音视频基础面试题

面试题1： 老师，rtmp延迟比较高，需要做什么延时处理吗?

RTMP基于TCP的，有1~4秒延迟正常。但我们也可以降低延迟。
1 可以降低编码视频的分辨率，码率，这样可以降低数据量。这些参数在编码的时候可以调整。
2 直播在某些应用场景比如安防监控领域，也可以降低帧率，比如从30帧降低到20帧左右。
3 将I帧间隔提升到3秒左右。 增加P帧，B帧数量。采用硬件编码提升编码效率.
4 单帧数据量限制到1个MTU以下。如果I帧数据量太大，可以分成小于1个MTU的包传输。优化数据发送接收缓冲区，接收缓冲区大小。
5 推流部分可以采用基于UDP的协议传输。能够起到降低延迟的目的。拉流部分也可以转化为WEBRTC，这样用户可以拉WEBRTC流。
6 使用硬件解码，在IOS上 安卓，嵌入式上，有解码加速芯片的PC上，可以使用对应平台的硬件解码，降低解码延迟。
7 优化图像渲染。优化CPU GPU渲染过程中的效率.

面试题2： 您的网络音视频项目方面开发经验一定很丰富吧!？您原来的项目视音频互动支持多少路？直播支持多少路呢？延迟率怎样？噪音、杂音怎么处理的?

视频互动P2P WEBRTC支持不了多少，受限于网络带宽，理论上，如果我们平均网速是300Mb+，那么除以8，就是40MB左右，如果每个用户分配512KB下行，那么理论上可以支持60~80人同时进行视频会议直播通话。 根据实际网络情况是，WEBRTC SFU架构平时也就支持十几路同时视频通话，会议。延迟一般1秒之内。噪音杂音使用WEBRTC的软件回声消除算法。

一般的情况下，常见的SUF架构建议开个十几路，也就是十几个人同时视频会议，比较常见，超过20人，估计有的人就会卡了。

这是怎么算出来的呢？ 因为这个时代，我们有些人的手机网络，有很大一部分还是4G。4G的下载速度，说的是100M。 打个折 80M。 除以8，也就10MB。10MB，给每个用户分配512~1M ,所以就十几路同时视频会议通话。

如果每个用户都是300M带宽。 极限情况，每个人分配512KB。就能容纳60~80人同时会议。

但实际情况是，我自己的电脑上万快，我拉的宽带是一个月400块的1000Mb电信宽带. 但我的网卡只支持100Mb。 所以我台式机，有线网，也仅仅只有几十兆b网速。绝大多数人还是百兆网卡。

注意这里 电信，移动公司给你的带宽，一般说的是小b, 比特。而真正我们常用的是字节，也就是大B。
webrtc 一般有三种架构， mesh架构，就是p2p，适合4个人左右的视频会议通话。
MCU架构适合多人视频会议，但服务器压力很大，成本很高。
SFU架构服务器只负责转发，服务器成本小一些。
具体选择什么架构，甚至一些融合性的架构，根据实际情况来选择。

面试题3： 简述LINUX网络编程中，select send/recv 函数的返回值以及含义。

1: select 超时返回 0 有数据大于0 socket出错返回 小于 0 。
2：recv函数 超时 返回小于0 。 有数据大于 0 。 关闭socket 返回值等于0.
socket如果对方非正常关闭，就是没调用close的关闭，我方再去访问socket 会产生一个sigpipe信号，
如果不处理sigpipe信号，会导致我方程序崩溃。

面试题4： 请你详细解释下 PTS 跟DTS的关系?

答： pts必须大于等于dts。 因为展示图片的时间 不可能发生在解码之前。没有B帧，PTS DTS顺序是一致的。有了B帧，因为B帧前后引用，所以它的DTS(解码时间) 就不是按顺序的。 那如何做到先进来的帧后解码呢？其实FFmpeg内部有缓冲的.

面试题5： 简述 void glLoadIdentity(void) 函数的意义。

glLoadIdentity 命令用一个单位矩阵来替换当前矩阵，对当前矩阵进行初始化。无论以前进行了多少次矩阵变换，执行该命令后，当前矩阵恢复为一个单位矩阵，就是没有进行任何矩阵变换状态。

面试题6：简述P2P跟流媒体转发区别,各自优劣?

