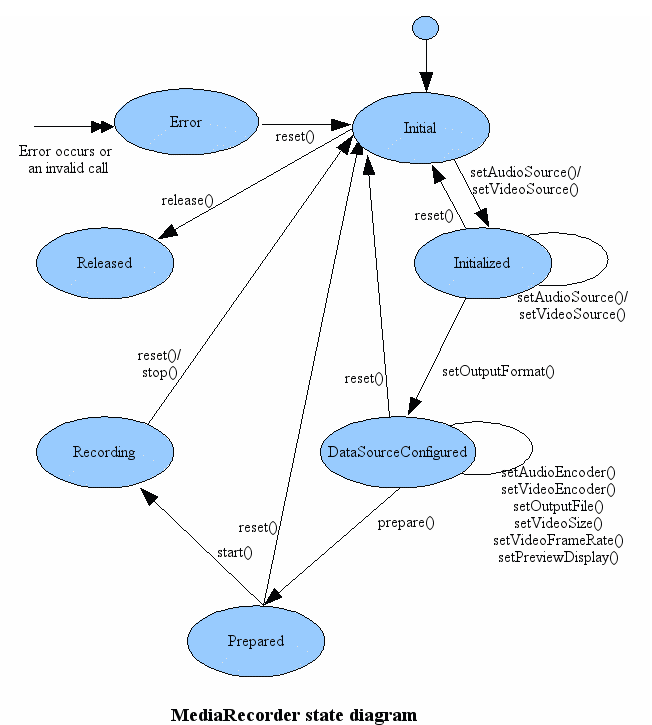


大家看到，有包名的路径我们都是调用Context中的方法来获得，没有包名的路径，我们直接调用Environment中的方法获得，那么其中有两个方法需要传入一个String类型的参数，这个参数我们使用了Environment中的常量，参数的意思是我们要访问这个路径下的哪个文件夹。

# Android视频录制总结

MediaRecorder类是Android sdk提供的一个专门用于音视频录制，一般利用手机麦克风采集音频，摄像头采集图片信息。

使用MediaRecorder类进行录制的过程可以用状态机进行表示，几个状态如下：



根据状态机图可知，我们需要安装一定的顺序去设置MediaRecorder的参数，在视频录制的时候需要使用到Camera类进行配合，除了调用其Camera.open()和Camera.release()来打开和释放相机外，还需要通过调用Camera.lock() 和Camera.unlock()从而允许MediaRecorder实例拿到Camera的使用权。

MediaRecorder使用时候的参数设定需要安装一定的顺序，如下：

打开相机，通过调用Camera.open()拿到一个相机的实例，可能是后置或者前置摄像机。

给相机设定一个预览的SurfaceView，Camera.setPreviewDisplay()，绑定绘制预览图像的surface。surface是指向屏幕窗口原始图像缓冲区（raw buffer）的一个句柄，通过它可以获得这块屏幕上对应的canvas，进而完成在屏幕上绘制View的工作。通过surfaceHolder可以将Camera和surface连接起来，当camera和surface连接后，camera获得的预览帧数据就可以通过surface显示在屏幕上了。

setPrameters设置相机参数，包括前后摄像头，闪光灯模式、聚焦模式、预览和拍照尺寸等。

开始图像预览：Camera.startPreview()，然后，接下来需要设置视频录制相关的参数，new一个MediaRecorder实例；

调用Camera.unlock() ，Unlock the Camera，从而允许MediaRecorder实例拿到Camera的使用权；

调用setCamera()，设置视频录制使用的camera

接下来需要设置录制中的一些参数，代码如下：

// 设置音频采集方式

mRecorder**.**setAudioSource**(**MediaRecorder**.**AudioSource**.**CAMCORDER**);**

//设置视频的采集方式

mRecorder**.**setVideoSource**(**MediaRecorder**.**VideoSource**.**CAMERA**);**

//设置文件的输出格式

mRecorder**.**setOutputFormat**(**MediaRecorder**.**OutputFormat**.**MPEG\_4**);**//aac\_adif， aac\_adts， output\_format\_rtp\_avp， output\_format\_mpeg2ts ，webm

//设置audio的编码格式

mRecorder**.**setAudioEncoder**(**MediaRecorder**.**AudioEncoder**.**AMR\_NB**);**

