**一14一 一14一 一12一 一11一** 一10.5一 一10.5一 '一10一"

**目 录**

[1. 视频相关 2](#_Toc16735)

[1.1. 视频码率kbps 2](#_Toc26518)

[1.2. 帧数fps 2](#_Toc24873)

[1.2.1. 帧数 2](#_Toc10742)

[1.2.2. 电影24帧为什么不卡 2](#_Toc112)

[1.3. 显示器的刷新率 3](#_Toc13845)

[1.4. 视频的分辨率 3](#_Toc18142)

[1.5. 码率、网速、帧数、分辨率的关系 3](#_Toc10377)

[2. 图片相关 3](#_Toc3045)

[2.1. 图像基本数据结构 3](#_Toc15208)

[2.2. 图片格式的必要性 3](#_Toc2052)

[2.2.1. BMP格式 4](#_Toc6389)

[2.2.2. JPEG格式 4](#_Toc10568)

[2.2.3. PNG格式 5](#_Toc30860)

[2.2.4. GIF格式 5](#_Toc27899)

[2.3. 其他的资料 5](#_Toc42)

[3. MarkDown 6](#_Toc28569)

[3.1. 简介 6](#_Toc30472)

[3.2. 标题 6](#_Toc11641)

[3.3. 列表 6](#_Toc31256)

[3.4. 链接和图片 7](#_Toc474)

[3.5. 引用 7](#_Toc10600)

[3.6. 粗体、斜体和删除线 7](#_Toc7981)

[3.7. 代码引用 8](#_Toc15809)

[3.8. 表格 8](#_Toc21201)

[4. 区块链 8](#_Toc12269)

[4.1. 前世今生、 8](#_Toc14242)

[4.2. 应用场景 9](#_Toc10130)

[4.3. 比特币是什么 9](#_Toc22025)

[4.3.1. 比特币是数字货币 9](#_Toc21891)

[4.3.2. 比特币是去中心化记账系统 9](#_Toc25959)

[4.4. 比特币原理 9](#_Toc15859)

[4.4.1. 账本如何验证：Hash函数 9](#_Toc16922)

[4.5. 账户的所有权 10](#_Toc22771)

[4.6. 为什么记账 12](#_Toc1133)

[4.7. 共识机制 13](#_Toc25629)

[5. 区块链二 13](#_Toc18474)

[5.1. 特征 13](#_Toc22106)

[5.2. 分类 14](#_Toc29104)

[5.3. 网络架构 14](#_Toc12075)

[5.4. 区块链工作原理 15](#_Toc25780)

[5.5. 交易过程 16](#_Toc25818)

[6. Google搜索技巧 17](#_Toc537)

[6.1. 双引号 17](#_Toc4468)

[6.2. 减号/加号 17](#_Toc11656)

[6.3. \*号 17](#_Toc15503)

[6.4. 波浪号 17](#_Toc24739)

[6.5. filetype 18](#_Toc20505)

[6.6. inurl 18](#_Toc16659)

[6.7. intitle 18](#_Toc30917)

[6.8. inanchor 18](#_Toc6078)

[6.9. site 18](#_Toc30426)

[6.10. related 19](#_Toc15073)

[7. 各种简写的含义 19](#_Toc19048)

[8. OTG 22](#_Toc18929)

[9. 开源许可证的一些介绍 23](#_Toc15918)

[9.1. BSD 2-Clause license 23](#_Toc20832)

[9.2. BSD (3-Clause) License 23](#_Toc11963)

[9.3. Apache v2 License 23](#_Toc24002)

[9.4. MIT License 23](#_Toc6034)

[9.5. GPL v2 24](#_Toc231)

[9.6. GPL v3 24](#_Toc32333)

[9.7. Artistic License 2.0 24](#_Toc8410)

[9.8. Affero GPL 24](#_Toc21537)

[9.9. LGPL v2.1 24](#_Toc26614)

[9.10. Eclipse Public License v1.0 24](#_Toc26666)

[9.11. Mozilla Public License Version 2.0 25](#_Toc9829)

[9.12. Public Domain 25](#_Toc6501)

[10. Windows清理 25](#_Toc14528)

[10.1. C盘中有什么 25](#_Toc11598)

[10.2. Windows 文件夹清理 26](#_Toc25784)

[10.3. Users目录下的 AppData 文件夹清理 26](#_Toc6253)

[11. 第三方登录 26](#_Toc25574)

[11.1. 三方登录是什么 26](#_Toc12383)

[11.2. 理解几个概念 26](#_Toc22070)

[11.3. 为什么需要加入第三方登录 28](#_Toc15135)

[12. doc和docx的区别 29](#_Toc24710)

# 视频相关

## 视频码率kbps

视频码率就是数据传输时单位时间传送的数据位数，一般我们用的单位是kbps即千位每秒。通俗一点的理解就是取样率，单位时间内取样率越大，精度就越高，处理出来的文件就越接近原始文件。

但是文件体积与取样率是成正比的，所以几乎所有的编码格式重视的都是如何用最低的码率达到最少的失真，围绕这个核心衍生出来的cbr（固定码率）与vbr（可变码率），都是在这方面做的文章

基本的算法是：【码率】(kbps)=【文件大小】(KB) \* 8 / 【时间】(秒)，举例，D5的碟，容量4.3G，其中考虑到音频的不同格式，姑且算为600M，（故剩余容量为4.3\*1024-600=3803.2M)，所以视频文件应不大于3.7G，本例中取视频文件的容量为3.446G，视频长度100分钟（6000秒），计算结果：码率约等于4818kbps(3.446 \* 1024 \* 1024 \* 8 / 6000 = 4817.857)。

码率几点原则：

1、码率和质量成正比，但是文件体积也和码率成正比

2、码率超过一定数值，对图像的质量没有多大影响。

3、DVD的容量有限，无论是标准的4.3G，还是超刻，或是D9，都有极限。视频码率　计算机中的信息都是二进制的0和1来表示，其中每一个0或1被称作一个位，用小写b表示，即bit（位）；大写B表示byte,即字节，一个字节=八个位，即1B=8b；前面的大写K表示1024的意思，即1024个位（Kb)或1024个字节(KB)。表示文件的大小单位，一般都使用字节（KB）来表示文件的大小。kbps：首先要了解的是，ps指的是/s，即每秒。Kbps指的是网络速度，也就是每秒钟传送多少个千位的信息（K表示千位，Kb表示的是多少千个位），为了在直观上显得网络的传输速度较快，一般公司都使用kb（千位）来表示。1KB/S=8Kbps。ADSL上网时的网速是512Kbps,如果转换成字节，就是512/8=64KB/S(即64千字节每秒）。

4、一般来说，如果是1M的宽带，在网上只能看不超过1024kbps的视频，超过1024kbps的视频只能等视频缓冲才能流畅的观看。

## 帧数fps

### 帧数

FPS每秒传输帧数(Frames Per Second)是图像领域中的定义，是指画面每秒传输帧数，通俗来讲就是指动画或视频的画面数。FPS是测量用于保存、显示动态视频的信息数量。每秒钟帧数愈多，所显示的动作就会愈流畅。通常，要避免动作不流畅的最低是30。某些计算机视频格式，每秒只能提供15帧。

FPS也可以理解为我们常说的“刷新率（单位为Hz）”，例如我们常在CS游戏里说的“FPS值”。我们在装机选购显卡和显示器的时候，都会注意到“刷新率”。一般我们设置缺省刷新率都在75Hz（即75帧/秒）以上。例如：75Hz的刷新率刷也就是指屏幕一秒内只扫描75次，即75帧/秒。而当刷新率太低时我们肉眼都能感觉到屏幕的闪烁，不连贯，对图像显示效果和视觉感观产生不好的影响。

电影以每秒24张画面的速度播放，也就是一秒钟内在屏幕上连续投射出24张静止画面。有关动画播放速度的单位是fps，其中的f就是英文单词Frame（画面、帧），p就是Per（每），s就是Second（秒）。用中文表达就是多少帧每秒，或每秒多少帧。电影是24fps，通常简称为24帧。

### 电影24帧为什么不卡

电影的24帧里，每一帧包含的是1/24秒的画面信息。举例来说吧，比如电影中的一把高速挥舞的刀，你在电影里看到的静止截图就是一个刀的运动残像的影子（就和相机没开运动模式拍摄高速物体一样，曝光之间的物体运动过程都呈现给你了），其实这一帧就已经把刀在1/24秒内的运动轨迹画出来了，电影在高速运动状态下的截图很少有与静止状态下一样十分清晰的，原因就在此，而这样连续运动轨迹的24帧构成了电影的画面，所以虽然电影的画面只有24帧，但却能反应出物体连续运动的状态。

游戏的30帧60帧是30幅60幅的逐渐变化的图，每张图都是一个静态的图，单张图无法反应出物体的运动轨迹。虽然现在也有了动态模糊这种特效，但毕竟那是后期加上去的特效，与电影的24帧的那种运动模糊还是有差别的。所以游戏要想连贯起来，必须要靠更多的帧数来堆起来，人才能觉得足够连贯。

所以电影的24帧看起来要比游戏的24帧甚至更高一些的帧数要连贯，起码不会感到明显的不适。

## 显示器的刷新率

刷新频率：即屏幕刷新的速度。刷新频率越低，图像闪烁和抖动的就越厉害，眼睛疲劳得就越快。采用70Hz以上的刷新频率时才能基本消除闪烁，显示器最好稳定工作在允许的最高频率下，一般是85Hz。

在显示器内部，有一些振荡电路。

人们通常所说的刷新频率，指的就是振荡电路的频率。刷新频率的计算公式是：水平同步扫描线X帧频=刷新频率。普通显示器的刷新频率在15.75kHz－95kHz间。15.75kHz是人体对显示器最低要求的刷新频率，是由525（线）X30（fps）=15.75kHz计算所得。

由此，我们可以逆推出显示器扫描一条水平线的周期：众所周知，周期和频率是倒数关系，即1/频率=周期。在这里，1/15.75kHz= 63.5us（微秒），也就是说在每帧525线、每秒30帧的模式下，显示器扫描一条水平线的周期是63.5微秒。