P2P是一套经济实惠的解决方案，流媒体转发成本高。P2P受限于家庭网络上行带宽,
支持同时在线观看的人数比较少,一般的情况下支持4~8个人同时看。
流媒体服务器转发, 相对于p2p成本高，优势是支持服务器扩展，百万人同时在线看。支持录制回放。

面试题7: 一台服务器能够支持多少人同时在线看视频流?

根据服务器带宽，服务器性能，每一个拉流的客户端分配的数据流大小，
来决定，一台服务器能服务多少人。

面试题8： 同一个路由器下的两个App，如何发现对方，并建立通讯的详细过程?

通过UDP广播发现对方，得到IP地址端口，然后建立通讯连接。
1 App1发广播消息。
2 App2 收到这个广播。
3 App2 回信息，告诉端口信息。
4 App1 得到了IP地址，端口。
5 App1 用这个IP地址+端口，建立socket通讯连接。

**面试题9：App 通过socket发送一个命令数据。但是发送的时候特别卡，
这个时候你如何去做，优化这个App不卡?**

我们可以做一个队列，主线程发送的时候，直接将数据加入到队列中，
另外开一个线程专门发送数据，如果有命令过来，它就发，如果没有，
他就等有数据再处理。

面试题10：简述音视频开发中，比特率，位深，通道之间的关系。

比特率 bit = 采样 x 位深 x 通道

面试题11：请问音视频领域为啥会采用YUV，YUV对比RGB的优势是什么?

YUV常见的有 YCbCr YPbPr 。YUV主要用于电视中。 YCbCr用户数字视频，图像压缩传输。Y表示亮度(luma),CbCr表示色度(chroma)

相比于RGB, YUV设计目的是为了编码，传输方便，减少带宽等。

人眼视觉的特点是对亮度信息更加敏感，对色彩信息相对不敏感。
因此我们在视频编码过程中，我们可以存储更多的亮度信息，保存较少的色差信息.

常见的YUV格式 YUV444 YUV422 YUV420

面试题12：简述图像缩放算法，各自的优劣

最邻近插值 速度快 效果差，容易产生锯齿。 双线性插值 效果中等 速度中等。双三次插值 效果最好 速度最慢。

面试题13：什么是CBR，什么是VBR，各自应用场景是?

视频码率有两种： 一种是固定码率CBR（Constants Bit Rate),一种是可变码率 VBR (Variable Bit Rate). CBR 非常适合流媒体直播 .VBR适合点播 视频转码 等应用.根据你的使用场景选择具体的技术.

面试题14：H.264 H.265是有损编码还是无损编码?

H.264 H.265都是有损编码.

面试题15：H.264相对于YUV420 压缩比是多少?

这个问题其实没有一个具体的值. 视频压缩比是由很多因素决定的. 包括视频内容的复杂度,分辨率,帧率以及一些编码参数来决定.一般情况下H.264支持百倍级别的压缩。使用H.264进行编码的时候,比如使用x264编码或者嵌入式设备进行H.264编码,通常可以通过调整这些编码参数来控制压缩比. 常见的编码参数比如码率, 量化参数等. 通过提高量化参数或者降低码率，都可以实现更高的压缩比,但这也会导致图像质量的损失.

面试题16：简述H.264压缩编码原理。

H.264 编码主要经历以下方法步骤,
1 帧内预测 减少空间冗余 I 帧
2 帧间预测 减少时间冗余 P B 帧
3 DTC 变换, 量化 减少视觉冗余
4 熵编码

面试题 17：简述 H.264 压缩编码流程。

通过帧内预测,帧间预测去除空间 时间上的冗余,得到小很多的残差块.
再通过 DTC 变换将低频信息和高频信息分开得到变换块，然后再对变换块的系数做量化,高频系数较小容易量化为 0,
同时人眼对高频信息不是很敏感，这样我们就得到一个含有连续值为 0 的像素串。
最后通过熵编码把图像压缩成很小的数据.