//设置video的编码格式

mRecorder**.**setVideoEncoder**(**MediaRecorder**.**VideoEncoder**.**H264**);**

//设置录制的视频编码比特率

mRecorder**.**setVideoEncodingBitRate**(**1024 **\*** 1024**);**

//设置录制的视频帧率,注意文档的说明:

mRecorder**.**setVideoFrameRate**(**30**);**

//设置要捕获的视频的宽度和高度

mSurfaceHolder**.**setFixedSize**(**640**,** 480**);**//最高只能设置640x480

mRecorder**.**setVideoSize**(**640**,** 480**);**//最高只能设置640x480

//设置记录会话的最大持续时间（毫秒）

mRecorder**.**setMaxDuration**(**60 **\*** 1000**);**

mRecorder**.**setOrientationHint**(**90**);**//默认的录制图像是横向的，设置一个提示的旋转角度，以便播放器进行视频方向调整

mRecorder**.**setPreviewDisplay**(**mSurfaceHolder**.**getSurface**());**

//设置输出文件的路径

mRecorder**.**setOutputFile**(**getSaveFile**().**getPath**());**

//准备录制

mRecorder**.**prepare**();**

//开始录制

mRecorder**.**start**();**

最后调用prepare()和start()开始录制，注意录制过程中看到的预览图像和实际录制的图像是没有什么联系的，并不是录制的就是预览看到的。

最后录制结束后需要，mRecorder.stop();并对mRecorder和camera进行资源释放。

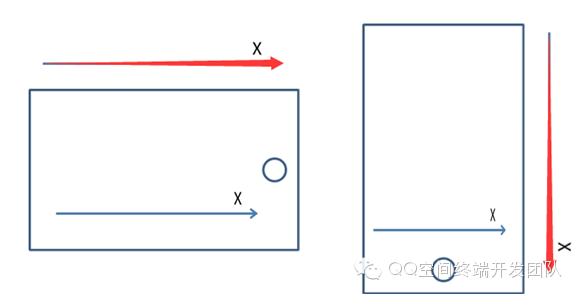
一些注意事项：

当Activity设为竖屏时，SurfaceView预览图像颠倒90度，这是因为在Android系统中，屏幕的左上角是坐标系统的原点（0,0）坐标。原点向右延伸是X轴正方向，原点向下延伸是Y轴正方向。

手机相机的图像数据都是来自于摄像头硬件的图像传感器，这个传感器在被固定到手机上后有一个默认的取景方向，如下图2所示，坐标原点位于手机横放时的左上角，即与横屏应用的屏幕X方向一致。换句话说，与竖屏应用的屏幕X方向呈90度角。



相机的预览方向：由于手机屏幕可以360度旋转，为了保证用户无论怎么旋转手机都能看到“正确”的预览画面（这个“正确”是指显示在UI预览界面的画面与人眼看到的眼前的画面是一致的），Android系统底层根据当前手机屏幕的方向对图像传感器采集到的数据进行了旋转处理，然后才送给显示系统，因此可以保证预览画面始终“正确”。在相机API中可以通过setDisplayOrientation()设置相机预览方向。在默认情况下，这个值为0，与图像传感器一致。因此对于横屏应用来说，由于屏幕方向和预览方向一致，预览图像不会颠倒90度。但是对于竖屏应用，屏幕方向和预览方向垂直，所以会出现颠倒90度现象。为了得到正确的预览画面，必须通过API将相机的预览方向旋转90，保持与屏幕方向一致，如图3所示。