## 视频的分辨率

可以简单的理解为：

宽p 长 x 宽

480p 720 x 480

720p 1280 x 720

1080p 1920 x 10800

## 码率、网速、帧数、分辨率的关系

码率 = 分辨率 \* 品质 \* 帧数 / 1024

码率 = 网速 \* 8

kbps = Mbps \* 1024

以直播举例：直播要上传的是视频+音频，所以上传码率是视频码率 + 音频码率 = 80%左右的上传网络带宽（留给网络游戏和其它的带宽需求）

例如：720P的1280 \* 720 \* 0.08 \* 24 / 1024 = 1728kbps（纯视频部分）。如果加上音频128kbps的话就是1728 + 128 = 1856kbps = 232KB/S = 1.81Mbps

# 图片相关

## 图像基本数据结构

在计算机中, 图像是由一个个像素点组成，像素点就是颜色点，而颜色最简单的方式就是用RGB或RGBA表示, 如图：

如果有A通道就表明这个图像可以有透明效果。R,G,B每个分量一般是用一个字节(8位)来表示，所以图(1)中每个像素大小就是3\*8=24位图, 而图(2)中每个像素大小是4\*8=32位。

## 图片格式的必要性

如果将图像原始格式直接存储到文件中将会非常大，比如一个5000\*5000 24位图，所占文件大小为5000\*5000\*24/8字节 = 71.5MB, 其大小非常可观。

如果用zip或rar之类的通用算法来压缩像素数据，得到的压缩比例通常不会太高，因为这些压缩算法没有针对图像数据结构进行特殊处理。

### BMP格式

bmp格式没有压缩像素格式，存储在文件中时先有文件头、再图像头、后面就都是像素数据了，上下颠倒存储。用windows自带的mspaint工具保存bmp格式时，可以发现有四种bmp可供选择：

* 单色: 一个像素只占一位，要么是0，要么是1，所以只能存储黑白信息
* 16色位图: 一个像素4位，有16种颜色可选
* 256色位图: 一个像素8位，有256种颜色可选
* 24位位图: 颜色可有2^24种可选，对于人眼来说完全足够了。

优点：BMP 支持 1 位到 24 位颜色深度。BMP 格式与现有 Windows 程序（尤其是较旧的程序）广泛兼容。

缺点： BMP 不支持压缩，这会造成文件非常大。

### JPEG格式

* jpeg是有损压缩格式, 将像素信息用jpeg保存成文件再读取出来，其中某些像素值会有少许变化。在保存时有个质量参数可在[0,100]之间选择，参数越大图片就越保真，但图片的体积也就越大。一般情况下选择70或80就足够了。
* jpeg没有透明信息。
* jpeg比较适合用来存储相机拍出来的照片，这类图像用jpeg压缩后的体积比较小。其使用的具体算法核心是离散余弦变换、Huffman编码、算术编码等技术，有兴趣的同学可以在网上找一大堆资料，本文就不详细介绍了。

接下来要介绍一个有关jpeg非常实用的技术——jpeg格式支持不完全读取整张图片，即可以选择读取原图、1/2、1/4、1/8大小的图片。比如5000\*5000的一张大图，可以只读取将其缩小成1/8后即625\*625大小的图片。 这样比先完全读取5000\*5000的图像，再用算法缩小成625\*625大小不知快多少倍。如果应用需求只需要一张小图时，这种读取方式就可以大显身手了。android平台下实现jpeg预读：

**private** **int** **calculateInSampleSize**(**BitmapFactory**.**Options** options, **int** reqWidth, **int** reqHeight) {

**int** width = options.outWidth;**// 源图片的宽度**

**int** height = options.outHeight;

**if** (width <= 0 || height <= 0) {**//有时候会出现-1,这时候就要用ExifInterface来获取**

**try** {

**ExifInterface** exifInterface = **new** ExifInterface(mFilePath);

height = exifInterface.getAttributeInt(**ExifInterface**.**TAG\_IMAGE\_LENGTH**, **ExifInterface**.**ORIENTATION\_NORMAL**);

width = exifInterface.getAttributeInt(**ExifInterface**.**TAG\_IMAGE\_WIDTH**, **ExifInterface**.**ORIENTATION\_NORMAL**);

} **catch** (**IOException** e) {

e.printStackTrace();

}

}

**int** inSampleSize = 1;

**while** (width / inSampleSize > reqWidth || height / inSampleSize > reqHeight) {

inSampleSize \*= 2;**//必须是2的幂数**

}

**return** inSampleSize;

}

**private** **Bitmap** **compressBitmap**(**String** pathName, **int** reqWidth, **int** reqHeight) {

**// 第一次解析将inJustDecodeBounds设置为true，来获取图片大小**

**final** **BitmapFactory**.**Options** options = **new** **BitmapFactory**.Options();

options.inJustDecodeBounds = **true**;**//仅仅只解析大小**

**BitmapFactory**.*decodeFile*(pathName, options);

**// 调用上面定义的方法计算inSampleSize值（比例）**

options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);

**// 使用获取到的inSampleSize值再次解析图片**

options.inJustDecodeBounds = **false**;

**Bitmap** bitmap = **BitmapFactory**.*decodeFile*(pathName, options);

**return** bitmap;

}

### PNG格式

* png是一种无损压缩格式， 压缩大概是用行程编码算法。
* png可以有透明效果。
* png比较适合适量图,几何图。 比如本文中出现的这些图都是用png保存，比用joeg保存体积要小。
* png可能有24位图和32位图之分。32位图就是带有alpha通道的图片。

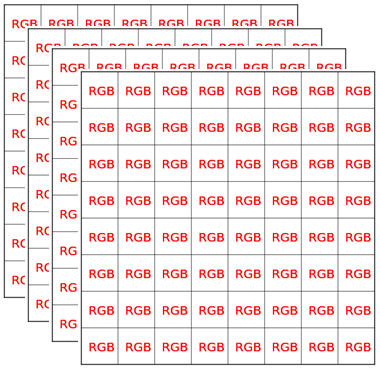
再强调一下: jpeg比较适合存储色彩“杂乱”的拍摄图片，png比较适合存储几何特征强的图形类图片。

### GIF格式

图形交换格式GIF（外语全称：Graphics Interchange Format），是一种基于LZW算法的连续色调的无损压缩格式。其压缩率一般在50%左右。

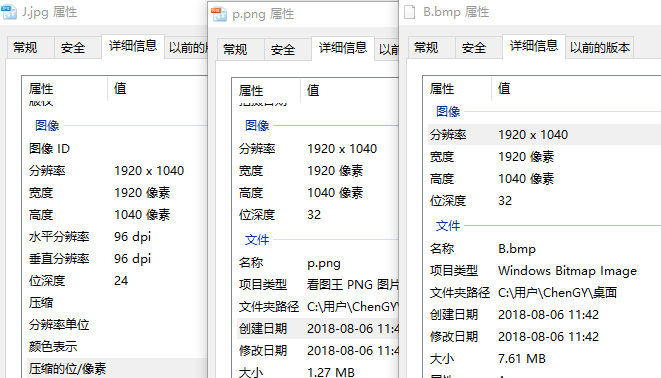
GIF图像文件的数据是经过压缩的，而且是采用了可变长度等压缩算法。所以GIF的图像深度从lbit到8bit，也即GIF最多支持256种色彩的图像。另外，GIF不支持Alpha透明通道。

上面提到的bmp,jpeg,png图片都只有一帧，而gif可以保存多帧图像，如图所示



## 其他的资料

用qq截同一个图，保存为三种格式的属性



位深度与色深

* 位深度、图像深度指的是存储每个像素所用的位数，主要用于存储
* 色深指的是每一个像素点用多少bit存储颜色，属于图片自身的一种属性

Android Bitmap中的Config参数其实指的就是色深

* Bitmap.Config ARGB\_4444：每个像素占四位，即A=4，R=4，G=4，B=4，那么一个像素点占4+4+4+4=16位
* Bitmap.Config ARGB\_8888：每个像素占四位，即A=8，R=8，G=8，B=8，那么一个像素点占8+8+8+8=32位
* Bitmap.Config RGB\_565：每个像素占四位，即R=5，G=6，B=5，没有透明度，那么一个像素点占5+6+5=16位
* Bitmap.Config ALPHA\_8：每个像素占四位，只有透明度，没有颜色

举个例子：

* 100像素\*100像素 色深32位(ARGB\_8888) 保存时位深度为24位 的图片
* 在内存中所占大小为：100 \* 100 \* (32 / 8)Byte
* 在文件中所占大小为 100 \* 100 \* ( 24/ 8 ) \* 压缩效率 Byte

# MarkDown

## 简介

Markdown胜出Html的地方就是其可读性。虽然都是标记语言，html不用浏览器解析， 是很难读的。而 Markdown即使不经过解析，也很好读。

以前的文章格式，并非生来如此，而是纸质印刷的需要。比如：段首要空两个字，段与段之间不留空行，直接引语都混在一段之中。这些并不是为了便于阅读，而是为了节约纸张。

而电子阅读时代，屏幕的空间滚动起来，可以无限大，但是跟纸张比起来，屏幕伤眼睛，容易使我们疲劳，因此人们需要更为清晰的排版，这就是Markdown存在的意义。

Markdown 的优点如下：

* 纯文本，所以兼容性极强，可以用所有文本编辑器打开。
* 让你专注于文字而不是排版。
* 格式转换方便，Markdown 的文本你可以轻松转换为 html、电子书等。
* Markdown 的标记语法有极好的可读性

## 标题

在文本前面加上 # 即可，同理、你还可以增加二级标题、三级标题、四级标题、五级标题和六级标题，总共六级，只需要增加 # 即可，标题字号相应降低。注：# 和「一级标题」之间建议保留一个字符的空格，这是最标准的 Markdown 写法。

# 一级标题

## 二级标题

...