面试题 18：如何理解 H.264 编码中的 I 帧?

I 帧 I Frame 帧内编码图像，关键帧 是视频的第一帧，GOP 的第一帧. I 帧对整张图像进行编码.
仅仅用 I 帧就能解码出完整的图像.I 帧是一种自带全部图像信息的独立帧，无需参考任何其他帧就能够进行独立解码。
简单理解为他就是一张完整的图像。

面试题 19：如何理解 H.264 编码中的 P 帧?

P 帧 P Frame 预测编码图像，前置引用,非关键帧.
P 帧不会对整张图像进行编码,以前面的 I 帧或 P 帧作为参考帧,只编码当前 P 帧与参考帧之间的差异数据.
P 帧需要先解码出前面的参考帧,再结合差异信息解码出当前 P 帧完整的图像. P 帧允许引用多个参考帧.

面试题 20：如何理解 H.264 编码中的 B 帧?

B 帧 B Frame 前后预测编码图像，双向引用预测帧 ，非关键帧.
B 帧不会对整张图像进行编码,它是双向引用预测帧，要参考其前一个 I 或者 P 帧及其后面的一个 P 帧来生成一张完整的图片。
压缩比更高.可参考的帧变多了，只需要存储更少的差异数据。
B 帧需要先解码出前后的参考帧,再结合差异信息解码出当前 B 帧完整的图像.

面试题 21：什么是 GOP？
Group of Picture 从一个 I 帧到下一个 I 帧之前的一组视频帧.
GOP 中的帧可以分为 3 种类型：I 帧 P 帧 B 帧.

面试题 22：什么是 IDR 帧？

当遇到 IDR 帧的时候，会清空参考帧队列。如果一个 GOP 序列出现错误，遇到 IDR 帧就可立即得到修复,

IDR帧之后的帧，不会参考IDR帧之前的帧。 写视频播放器的时候，seek到指定位置，播放器就会选择最近的IDR帧,
然后开始解码播放。

面试题23：GOP有哪两种类型？

GOP 也可以分为两种类型. Open GOP Closed GOP
Open GOP 可以跨GOP引用 Closed GOP 不可以跨GOP引用
Open GOP 允许前一个GOP的B帧参考下一个GOP 的I帧，更加复杂。
Closed GOP 前一个GOP的B帧不能参考下一个GOP的I帧.
在closed GOP中有一种特殊的I帧，叫IDR帧,即使解码刷新帧.

面试题23：什么是向量点乘 它的几何意义是?

向量的点乘,也叫向量的内积,数量积. 点乘的结果是一个向量在另外一个向量上的投影长度.
点乘的几何意义可以用来计算两个向量之间的夹角.以及一个向量在另外一个向量上的投影.

面试题24：什么是向量叉乘 它的几何意义是?

向量a和向量b的叉乘(外积)结果是一个向量，有个更通俗易懂的叫法是法向量，该向量垂直于a和b向量构成的平面。cross(vec3 a, vec3 b);在二维空间中，
外积还有另外一个几何意义就是：|a×b|在数值上等于由向量a和向量b构成的平行四边形的面积。

面试题25：如何判定向量差乘的方向?

方向判定：向量c的方向与a,b所在的平面垂直，用“右手法则”判断
1.大拇指垂直于四指指向的方向；
2.四指与A旋转到B方向一致，大拇指指向为C向量的方向。

面试题26：矩阵乘法满足的条件是?

两个矩阵的乘法仅当第一个矩阵A的列数和另一个矩阵B的行数相等时才能定义。
如A是m×n矩阵和B是n×p矩阵，它们的乘积C是一个m×p矩阵 。

面试题27：矩阵乘法是否满足交换律?

矩阵乘法不满足交换律

面试题28：什么是逆矩阵?

设A是一个n阶矩阵，若存在另一个n阶矩阵B，使得： AB=BA=E ，则称方阵A可逆，
并称方阵B是A的逆矩阵。单位矩阵的逆矩阵是它本身。

面试题 29：一般用什么方法求解逆矩阵?