相机的拍照方向：当点击拍照按钮，拍摄的照片是由图像传感器采集到的数据直接存储到SDCard上产生的，因此，相机的拍照方向与传感器方向是一致的。

官方推荐的设置预览方向的代码：

public static void setCameraDisplayOrientation**(**Activity activity**,**

int cameraId**,** Camera camera**)** **{**

Camera**.**CameraInfo info **=** **new** Camera**.**CameraInfo**();**

Camera**.**getCameraInfo**(**cameraId**,** info**);**

int rotation **=** activity**.**getWindowManager**().**getDefaultDisplay**()**

**.**getRotation**();**

int degrees **=** 0**;**

**switch** **(**rotation**)** **{**

**case** Surface**.**ROTATION\_0**:**

degrees **=** 0**;**

**break;**

**case** Surface**.**ROTATION\_90**:**

degrees **=** 90**;**

**break;**

**case** Surface**.**ROTATION\_180**:**

degrees **=** 180**;**

**break;**

**case** Surface**.**ROTATION\_270**:**

degrees **=** 270**;**

**break;**

**}**

int result**;**

**if** **(**info**.**facing **==** Camera**.**CameraInfo**.**CAMERA\_FACING\_FRONT**)** **{**

result **=** **(**info**.**orientation **+** degrees**)** **%** 360**;**

result **=** **(**360 **-** result**)** **%** 360**;**

**}** **else** **{**

result **=** **(**info**.**orientation **-** degrees **+** 360**)** **%** 360**;**

**}**

camera**.**setDisplayOrientation**(**result**);**

**}**

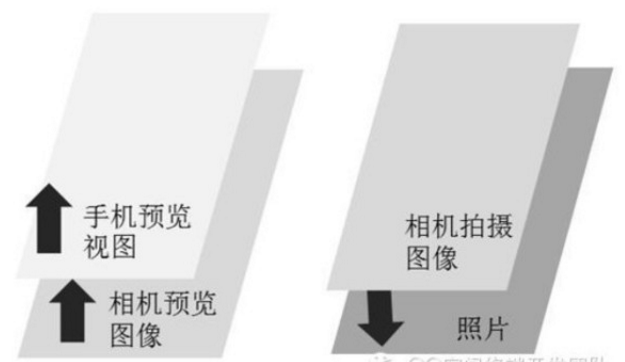
**SurfaceView预览图像、拍摄照片拉伸变形**

说明这个问题之前，同样先说一下几个跟相机有关的尺寸。

SurfaceView尺寸：即自定义相机应用中用于显示相机预览图像的View的尺寸，当它铺满全屏时就是屏幕的大小。这里surfaceview显示的预览图像暂且称作手机预览图像。

Previewsize：相机硬件提供的预览帧数据尺寸。预览帧数据传递给SurfaceView，实现预览图像的显示。这里预览帧数据对应的预览图像暂且称作相机预览图像。

Picturesize：相机硬件提供的拍摄帧数据尺寸。拍摄帧数据可以生成位图文件，最终保存成.jpg或者.png等格式的图片。这里拍摄帧数据对应的图像称作相机拍摄图像。图4说明了以上几种图像及照片之间的关系。手机预览图像是直接提供给用户看的图像，它由相机预览图像生成，拍摄照片的数据则来自于相机拍摄图像。



在开发过程中遇到的三种拉伸变形现象：

1、手机预览画面中物体被拉伸变形。

2、拍摄照片中物体被拉伸变形。

3、点击拍照瞬间，手机预览画面会停顿下，此时的图像是拉伸变形的，然后预览画面恢复后图像又正常了。

现象1的原因是SurfaceView和Previewsize的长宽比率不一致。因为手机预览视图的图像是由相机预览图像根据SurfaceView大小缩放得来的，当长宽比不一致时必然会导致图像变形。后两个现象的原因则是Previewsize和Picturesize的长宽比率不一致所致，查了相关的资料，发现其具体原因跟某些手机相机硬件的底层实现有关。总之为了避免以上几种变形现象的发生，在开发时最好将SurfaceView、PreviewSize、PictureSize三个尺寸保证长宽比例一致。具体实现可以先通过camera.getSupportedPreviewSizes()和camera.getSupportedPictureSizes()获得相机硬件支持的所有预览和拍摄尺寸，然后在里面筛选出和SurfaceView的长宽比一致并且大小合适的尺寸，通过camera.setPrameters来更新设置。注意：市场上手机相机硬件支持的尺寸一般都是主流的4:3或者16:9，所以SurfaceView尺寸不能太奇葩，最好也设置成这样的长宽比。

参考：

<https://developer.android.com/guide/topics/media/camera.html#capture-video>

<http://www.cnblogs.com/whoislcj/p/5583833.html>

<https://zhuanlan.zhihu.com/p/20559606>

<https://mp.weixin.qq.com/s/PBqAm5bgLpyuG19rwwx7ng>