###### 六级标题

## 列表

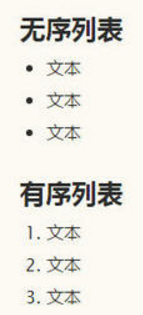
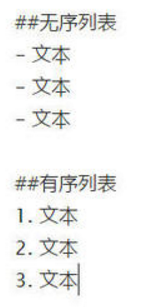
在文字前面加上 - ，也可以在文字前面加上 1. 2. 3. ，注：-、1.和文本之间要保留一个字符的空格。

- 文本1

- 文本2

1. 文本1

2. 文本2



## 链接和图片

在 Markdown 中，插入链接不需要其他按钮，你只需要使用 [显示文本](链接地址) 这样的语法即可：

[简书](http://www.jianshu.com)

在 Markdown 中，插入图片不需要其他按钮，你只需要使用 ![错误图片占位文本](图片链接地址) 这样的语法即可：

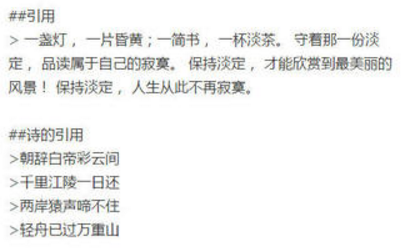
![](http://upload-images.jianshu.io/upload\_images/259-0ogr2/autent/strageView2/2/w/1240)

## 引用

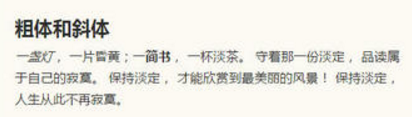
在希望引用的文字前面加上 > 就好了，注：> 和文本之间要保留一个字符的空格例如：

> 一盏灯， 一片昏黄

## 粗体、斜体和删除线

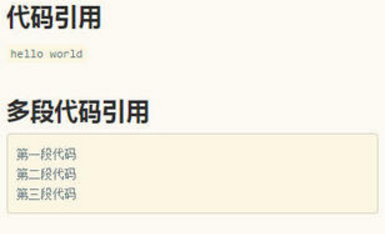
用两个 \* 包含一段文本是粗体的语法，用一个 \* 包含一段文本是斜体的语法，用两个 ~ 包含一段文本是粗体的语法：

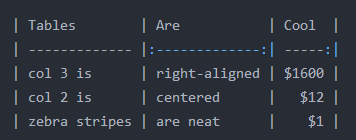
## 代码引用

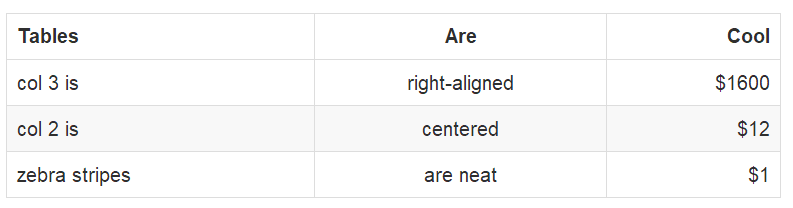
需要引用代码时，如果引用的语句只有一段，不分行，可以用 ` 将语句包起来。如果引用的语句为多行，可以将```置于这段代码的首行和末行。

## 表格

默认居左，可设置居中和居右



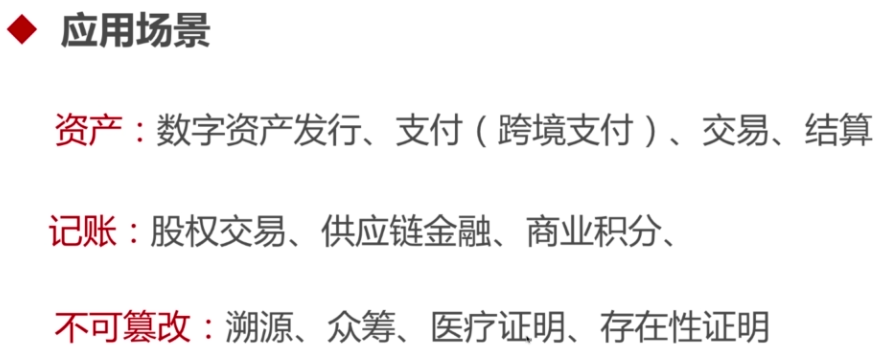


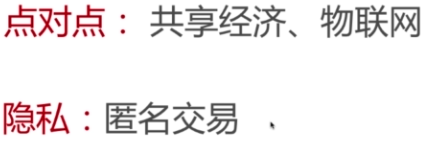
# 区块链

## 前世今生、

密码朋克、中本聪 2018年

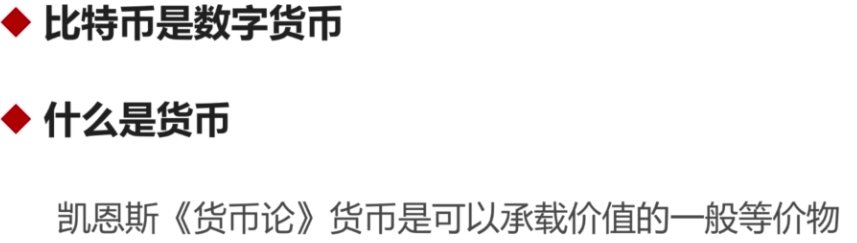
## 应用场景



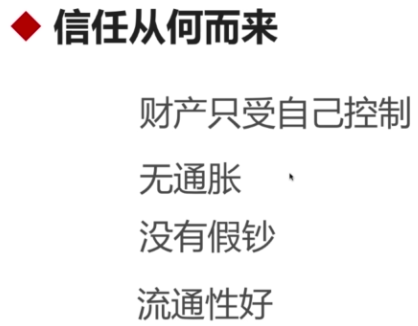


## 比特币是什么

### 比特币是数字货币



数字货币：一串数字，价值来源于使用它的人对它价值的认可

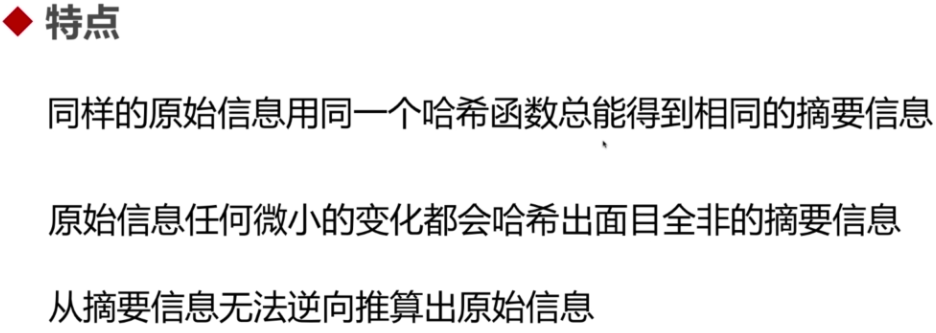


### 比特币是去中心化记账系统

## 比特币原理

### 账本如何验证：Hash函数



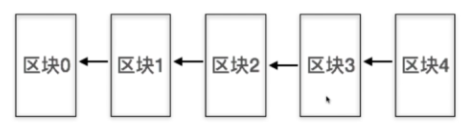


区块：





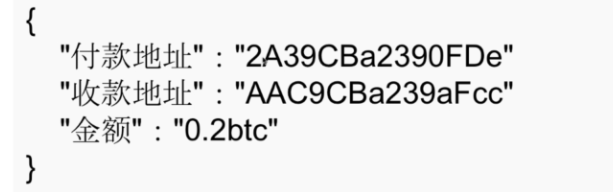




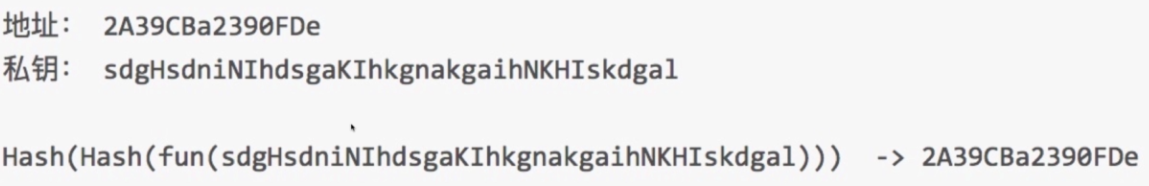
Hash函数会把上一个节点的摘要值一起算，所以对比最后一次的摘要值即可，最后一次的值正确，之前的肯定也正确