1 初等变换求逆矩阵 2 根据伴随矩阵求解逆矩阵

面试题 30：什么要引入齐次坐标。

齐次坐标是一种用 N+1 个数字来表示 N 维坐标系的方法。
我们希望矩阵变换都统一写成线性变换的形式。不能让平移变换成为
一种特殊的变换. 就需要寻找一种方法,把平移缩放旋转变换统一表示。
这个时候，我们就引入一个新的东西，叫齐次坐标.
在齐次坐标的表示下, 平移 旋转 缩放 矩阵．利用齐次坐标坐标，我们就
可以把不同的变换写成统一的表示形式.

面试题 31： Android OpenGL ES 渲染跟 GUI 界面同一个线程吗?

在 Android 中，OpenGL ES 的渲染通常是在单独的渲染线程中执行的，而 GUI 则由主线程负责。
这样做的好处是可以避免渲染操作对 UI 的影响，同时也可以提高渲染的效率和流畅度。
因此，为了避免出现渲染和 UI 界面操作之间的竞争和冲突，推荐将 OpenGL ES 的渲染操作放在单独的线程中进行处理。

别的 iOS Qt，Opengl(ES) 渲染都是跟 GUI 在一个主线程的。 安卓这里是完全不一样的。

面试题 32：H.264 编码采用 Golomb(哥伦布) 编码还是 huffman(哈夫曼) 编码?

H.264 Golomb 编码和 Huffman 编码都有用到。

面试题 33：OpenGL ES 用 stb_image 加载渲染正常的 png 图片没问题，但从网上随便下载一张 jpg 图片渲染出错，图片能显示，但是显示各种小格子，请问主要是什么问题?

这种情况一般是图片格式问题，如果读取图片用的 rgba 四个通道，但是从网络下载的图片只有三个通道 rgb，一般会出现这种问题。

面试题 34：OpenGL 渲染的图片骗色，偏蓝色，请问一般是什么问题?

如果渲染的是 RGB 纹理，一般是纹理的排列顺序问题。如果是 YUV 纹理，一般是 UV 分量顺序出错。

面试题 35: 旋转矩阵的特点是什么?

1 旋转矩阵是正交矩阵，其列向量 (或行向量) 构成的基向量组成了一个正交归一的空间.
2 旋转矩阵它的逆等于它的转置。

面试题 36: 向量点乘的几何意义?

1. 计算投影：点乘可以用来计算一个向量在另一个向量方向上的投影。具体而言，在二维空间中，给定两个向量 $\mathbf{A}$ 和 $\mathbf{B}, \mathbf{A} \cdot \mathbf{B}$ 的结果等于向量 $\mathbf{A}$ 在向量 $\mathbf{B}$ 方向上的投影长度 (或投影的倍数)。通过点乘的结果可以判断向量是否在同一方向或相反方向。
2. 判断垂直性：如果两个非零向量的点乘为 0，即 $\mathbf{A} \cdot \mathbf{B} = 0$，那么这两个向量是垂直的。这是因为点乘结果为 0 表示两个向量之间没有共享的方向分量，也就是说它们在方向上正交或垂直。
3. 计算夹角：点乘还可以用来计算两个非零向量之间的夹角。通过计算 $\mathbf{A} \cdot \mathbf{B} / (||\mathbf{A}|| * ||\mathbf{B}||)$，可以得到两个向量之间的余弦值。然后可以使用反余弦函数 (arccos) 将余弦值转换为夹角的弧度值。
4. 判断同向与反向：根据点乘的正负可以判断两个向量是同向还是反向。如果点乘结果为正值，即 $\mathbf{A} \cdot \mathbf{B} > 0$，表示两个向量在方向上同向；如果点乘结果为负值，即 $\mathbf{A} \cdot \mathbf{B} < 0$，表示两个向量在方向上反向。

面试题 37: C++ 中为什么使用 extern "C"他的意义是什么?