## 账户的所有权

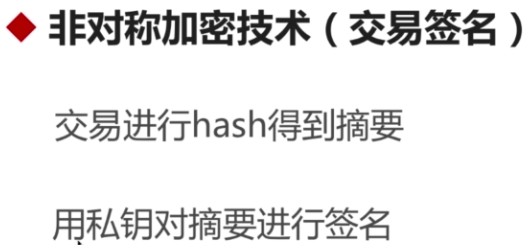
账号就是地址，谁能使用这个地址进行支付，谁就拥有这个地址的账户的所有权



私钥 → 地址

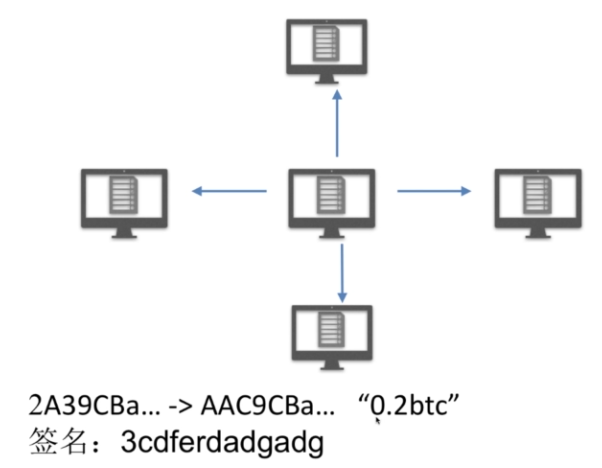


如何在不泄露私钥的情况下证明我们拥有某个地址的私钥





付款的节点在整个网络里进行广播，通知周围的节点，内容包含原始信息和签名信息，周围的节点进行验证，如果通过，周围节点会继续往自己的周围节点通知

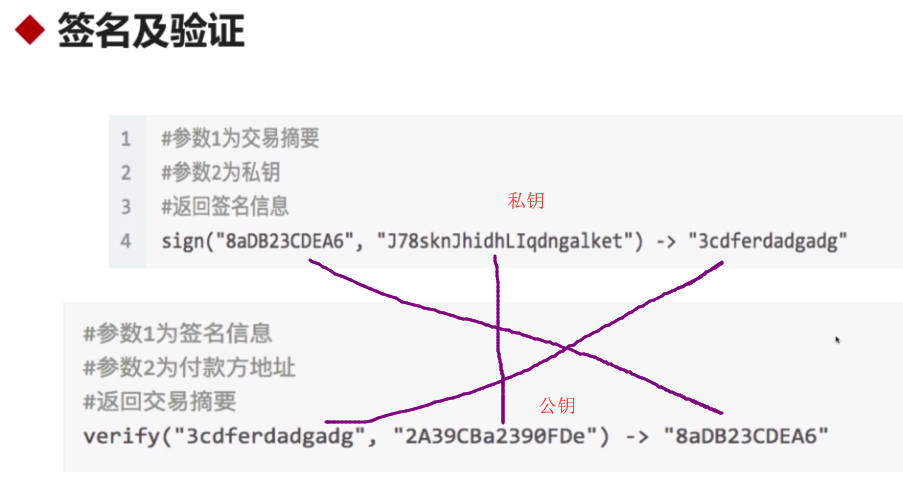




付款的地址是公钥

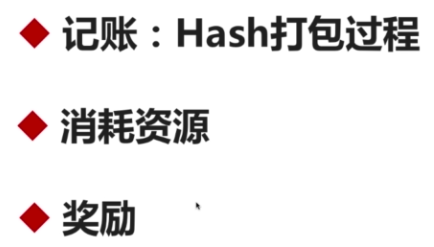
签名的过程是用私钥，对摘要加密，生成签名

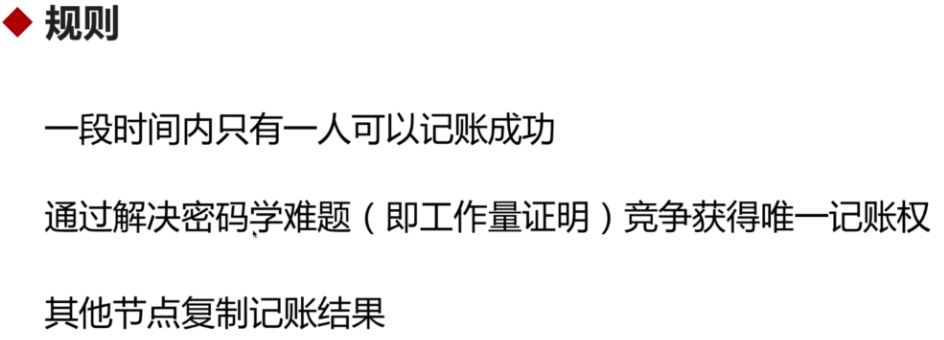
验证的过程是用公钥，对签名解密，生成摘要

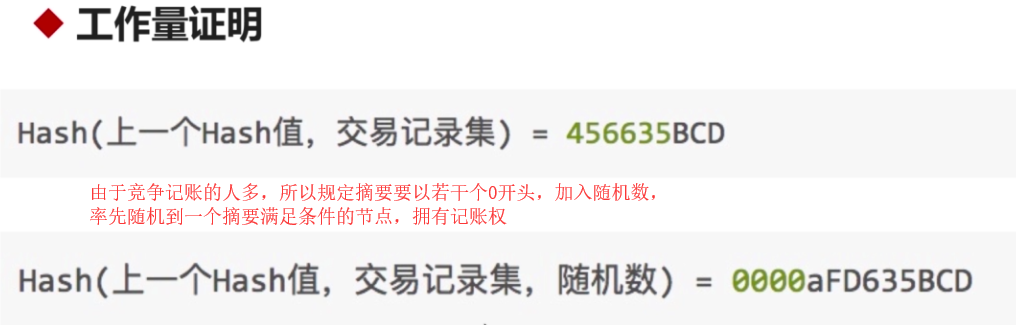


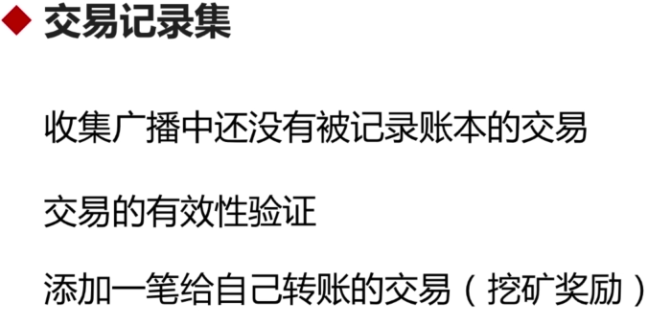
验证的时候是没有私钥的信息的，所以隐私、安全

## 为什么记账





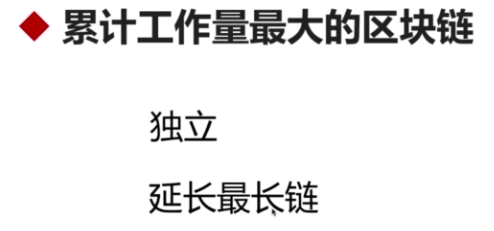


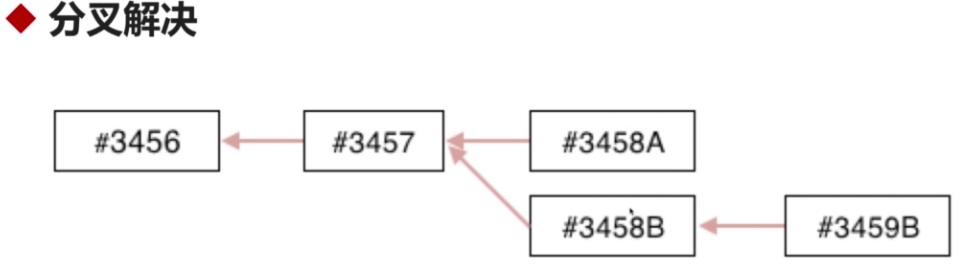




## 共识机制







# 区块链二

## 特征

开放,共识

任何人都可以参与到区块链网络,每一台设备都能作为个节点,每个节点都允许获得份完整的数据库拷贝。节点间基于套共识机制,通过竟争计算共同维护整个区块链。任一节点失效,其余节点仍能正常工作

去中心,去信任

区块链由众多节点共同组成一个端到端的网络,不存在中心化的设备和管理机构。节点之间数据交换通过数字签名技术进行验证,无需互相信任,只要按照系统既定的规则进行,节点之间不能也无法欺骗其它节点。

交易透明,双方匿名

区块链的运行规则是公开透明的,所有的数据信息也是公开的,因此每笔交易都对所有节点可见。由于节点与节点之间是去信任的,因此节点之间无需公开身份,每个参与的节点都是匿名的

可篡改,可追溯

单个甚至多个节点对数据库的修改无法影响其他节点的数据库,除非能控制整个网络中超过51%的节点同时修改,这几乎不可能发生。区块链中的每一笔交易都通过密码学方法与相邻两个区块串联,因此可以追溯到任何一笔交易的前世今生。

## 分类

公有链

无官方组织及管理机构,无中心服务器,参与的节点按照系统规则自由接入网络、不受控制,节点间基于共识机制开展工作。

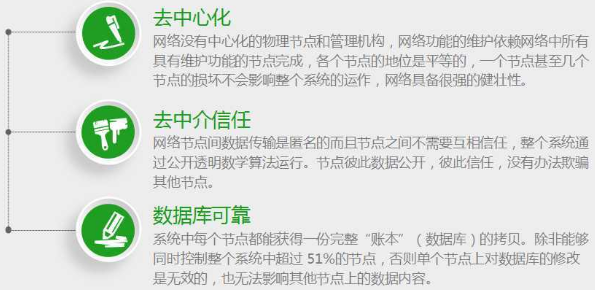
私有链

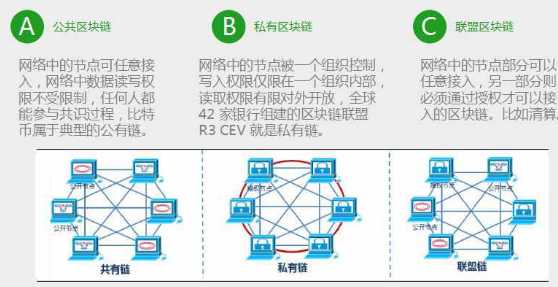
建立在某个企业内部,系统的运作规则根据企业要求进行设定,修改甚至 是读取权限仅限于少数节点,同时仍保留着区块链的真实性和部分去中心化的特性

联盟链

由若干机构联合发起,介于公有链和私有链之间,兼具部分去中心化的特性。

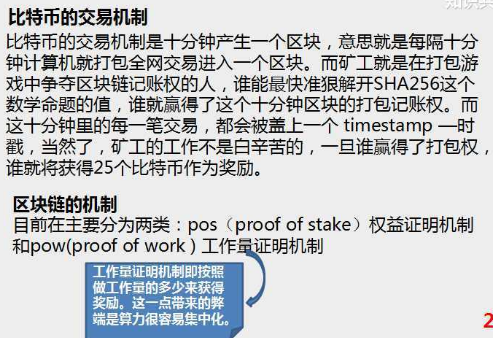
## 网络架构

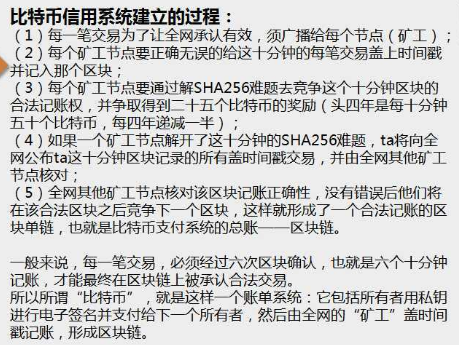






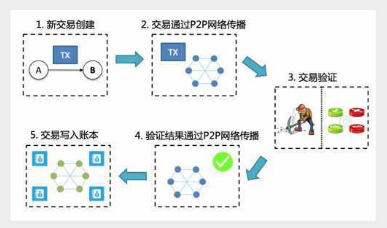
## 区块链工作原理

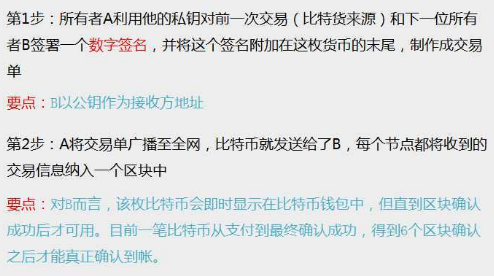


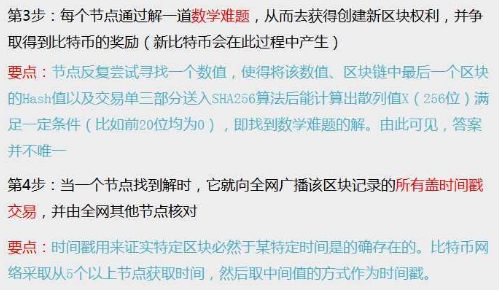


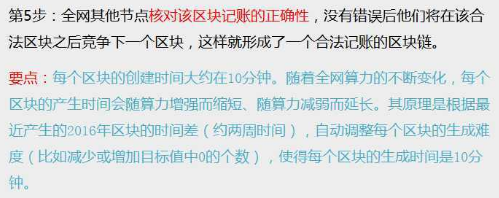


## 交易过程









# Google搜索技巧

## 双引号

* 格式："xxx"，xxx表示关键词。
* 功能：搜索包含完整xxx关键词的信息。通常如果你直接搜索xxx关键词，出来的结果会包含很多无用页面，它们仅仅包含了xxx中的某一个字。在大量的信息中，这对我们是种迷惑。
* 例子：CoorChice帅 pk "CoorChice帅"

## 减号/加号

* 格式：xxx -yyy或xxx +zzz +ccc，减号/加号前需要一个空格，减号/加号后不要空格，多个词之间需要空格。
* 功能：搜索关于xxx，但是排出包含yyy的信息，或者同时包含zzz和ccc的信息。这个技巧可以帮助我们过滤大量无用信息，或者使搜索结果更加精确。
* 例子：CoorChice pk CoorChice -简书

## \*号

* 格式："x\*x"，配合双引号使用。
* 功能：搜索包含xx结构的信息。号表示任意字符。这个技巧可以帮助你搜索到结构类似的信息。
* 例子："程序员" pk "程\*员"

## 波浪号

* 格式："xxx"～yyy，不要空格。
* 功能：搜索包含xxx，并且与yyy有关的信息。yyy是用来补充修饰xxx的，搜索结果中可能包含yyy或其近义词。这个技巧可以让你更加精确的获得搜索结果。
* 例子："好书" pk "好书"～技术

## filetype

* 格式："xxx" filetype:yyy，filetype前需要一个空格。
* 功能：搜索包含xxx，并且是yyy格式的文件。用这个技巧去搜索文献、文件，简直不要太爽！
* 例子："MachineLearning" pk "MachineLearning" filetype:xls

## inurl

* 格式：inurl:xxx，不要有空格！
* 功能：搜索url（姑且称为网址）中包含xxx的网页。通常url中包含关键字的结果与你想找的内容相关性更大。
* 例子：inurl:程序员



## intitle

* 格式：intitle:xxx，不要有空格！
* 功能：搜索到的网页Title中包含xxx。通常网页的Title标题中如果包含了关键字，那就更可能是你要找的信息。
* 例子：intitle:CoorChice



## inanchor

* 格式：inanchor:xxx，不要有空格！
* 功能：如果页面A中存在一个“超链接”链接到页面B，并且这个“超链接”的文本内容中包括了xxx，那么页面B将作为搜索结果出现（搜索到的结果页面是B哦！）。实际上，这个“超链接”的文本被称作“链接锚文本”。这个技巧能够让你看到竞争对手是否做过推广（即在其它网站投放了推广链接）。当然它的运用十分广泛。
* 例子：inanchor:coorchice

## site

* 格式：site:url，不要有空格！url就是网址喽。
* 功能：在指定的网站内搜索。有时候，你可能只想搜索某个网站内的信息，就可以使用这个技巧。
* 例子：site:https://chenbingx.github.io/



## related

* 格式：related:url，不要有空格！url就是网址喽。
* 功能：搜索和指定网站相关或类似的网站。这个方法通常比你直接搜“和xxx类似的网站”靠谱太多了。
* 例子：related:http://www.jianshu.com/



# 各种简写的含义

**ADT**

ADT（Android Development Tools，安卓开发工具）在Eclipse编译IDE环境中，需安装ADT(Android Developer Tools)Plug-in,这是Android在Eclipse上的开发工具

**AIDL**

AIDL（Android Interface Definition Language，Android接口定义语言）Android系统中的进程之间不能共享内存，因此，需要提供一些机制在不同进程之间进行数据通信。为了使其他的应用程序也可以访问本应用程序提供的服务，Android系统采用了远程过程调用（Remote Procedure Call，RPC）方式来实现。与很多其他的基于RPC的解决方案一样，Android使用一种接口定义语言（Interface Definition Language，IDL）来公开服务的接口。我们知道4个Android应用程序组件中的3个（Activity、BroadcastReceiver和ContentProvider）都可以进行跨进程访问，另外一个Android应用程序组件Service同样可以。因此，可以将这种可以跨进程访问的服务称为AIDL服务。

**ANR**

ANR（Application Not Responding，应用程序无响应）在Android上，如果你的应用程序有一段时间响应不够灵敏，系统会向用户显示一个应用程序无响应（对话框。用户可以选择“等待”而让程序继续运行，也可以选择“强制关闭”。所以一个流畅的合理的应用程序中不能出现anr，而让用户每次都要处理这个对话框。因此，在程序里对响应性能的设计很重要，这样系统不会显示ANR给用户。默认情况下，在android中Activity的最长执行时间是5秒，BroadcastReceiver的最长执行时间则是10秒。

**API**

API（Application Programming Interface，应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节

**DNS**

DNS（Domain Name System，域名系统），因特网上作为域名和[IP地址](http://baike.baidu.com/view/3930.htm" \t "http://baike.baidu.com/_blank)相互映射的一个[分布式数据库](http://baike.baidu.com/view/68389.htm" \t "http://baike.baidu.com/_blank)，能够使用户更方便的访问[互联网](http://baike.baidu.com/view/6825.htm" \t "http://baike.baidu.com/_blank)，而不用去记住能够被机器直接读取的IP数串。通过[主机](http://baike.baidu.com/view/23880.htm" \t "http://baike.baidu.com/_blank)名，最终得到该主机名对应的IP地址的过程叫做域名解析（或主机名解析）。DNS协议运行在[UDP](http://baike.baidu.com/view/30509.htm" \t "http://baike.baidu.com/_blank)协议之上，使用端口号53。每个[IP地址](http://baike.baidu.com/view/3930.htm" \t "http://baike.baidu.com/_blank)都可以有一个主机名，主机名由一个或多个字符串组成，字符串之间用小数点隔开。有了主机名，就不要死记硬背每台IP设备的IP地址，只要记住相对直观有意义的主机名就行了。这就是DNS协议所要完成的功能。

**GUI**

GUI（Graphical User Interface，图形用户[界面](http://baike.baidu.com/view/240704.htm" \t "http://baike.baidu.com/_blank)，又称图形[用户接口](http://baike.baidu.com/view/1309799.htm" \t "http://baike.baidu.com/_blank)）是指采用图形方式显示的计算机操作用户[界面](http://baike.baidu.com/view/240704.htm" \t "http://baike.baidu.com/_blank)

**IDE**

IDE（Integrated Development Environment，完全开发环境），用于程序开发的软件（比如eclipse），一般包括代码编辑器、编译器、调试器和图形用户界面工具。可以辅助开发程序的应用软件

**JDK**

JDK（Java Development Kit，Java 语言的软件开发工具包），主要用于移动设备、嵌入式设备上的java应用程序。JDK是整个java开发的核心，它包含了JRE，JAVA开发工具（编译工具--javac.exe，打包工具--jar.exe）和JAVA基础的类库。所以安装了JDK就不用单独安装JRE了。简单而言，使用JDK开发程序，交个JRE去运行

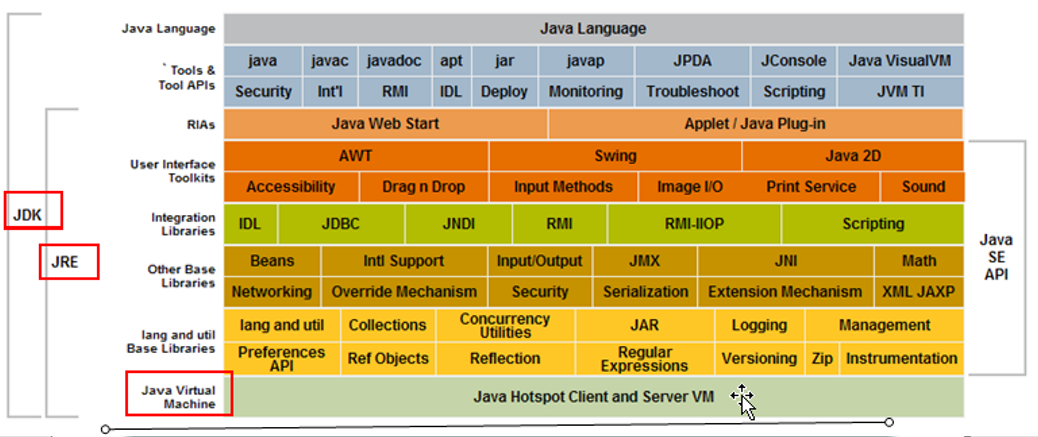
JDK(Java Development Kit) 是 Java 语言的软件开发工具包(SDK)。

SE(J2SE)，standard edition，标准版，是我们通常用的一个版本，从JDK 5.0开始，改名为Java SE。

EE(J2EE)，enterprise edition，企业版，使用这种JDK开发J2EE应用程序，从JDK 5.0开始，改名为Java EE。

ME(J2ME)，micro edition，主要用于移动设备、嵌入式设备上的java应用程序，从JDK 5.0开始，改名为Java ME。

没有JDK的话，无法编译Java程序，如果想只运行Java程序，要确保已安装相应的JRE。



**JNI**

JNI（Java Native Interface，Java本地接口），它提供了若干的[API](http://baike.baidu.com/subview/16068/5889234.htm" \t "http://baike.baidu.com/_blank)实现了Java和其他语言的通信（主要是[C](http://baike.baidu.com/subview/10075/6770152.htm" \t "http://baike.baidu.com/_blank)&[C++](http://baike.baidu.com/view/824.htm" \t "http://baike.baidu.com/_blank)），它允许Java代码和其他语言写的代码进行交互

**JRE**

JRE（Java Runtime Environment，Java运行环境），运行JAVA程序所必须的环境的集合，包含JVM标准实现及Java核心类库。

**JVM**

JVM（Java Virtual Machine，Java虚拟机），JVM是一种用于计算设备的规范，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。（JVM依赖计算机，不同系统的JVM不同的）

Java语言的一个非常重要的特点就是与平台的无关性。而使用Java虚拟机是实现这一特点的关键。一般的高级语言如果要在不同的平台上运行，至少需要编译成不同的目标代码。而引入Java语言虚拟机后，Java语言在不同平台上运行时不需要重新编译。Java语言使用Java虚拟机屏蔽了与具体平台相关的信息，使得Java语言编译程序只需生成在Java虚拟机上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。Java虚拟机在执行字节码时，把字节码解释成具体平台上的机器指令执行。这就是Java的能够“一次编译，到处运行”的原因

**LBS**

LBS（Location Based Service，基于位置的服务）它是通过电信移动运营商的无线电通讯网络（如GSM网、CDMA网）或外部定位方式(如GPS)获取移动终端用户的位置信息。