C++ 和 C 语言在函数调用约定和名称修饰等方面存在差异。C++ 支持函数重载和类的成员函数等特性，因此编译器会对函数名进行修饰以区分不同函数。而 C 语言没有函数重载和名称修饰的概念，函数名直接就是符号名称。
使用 extern "C" 可以使得 C++ 函数使用 C 语言的链接规范，避免了在 C++ 代码中调用 C 语言函数时出现链接错误的问题。具体来说，使用 extern "C" 会关闭 C++ 的名称修饰机制，使得函数名按照 C 语言的方式进行编译和链接。

面试题 38: C++ 一个空类的大小是多少?

空类的大小取决于编译器的实现。根据 C++ 标准规定，空类的大小至少为 1 字节，以确保每个对象在内存中具有唯一的地址。
虽然空类不包含任何成员变量，但它可能会包含一些隐藏成员，如虚函数表指针 (vptr) 或用于实现多态性的其他信息。这些隐藏成员可能会增加空类的大小。
此外，编译器还可能会对空类进行内存对齐，以满足特定平台的要求。因此，即使一个类没有显式的成员变量，其大小可能也会大于 1 字节。
综上所述，空类的大小通常为 1 字节或更大，具体取决于编译器和平台的实现细节。你可以使用 sizeof 运算符来获取空类的大小。

面试题 39: 为什么要使用结构体字节对齐?
结构体字节对齐是为了优化内存访问的效率和保证处理器的正确性。
当结构体中的成员变量不按照自然对齐方式排列时，处理器需要进行多次内存访问来获取完整的数据。这样的内存访问会降低程序的性能，特别是在处理大量数据时，这种影响会更加明显。

字节对齐的原则是将结构体的每个成员变量放在适当的内存地址上，使得所有成员变量都能够被高效地访问。通过设置适当的对齐规则，可以消除处理器的额外内存访问，并提高程序的运行效率。此外，还可以避免一些因内存对齐不当而导致的处理器错误，例如内存泄漏、越界访问等问题。
C++中默认的字节对齐方式通常是4字节对齐或8字节对齐，但可以使用#pragma pack(n)指令来改变字节对齐方式，其中n表示对齐大小。需要注意的是，改变字节对齐方式可能会增加内存消耗，因此应该谨慎使用。

面试题40: H.264 Open GOP 和closed GOP区别
H.264 编码中，GOP (Group of Pictures) 可以分为两种类型：Open GOP 和 Closed GOP。
Open GOP 是指 GOP 的第一帧是 B 帧或 P 帧，也就是说，第一帧不是 I 帧。这种方式可以提高编码的效率和压缩比，但会使得 GOP 之间的切换更加困难，容易出现画面断层的问题。
Closed GOP 是指 GOP 的第一帧是 I 帧。这种方式可以保证每个 GOP 都是自包含的，不依赖于其他 GOP 进行解码。这样可以实现更好的随机访问性能和错误恢复性能，但会对编码效率和压缩比产生一定的影响。
在实际应用中，选择 Open GOP 还是 Closed GOP 取决于具体的场景和需求。如果需要实现较高的压缩比和编码效率，并且可以容忍一定的画面断层问题，可以选择 Open GOP。如果需要更好的随机访问性能和错误恢复性能，并且可以牺牲一定的编码效率和压缩比，可以选择 Closed GOP。

注意市面上大多数使用的都是Closed GOP。

面试题41: H.264和H.265编码算法的区别
H.264和H.265是两种视频编码标准，它们使用了不同的编码算法来实现视频的压缩和解码。以下是它们主要的区别：

1. 压缩效率：H.265相对于H.264具有更高的压缩效率。H.265采用了更多先进的编码技术，如更强大的运动估计、变换和预测算法等，以提供更好的图像质量和更低的比特率。相同画质的视频，使用H.265编码可以获得更小的文件大小。
2. 编码单元：H.264编码将视频帧分割为宏块 (macroblock)，而H.265将视频帧分割为更小的最小编码单元 (CU，Coding Unit)。这样做可以更好地适应视频内容的复杂性，并提供更精细的运动估计和编码。
3. 运动估计：H.265引入了更复杂和精确的运动估计算法，例如利用更多的参考帧进行运动搜索和估计。这使得H.265能够更有效地处理视频中的运动场景，并提供更好的压缩效果。
4. 变换和量化：H.265采用了更高级的变换和量化算法，如高效的多方向变换和可变分辨率量化。这些技术能够更好地处理视频细节，并提供更好的压缩效率。
5. 熵编码：H.265引入了更高效的熵编码算法，如双重索引编码 (CABAC) 等。这种编码方法可以更好地利用统计特性和上下文信息，以提供更高的压缩效率。