**NDK**

NDK（Native Development Kit，原生开发包）跟sdk差不多的是他也是一个开发工具包。用他开发[c/c++](https://www.baidu.com/s?wd=c/c++&rsv_qp=87b99ui4339p&rsv_xpth=http://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&tn=&wd=jquery get&rsv_qp=85cb9ab80001cc79&rsv_t=5e5dhOzLfuv4s&rsv_enter=1&rsv_sug3=11&rsv_sug1=7&rsv_sug2=0&inputT=2736&rsv_sug4=4394&tn=90011882_hao_pg" \t "http://zhidao.baidu.com/_blank)是很方便的。他有一个强大的编译集合。严格说AndroidNDK只是AndroidSDK的一个组件，通过NDK，你就可以用c或c＋＋来[开发app](https://www.baidu.com/s?wd=%E5%BC%80%E5%8F%91app&rsv_qp=87b99ui4339p&rsv_xpth=http://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&tn=&wd=jquery get&rsv_qp=85cb9ab80001cc79&rsv_t=5e5dhOzLfuv4s&rsv_enter=1&rsv_sug3=11&rsv_sug1=7&rsv_sug2=0&inputT=2736&rsv_sug4=4394&tn=90011882_hao_pg" \t "http://zhidao.baidu.com/_blank)了。很早以前android是只有sdk的。并没有ndk。这就意味着一旦android的开发者要使用[c/c++](https://www.baidu.com/s?wd=c/c++&rsv_qp=87b99ui4339p&rsv_xpth=http://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&tn=&wd=jquery get&rsv_qp=85cb9ab80001cc79&rsv_t=5e5dhOzLfuv4s&rsv_enter=1&rsv_sug3=11&rsv_sug1=7&rsv_sug2=0&inputT=2736&rsv_sug4=4394&tn=90011882_hao_pg" \t "http://zhidao.baidu.com/_blank)的三方库或者需要用到[c/c++](https://www.baidu.com/s?wd=c/c++&rsv_qp=87b99ui4339p&rsv_xpth=http://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&tn=&wd=jquery get&rsv_qp=85cb9ab80001cc79&rsv_t=5e5dhOzLfuv4s&rsv_enter=1&rsv_sug3=11&rsv_sug1=7&rsv_sug2=0&inputT=2736&rsv_sug4=4394&tn=90011882_hao_pg" \t "http://zhidao.baidu.com/_blank)就必须使用非官方的法子。用java的jni去调用c/c++。耍小聪明走后门一样。而ndk的出现就意味着jni调用的这种方法转正了变成官方了

**OOA**

OOA（Object Oriented Analysis，[面向对象分](http://baike.baidu.com/view/249254.htm" \t "http://baike.baidu.com/_blank)析方法）是确定需求或者业务的角度，按照面向对象的思想来分析业务。

**OOD**

OOD（Object Oriented Design，[面向对象设计](http://baike.baidu.com/view/249254.htm" \t "http://baike.baidu.com/_blank)）是OO方法中一个中间过渡环节。其主要作用是对OOA分析的结果作进一步的[规范](http://baike.baidu.com/view/113045.htm" \t "http://baike.baidu.com/view/_blank)化整理，以便能够被OOP直接[接受](http://baike.baidu.com/subview/64274/7597876.htm" \t "http://baike.baidu.com/view/_blank)。

**OOM**

OOD（Out Of Memory，内存溢出），一般是由于程序编写者对内存使用不当，如对该释放的内存资源没有释放，导致其一直不能被再次使用而使计算机内存被耗尽的现象。重启计算机即可，但根本解决办法还是对代码进行优化。

**OOP**

OOP（Object Oriented Programming，[面向对象](http://baike.baidu.com/view/249254.htm" \t "http://baike.baidu.com/_blank)编程）是一种计算机编程架构。OOP 的一条基本原则是计算机程序是由单个能够起到[子程序](http://baike.baidu.com/view/933538.htm" \t "http://baike.baidu.com/_blank)作用的单元或对象组合而成。

**SDK**

SDK（Software Development Kit，软件开发包）一般是一些被软件工程师用于为特定的软件包、软件框架、硬件平台、操作系统等建立应用软件的开发工具的集合。在Android中，它为开发者提供了库文件以及其它开发所用到的工具，简单理解为开发工具包集合。SDK 还经常包括示例代码、支持性的技术注解或者其他的为基本参考资料澄清疑点的支持文档。可以说只要你使用java去开发Android这个东西就必须用到。他包含了SDK Manager 和 AVD Manage，对于android系统的一些开发版本的管理以及模拟器管理。

**UML**

UML(Unified Modeling Language,统一建模语言或[标准建模语言](http://baike.baidu.com/view/3458435.htm" \t "http://baike.baidu.com/_blank))，是始于1997年一个[OMG](http://baike.baidu.com/subview/138917/5088704.htm" \t "http://baike.baidu.com/_blank)标准，它是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持，包括由需求分析到规格，到构造和配置。 面向对象的分析与设计(OOA&D，OOAD)方法的发展在80年代末至90年代中出现了一个高潮，[UML](http://baike.baidu.com/view/23396.htm" \t "http://baike.baidu.com/_blank)是这个高潮的产物。它不仅统一了Booch、Rumbaugh和Jacobson的表示方法，而且对其作了进一步的发展，并最终统一为大众所接受的[标准建模语言](http://baike.baidu.com/view/3458435.htm" \t "http://baike.baidu.com/_blank)。

**URI**

URI（Uniform Resource Identifier，统一资源标识符)是一个用于标识某一互联网资源名称的字符串。该种标识允许用户对任何（包括本地和互联网）的资源通过特定的协议进行交互操作。URI由包括确定语法和相关协议的方案所定义。URI实例可以代表绝对的，也可以是相对的，只要它符合URI的语法规则。

**URL**

URL（Uniform Resource Locator，统一资源定位器)它是一种具体的URI，即URL可以用来标识一个资源，而且还指明了如何locate这个资源。基本URL包含模式（或称协议）、服务器名称（或IP地址）、路径和文件名。URL类则不仅符合语义，还包含了定位该资源的信息，因此它不能是相对的，schema必须被指定。

# OTG

OTG（On-The-Go）是近年发展起来的技术，2001年12月18日由USB Implementers Forum公布，主要应用于各种不同的设备或移动设备间的联接，进行数据交换。特别是Pad、移动电话、消费类设备。改变如数码照相机、摄像机、打印机等设备间多种不同制式连接器，多达7种制式的存储卡间数据交换的不便。2014年左右开始在市场普及。

**产生背景**

USB技术的发展，使得PC和周边设备能够通过简单的方式、适度的制造成本，将各种数据传输速度的设备连接在一起。但都是通过USB连接到PC，并在PC的控制下进行数据交换。一旦离开了PC，各设备间无法利用USB口进行操作，因为没有一个从设备能够充当PC一样的主机。

**作用**

OTG技术就是在没有Host的情况下，实现设备间的数据传送。例如数码相机直接连接到打印机上，通过OTG技术，连接两台设备间的USB口，将拍出的相片立即打印出来；也可以将数码照相机中的数据，通过OTG发送到USB接口的移动硬盘上。野外操作就没有必要携带价格昂贵的存储卡，或者背一个便携电脑。

通过OTG技术，可以给智能终端扩展USB接口配件以丰富智能终端的功能，比如扩展遥控器配件，把手机、平板变成万能遥控器使用。

**分类**

根据USB接口所属协议，OTG可分为3类：

* USB2.0

Micro 5PIN：常见安卓手机OTG接口

Mini 5PIN：常见蓝魔等安卓平板OTG接口

* Micro USB3.0：三星NOTE3、Galaxy5高端手机OTG接口
* TYpe C：乐视手机Type C OTG接口

Micro 5PIN Mini 5PIN Micro USB3.0 TYpe C

# 开源许可证的一些介绍

## BSD 2-Clause license

BSD允许使用者修改和重新发布代码(以其他协议形式)，允许闭源商业发布和销售。BSD鼓励代码共享的同时，要求尊重代码作者的著作权。使用BSD协议，需要遵守以下规则：

1. 再发布的产品中包含源代码，则在源代码中必须带有原来代码中的BSD协议；
2. 如果再发布的只是二进制类库/软件，则需要在类库/软件的文档那个和版权声明中包含原来代码中的BSD协议。

## BSD (3-Clause) License

BSD允许使用者修改和重新发布代码(以其他协议形式)，允许闭源商业发布和销售。BSD鼓励代码共享的同时，要求尊重代码作者的著作权。使用BSD协议，需要遵守以下规则：

1. 再发布的产品中包含源代码，则在源代码中必须带有原来代码中的BSD协议；
2. 如果再发布的只是二进制类库/软件，则需要在类库/软件的文档那个和版权声明中包含原来代码中的BSD协议;
3. 不可以用开源代码的“作者/机构的名字”或“原来产品的名字”做市场推广。

## Apache v2 License

Apache Licence是著名的非盈利开源组织Apache采用的协议。该协议和BSD类似，同样鼓励代码共享和尊重原作者的著作权，同样允许代码修改，再发布（作为开源或商业软件）。需要满足的条件也和BSD类似：

1. 需要给代码的用户一份Apache Licence
2. 如果你修改了代码，需要再被修改的文件中说明。
3. 在延伸的代码中（修改和有源代码衍生的代码中）需要带有原来代码中的协议，商标，专利声明和其他原来作者规定需要包含的说明。
4. 如果再发布的产品中包含一个Notice文件，则在Notice文件中需要带有Apache Licence。你可以在Notice中增加自己的许可，但不可以表现为对Apache Licence构成更改。

Apache Licence也是对商业应用友好的许可。使用者也可以在需要的时候修改代码来满足需要并作为开源或商业产品发布/销售。

## MIT License

MIT许可证之名源自麻省理工学院（Massachusetts Institute of Technology, MIT），又称「X条款」（X License）或「X11条款」（X11 License）

MIT内容与三条款BSD许可证（3-clause BSD license）内容颇为近似，但是赋予软体被授权人更大的权利与更少的限制。

1. 被授权人有权利使用、复制、修改、合并、出版发行、散布、再授权及贩售软体及软体的副本。
2. 被授权人可根据程式的需要修改授权条款为适当的内容。
3. 在软件和软件的所有副本中都必须包含版权声明和许可声明。
4. 此授权条款并非属copyleft的自由软体授权条款，允许在自由/开放源码软体或非自由软体（proprietary software）所使用。此亦为MIT与BSD（The BSD license, 3-clause BSD license）本质上不同处。MIT条款可与其他授权条款并存。另外，MIT条款也是自由软体基金会（FSF）所认可的自由软体授权条款，与GPL相容。

## GPL v2

GPL 是GNU通用公共许可协议(GNU General Public License)的简称，我们很熟悉的 Linux 和 Git 就是采用了 GPL，该协议力图保障你分享和修改某程序全部版本的权利——确保自由软件对其用户来说是自由的。

所谓自由软件，强调自由，而非免费。GNU通用公共许可协议设计用于确保你享有分发自由软件的自由（你可以为此服务收费），确保你可以在需要的时候获得这些软件的源码，确保你可以修改这些软件或者在新的自由软件中复用其中某些片段，并且确保你在这方面享有知情权。

自由权利包括复制、分发和修改。源码是指所有修改作品及生成、安装、运行（对可执行作品而言）目标码所需的源码，包括控制上述行为的脚本，但其中不包括系统库、通用工具。

简而言之，如果你分发自由软件的副本，那么副本中必须包含许可协议和版权声明，并确保接收者能够获取到该副本的源代码及其依赖库的源码。

## GPL v3

GPL v3与GPL v2类似。区别在于，不仅要求用户公布修改的源代码，还阻止了其他一些私有化的方式，例如：不得将产品内的软件“tivo化”或“锁定”，或者（用行业内的话来说）“安全启动”，也就是说，不得以任何形式阻止用户修改产品内的以 GPL 许可协议发布的软件。

## Artistic License 2.0

Artistic License，又称艺术许可协议（英语：Artistic License），通常指最初的艺术许可协议（1.0版），是一个自由软件授权条款，主要用在官方发布的Perl解释器和大部分CPAN模块的授权。原始的艺术许可协议是由Perl的创始人Larry Wall编写发布的。

## Affero GPL

是一个广泛被使用的自由软件特许条款，最初由Affero, Inc撰写。此特许条款最新版本为“第3版”（v3），2007年11月发布。Affero 通用公众特许条款是改自GNU通用公众特许条款，并加入额外条款，其目的是为了Copyleft条款应用于在网络上运行的应用程式（如Web应用），从而避免有人以应用服务提供商方式逃避GNU通用公众特许条款。

## LGPL v2.1

LGPL是GPL的一个为主要为类库使用设计的开源协议。和GPL要求任何使用/修改/衍生之GPL类库的的软件必须采用GPL协议不同。LGPL允许商业软件通过类库引用(link)方式使用LGPL类库而不需要开源商业软件的代码。这使得采用LGPL协议的开源代码可以被商业软件作为类库引用并发布和销售。

但是如果修改LGPL协议的代码或者衍生，则所有修改的代码，涉及修改部分的额外代码和衍生的代码都必须采用LGPL协议。因此LGPL协议的开源代码很适合作为第三方类库被商业软件引用，但不适合希望以LGPL协议代码为基础，通过修改和衍生的方式做二次开发的商业软件采用。

GPL/LGPL都保障原作者的知识产权，避免有人利用开源代码复制并开发类似的产品

## Eclipse Public License v1.0

EPL允许使用者任意使用、复制、分发、传播、展示、修改以及改后闭源的二次商业发布。使用EPL协议，需要遵守以下规则：

1. 当一个代码贡献者将源码的整体或部分再次开源发布的时候,必须继续遵循EPL开源协议来发布,而不能改用其他协议发布.除非你得到了原“源码”拥有者的授权；
2. EPL协议下,你可以将源码不做任何修改来商业发布.但如果你要发布修改后的源码,或者当你再发布的是二进制文件的时候,你必须声明它的源代码是可以获取的,而且要告知获取方法；
3. 当你需要将EPL下的源码作为一部分跟其他私有的源码混和着成为一个Project发布的时候,你可以将整个Project/Product以私人的协议发布,但要声明哪一部分代码是EPL下的,而且声明那部分代码继续遵循EPL；
4. 独立的模块(Separate Module),不需要开源。
5. LGPL v3
6. 相对于LGPL v2，不仅要求用户公布修改的源代码，还阻止了其他一些私有化的方式，例如：不得将产品内的软件“tivo化”或“锁定”，或者（用行业内的话来说）“安全启动”，也就是说，不得以任何形式阻止用户修改产品内的以 GPL 许可协议发布的软件。

## Mozilla Public License Version 2.0

MPL是The Mozilla Public License的简写，是1998年初Netscape的 Mozilla小组为其开源软件项目设计的软件许可证。MPL许可证出现的最重要原因就是，Netscape公司认为GPL许可证没有很好地平衡开发者对 源代码的需求和他们利用源代码获得的利益。同著名的GPL许可证和BSD许可证相比，MPL在许多权利与义务的约定方面与它们相同（因为都是符合OSIA 认定的开源软件许可证）。但是，相比而言MPL还有以下几个显著的不同之处:

1. MPL虽然要求对于经MPL许可证发布的源代码的修改也要以MPL许可证的方式再许可出来，以保证其他人可以在MPL的条款下共享源代码。但是，在MPL 许可证中对“发布”的定义是“以源代码方式发布的文件”，这就意味着MPL允许一个企业在自己已有的源代码库上加一个接口，除了接口程序的源代码以MPL 许可证的形式对外许可外，源代码库中的源代码就可以不用MPL许可证的方式强制对外许可。这些，就为借鉴别人的源代码用做自己商业软件开发的行为留了一个 豁口。
2. MPL许可证第三条第7款中允许被许可人将经过MPL许可证获得的源代码同自己其他类型的代码混合得到自己的软件程序。
3. 对软件专利的态度，MPL许可证不像GPL许可证那样明确表示反对软件专利，但是却明确要求源代码的提供者不能提供已经受专利保护的源代码（除非他本人是 专利权人，并书面向公众免费许可这些源代码），也不能在将这些源代码以开放源代码许可证形式许可后再去申请与这些源代码有关的专利。
4. 对源代码的定义 而在MPL（1.1版本）许可证中，对源代码的定义是:“源代码指的是对作品进行修改最优先择 取的形式，它包括:所有模块的所有源程序，加上有关的接口的定义，加上控制可执行作品的安装和编译的‘原本’（原文为‘Script’），或者不是与初始 源代码显著不同的源代码就是被源代码贡献者选择的从公共领域可以得到的程序代码。”
5. MPL许可证第3条有专门的一款是关于对源代码修改进行描述的规定，就是要求所有再发布者都得有一个专门的文件就对源代码程序修改的时间和修改的方式有描述。

## Public Domain

Public Domain 是人类的一部分作品与一部分知识的总汇，可以包括文章、艺术品、音乐、科学、发明等等。对于领域内的知识财产，任何个人或团体都不具所有权益（所有权益通常由版权或专利体现）。这些知识发明属于公有文化遗产，任何人可以不受限制地使用和加工它们（此处不考虑有关安全、出口等的法律）。创立版权制度的初衷是借由给予创作者一段时期的专有权利作为（经济）刺激以鼓励作者从事创作。当专有权利期间届止，作品便进入公有领域。公有领域的作品由于没有专属权利人，因此公众有权自由使用它们。

# Windows清理

## C盘中有什么

刚刚安装的电脑系统 C 盘主要包含以下文件：PerfLogs 、Program Files 、Windows 和Users。如果你在文件夹选项当中显示隐藏的文件夹，就会发现，这里还有一个叫做 ProgramData 的文件夹，另外，64 位系统会多出一个 Program Files（x86）。

PerfLogs 就是 performance logs ，是系统的信息日志，其实现在的自动化机械、电脑系统、软件以及其他人工智能，都会拥有专门的日志文件，便于专业人员查看系统表现，查看具体问题和运行情况。你可以直接删除 PerfLogs 文件夹，当然不删除也没什么，因为它根本就不占用多少空间，呵呵哒。

Program Files 是软件的安装目录，对 64 位系统的电脑来说，32 位软件会默认安装在 Program Files（x86）文件夹当中，打开文件夹，每个安装的软件都拥有自己的目录。对于 Program Files 的清理，一般是删除你安装了但基本没有使用的软件，另外，这里边有一个 Common Files 文件夹，也是系统自动生成的插件、补丁以及程序运行所需要调取的公用文件夹，一般别去动它。且整个 Program Files 根本就不大，也就几个G，所以它绝不是清理的重灾区。

ProgramData 文件夹是程序的使用数据和设置文件，胡乱删除的话，会造成程序不能正常使用。同样我们看看文件夹属性，它也不大，并非清理重灾区。

Windows 开头的一般都是系统自带软件，Internet Explorer 就是 IE 浏览器，AMD 或 NVIDIA 字样是电脑显卡的相关软件，这个不了解没关系；其他软件基本上都是你自己安装的内容，比如Microsoft Office 15 ，Office 办公软件，包括 Word 、PPT 、Excel 等；这里边几乎存放了 Windows 系统运行所需的所有内容，查看文件夹属性会发现，它还挺大的，而且按照我们使用的情况来看，它会越来越大，所以，是清理的重灾区；

Users也是咱们的清理重灾区，它是存放我们所有用户文件的地方。这里边应该会有一个公用文件夹和你的账户文件夹，比如我的账户名称是 Chengy；如果你没有手欠的话，公用文件夹不会有什么内容；而账户文件夹到底装了些什么，让它看起来这么大。经过查看，罪魁祸首应该就是这个 AppData 文件夹啦。它应该算是清理重灾区。

综上所述，系统盘根目录下的 Windows 系统文件夹和用户目录下的 AppData 是我们要清理的两大重灾区。那么到底应该删除这两大重灾区的哪些文件，或者，还有没有其他更妙的清理方法呢？

## Windows 文件夹清理

系统文件夹存放了 Windows 运行所需的几乎所有文件，那么临时文件、系统备份、系统预读等文件也在包括在其中。理论上这些文件可以删除，并且能节省一定空间，但比起被占用的几十个 G 空间，简直九牛一毛；

另外，有的教程或文章教你关闭系统自动休眠、修改虚拟缓存、停止系统备份等等。这些我们都不提倡，这就好比你家里因为堆了太多东西，你就砸墙腾空间

Windows 文件夹成为重灾区的原因，其实主要与一个叫做 WinSxS 的文件夹有关。每一个版本的系统并非完美无缺，从你安装那一刻开始，系统开发公司还在不断地为这个系统缝缝补补或添加更加适宜的功能，所以才有了“打补丁”这一说。如此也才能防止电脑遭到某些病毒的侵袭、或因为某些漏洞而出现安全问题甚至崩溃。而这些升级内容和补丁，最后都安装在了一个地方，那就是 Windows 下的 WinSxS 文件夹。