总的来说，H.265相对于H.264在压缩效率方面有了很大的改进。它采用了更多先进的编码技术和算法，以提供更好的图像质量和更低的比特率。然而，由于H.265的复杂性和计算要求更高，它的编码和解码过程可能需要更多的计算资源。在选择使用哪种编码标准时，需要权衡压缩效率、设备兼容性和计算资源消耗等因素。

面试题42: H.264 SPS PPS
在H.264标准中，SPS (Sequence Parameter Set) 和PPS (Picture Parameter

Set) 是两个重要的参数集，它们的作用如下：

1. Sequence Parameter Set (SPS)：SPS用于描述视频序列的全局参数信息，包括视频的编码格式、分辨率、帧率、图像大小、颜色空间等。每个视频序列只需要一个SPS进行描述，它定义了整个视频序列的基本特征。SPS中包含的参数信息可以被多个视频帧所共享。
2. Picture Parameter Set (PPS)：PPS用于描述特定视频帧的参数信息，包括参考帧列表、解码器参数、图像编码方式等。每个视频帧都需要对应一个PPS进行描述，以指定其特定的参数设置。不同的视频帧可能有不同的PPS。

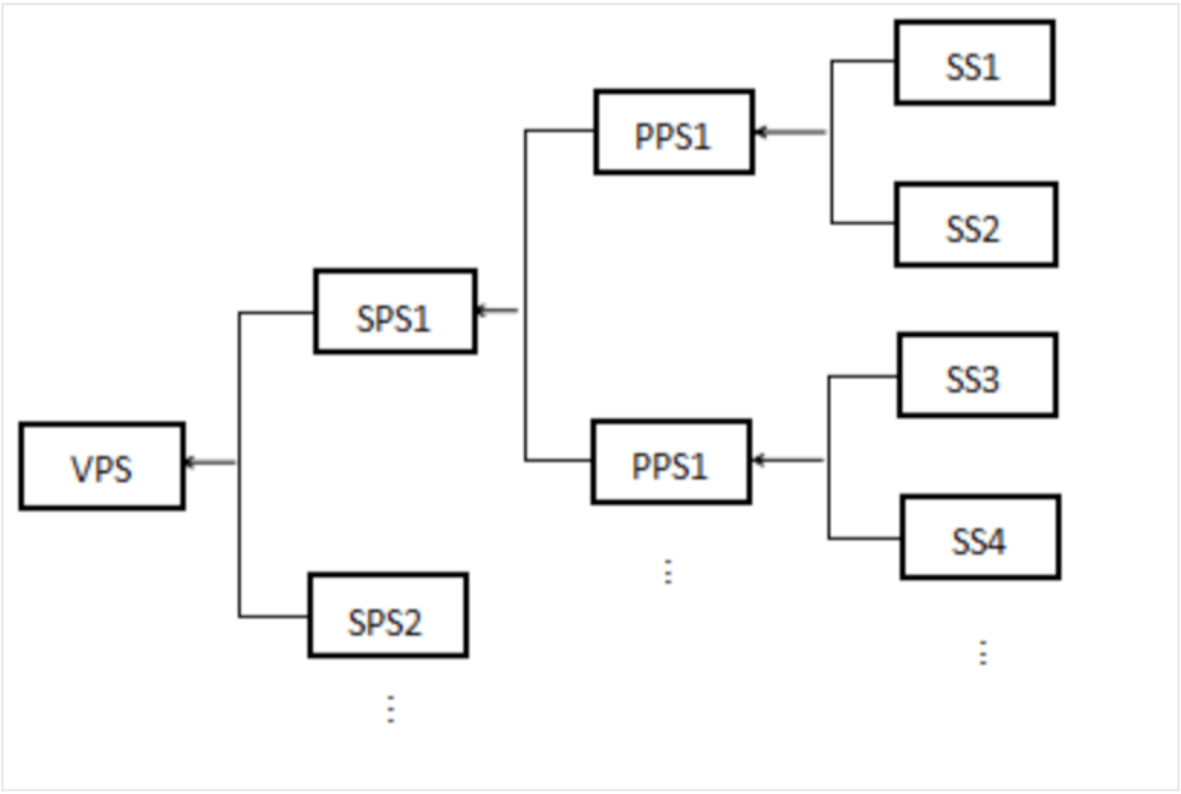
面试题43: H.265 VPS SPS PPS之间的关系

在H.265标准中，VPS (Video Parameter Set)、SPS (Sequence Parameter Set) 和PPS (Picture Parameter Set) 是三个不同的参数集，它们之间存在一定的关系和依赖。

具体而言，VPS用于描述视频序列的全局参数信息，包括视频序列的层次结构、图像层次结构、参考帧列表等。每个视频序列只需要一个VPS进行描述。而SPS和PPS则分别用于描述视频序列中的特定帧的参数信息，如解码器参数、参考帧列表、图像大小等。每个视频帧可以有对应的SPS和PPS，以指定其特定的参数信息。

SPS和PPS都依赖于VPS。具体而言，SPS和PPS中会引用VPS中的某些参数，如图像层次结构、参考帧列表等。这些参数的引用需要使用VPS的ID号来标识。因此，在解码过程中，解码器需要先解析VPS，然后才能正确解析SPS和PPS中的参数信息，以正确地解码视频帧。