让我们来看看它的大小。我的这台电脑是 Windows 10 的系统，还比较新，补丁不多，文件夹大小只有 6 个 G 左右，而大家可以看看自己的 Windows 7 系统，用了一段时间之后，这个文件夹一般都会在 10 个 G 以上。既然是系统补丁，咱们不能轻易把它删除，但其中有一个备份文件夹 Backup ，这里边的内容你是完全可以删除的。

另外，微软每过一段时间会把这些补丁做成优化的补丁包，就是 Service Pack ，简称 SP ，所以才有了 Windows7 SP1 , Windows7 SP2版本的说法。在“计算机”上单击右键，看看系统属性中有没有 Service Pack 的字样，如果没有，你可以通过百度或直接去微软官网下载安装，安装了补丁包之后，原先的囤积的补丁就会被替换，系统会腾出一大部分空间。

## Users目录下的 AppData 文件夹清理

当 Windows 系统刚刚安装到你电脑上的时候，它只是个普通的机器，但你使用一段时间之后，它其实就拥有了灵魂。否则为什么你会觉得它越来越顺手呢？主要原因就在这个 AppData 文件夹上。你在使用电脑的时候，你的电脑设置、软件设置、使用习惯等等，都被记录在这个文件夹当中。AppData 文件夹按照安全级别分为 Local ，Locallow 和 Roaming 三个文件夹。

* Local ，就是本地储存，也就是只有本地用户才能读取，其中有一个文件夹叫做 Temp ，对软件进行安装时，Windows 可能会帮你存留你安装时解压的源文件，方便下次安装直接调取使用，节省解压时间。但平时这东西很少用到的，很占空间，所以，可以直接删除。
* Locallow 文件夹顾名思义就是安全级别比 Local 文件夹 low 一点，假如你的工作环境里还有其他电脑，其他电脑就可能读取本地电脑的用户文件；
* Roaming 的意思是漫游，安全级别比前两个还低，这里的数据可以在工作环境的电脑之间漫游；

微软这么做，是为了让你办公更加方便，假如你原本在使用的工作环境中电脑A，你可以直接在工作环境中的电脑 B、C、D 中任何一台登录你的账户，你的 Roaming 文件夹中用户习惯就自动跑到另一个电脑去了，换台电脑会和使用原电脑一样顺手。

那么除开刚刚的 Temp 文件夹外，这三个文件夹中其他文件可以删除吗？坦率地说，除了 Windows 系统自带的内容，这些软件数据文件你都可以删除。只不过删除相应的软件数据，下次在使用该软件的时候，你心爱的电脑就啥也不记得了，虽然这样腾出了大大的空间，不过你还要从头调教它。

# 第三方登录

通过第三方授权授权来登录自家产品的方式，统称为第三方登录

## 三方登录是什么

第三方登录的实质第三方登录的实质，就是在OAuth2.0协议的基础上，利用外部的标识来获取帐号的内部标识然后完成登录的过程。具体的讲，就是将三方的帐号绑定到产品自身的帐号上，当查询到用户第三方的帐号已经绑定了平台的某个user\_id时，直接登录对应的帐号。

## 理解几个概念

外部标识

用来使用用户身份的标志，可以是用户名，手机号，邮箱等，每一个外部标识一定和一个内部标识相关联用以确定一个用户。需要注意的是，外部标识可能改变，例如小明12年用的是手机号A，17年用的是手机号B，在APP内更改了登录的手机号，就完成了外部标识的改变。所以，外部标识的作用有两个

* 让用户通过自己熟知且占有的外部标识来登录产品
* 可以通过校验外部标识来实现找回或转移数据资产（用户注册帐号后使用产品过程中产生的所有数据，例如在微信，数据资产是你的好友，历史聊天记录，所发的朋友圈；而在游戏中是你的游戏角色，拥有的装备，好友等）

内部标识

即产品中用于标识用户唯一性的标志，例如user\_id，必须有，不可更改且唯一，用户一般接触不到内部标识。当一个内部标识建立后，用户所有的数据资产都会绑定到这个内部标识上

user\_id

一个常用的内部标识，类似你的18位身份证ID

app\_id

用于区别不同APP的ID，具有唯一性

open\_id

第三方平台为了用户信息的安全，一般不会直接将用户的内部标识给到其他产品，而是选择了给一个外部标识，这个open\_id就是第三方平台给各个APP用以区分第三方平台用户身份的外部标识

需要注意的是，即使是同一个用户，第三方平台给不同的APP的open\_id也是不同的，这会导致一个问题。例如我有两款产品，一款美团，一款美团外卖。对第三方平台而言，这是两个app\_id，所以当用户授权美团使用第三方平台信息后，第三方平台给的是open\_id\_1，用户再来到美团外卖公众号，这时候还要再授权一次，第三方平台给的是open\_id\_2。

但实际上在美团的帐号体系中，这都是同一个用户，却绑定了两个open\_id，可能导致用户注册了两个帐号，数据不会互通，这显然是有问题的，要通过下面的union\_id解决

union\_id

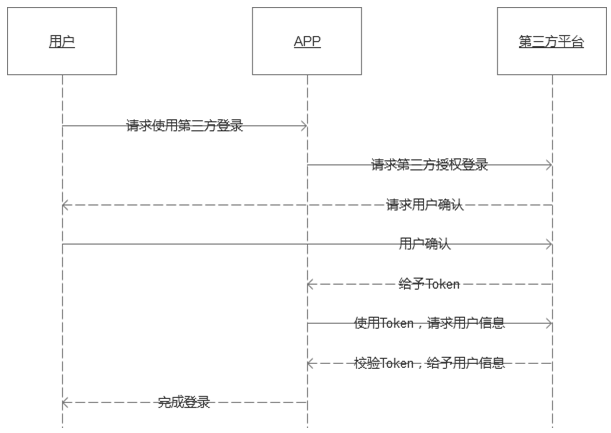
为了解决上面的问题，union\_id诞生了。不同的产品的可以使用同一个union\_id来确认用户的身份。对于使用同一套用户数据的不同产品，可以向第三方平台申请union\_id，当A产品获得了用户授权后，拿到该用户的union\_id，这时候再使用B产品再进行授权后，发现这是同一个union\_id，直接登录之前的帐户，避免了同个第三方平台用户注册了两个帐号数据不同步的问题。

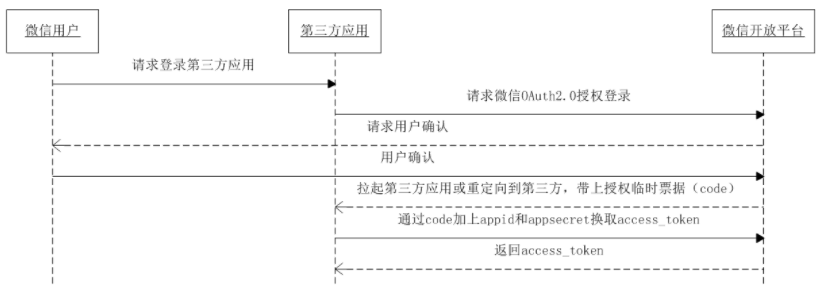
access\_token

可以理解为通行证，有了这个通行证，就能获取到第三方平台指定用户的有限信息。

OAuth 2.0

这个不太容易解释，我们简单的理解为授权的通用协议。举例：APP（快递小哥）需要从第三方应用那拿到你的资料（快件），显然第三方应用不可能把帐号和密码（你家的门禁卡）给APP，这时候提供一个Token（临时的小区门禁卡），让APP既能获取到你的用户信息（快件），又保证了安全。





## 为什么需要加入第三方登录

提高登录转化率

登录更加快捷，不需要输入密码在PC时代，登录不同的网站，为了避免网站被攻击后导致的用户数据泄露，自己会将网站分类，不熟的网站使用密码A，常用的网站使用密码B，重要的网站用密码C，极其重要的几个网站单独用密码DEF等等，但是记忆起来很费事。到了移动互联网时代，几大第三方平台已经成为了人手必备了产品了，特别是微信，简直可以看成移动互联网的用户身份证。

产品接入了第三方登录后，用户跳转到第三方应用，直接在授权页面点击授权即可进入产品，免去了输入密码的环节，能提高登录的转化率。

提高注册转化率

注册更加快速，方便获取用户信息第三方登录能简化注册流程，快速获取到用户信息，例如头像，昵称，性别，所在城市等等，让用户减少了信息填写的步骤，能更快的使用产品。

以微信小程序为例，对于不需要自建帐号体系的产品来说，有了微信授权后，仅仅需要点击一次“允许”就完成了帐号的注册，产品也能直接获取到用户的一系列信息，用户体验极好。

信赖感

有三方登录的产品，让用户觉得这个产品和大厂是有合作的，提高对产品的信赖感，最终能实现提高产品的登录和注册转化率

# doc和docx的区别

1. docx是Word2007以上版本的文件格式，docx格式的Word文件低于2007版就打不开了。要想在Word2003中打开docx文件，就必须得安装一个官方“office docx”补丁。
2. docx和doc的区别显著的一点就是体积大小的不同。docx格式其实就是一个zip文件，我们可以拿win rar打开docx文件，得到一堆的文件，很类似android程序或者win7桌面窗体的源码，用户在里面可以找到各种配置文件，文本文件和媒体文件。其原理就是相当于用两个文本文档，一个用来放文本信息，另一个用来配置个里面的格式，比如字体，大小等。这样用两个文本文件就能实现doc格式类似的功能，所以很节省体积。当我们创建一个空的文档文件的时候会发现，doc格式是25KB左右，而docx是0字节。
3. 由此可见，docx对比doc而言，可以节省大量的空间。这个特点对于我们一般的电脑而言貌似没有什么太大的用处，但是对于服务器、图书馆来说可有用途大了。docx至少能节省一半的空间（图片少的时候更明显），这么一来，图书馆的电子书就能多一倍，其效果由此可见一斑！
4. docx与doc的另一个区别是响应速度。比如在docx文件里，很多特效之类的东西只要修改配置文件就可以了，而不用像doc那样反复修改那个大文件
5. 还有一点要提的就是：doc格式是微软专用格式，并没有对外完全授权，其他office，比如wps、永中、open office等虽然能打开doc格式，并且大部分都是没有问题的，但是在极少地方会有错误（格式或字体，或者行间距之类的）。