总之，H.265中的VPS、SPS和PPS是三个不同的参数集，它们之间存在一定的关系和依赖。VPS用于描述视频序列的全局参数信息，SPS和PPS用于描述视频帧的特定参数信息，并且都依赖于VPS。在解码过程中，需要先解析VPS，然后才能正确解析SPS和PPS中的参数信息。



44 gl_Position 是投影后的空间坐标还是裁剪后的空间坐标?
gl_Position 表示的是顶点经过模型视图投影变换后的裁剪空间坐标，是顶点着色器中非常重要的变量之一，用于确定顶点在屏幕空间中的位置。

45 什么是 MVP 矩阵?
MVP 矩阵是图形学中常用的一种矩阵，用于将一个顶点从模型空间 (Model Space) 转换到世界空间 (World Space)，
然后经过投影变换 (Projection Transform) 进入裁剪空间 (Clip Space)。MVP 矩阵是模型-视图-投影 (Model-View-Projection) 矩阵的缩写。

46 请问 OpenGL 使用的是行主序还是列主序的矩阵?
在 OpenGL 中，通常使用列主序 (column-major order) 的矩阵。这意味着矩阵中的元素是按列存储的，而不是按行存储的。
在 OpenGL 中，矩阵通常表示为一个一维数组，按列主序排列。例如，一个 4x4 的矩阵在内存中的存储顺序如下：

[0] [4] [8] [12]
[1] [5] [9] [13]
[2] [6] [10] [14]
[3] [7] [11] [15]

47 请问如何实现多路视频流画面拼接功能，类似实现画中画的功能?
多路视频流画面拼接，效率最高的方式是使用 OpenGL 来实现。 一屏幕多窗口，如果只显示，可以用多个 OpenGL 对象，每个对象彼此独立。
如果需要显示 + 编码录制功能，并且录制出来也是多窗口画中画。 就只能用一个

OpenGL对象来实现。 将解码后的所有纹理传给OpenGL，
然后每个纹理根据UV划分到不同的小窗口。实现画中画功能。 渲染的时候，渲染到fbo，就可以通过pbo把数据从gpu取出来，然后编码
成和H.264，然后写成mp4。 fbo的纹理，同时也可以渲染到窗口显示出来.
传统的FFmpeg也能实现，但实现效率相对来说低一些。

48 H264的间隔码001，0001为什么不会跟帧数据混合在一起?
H.264编码时，在每个NAL前添加起始码 0x000001，解码器在码流中检测到起始码，当前NAL结束。
为了防止NAL内部出现0x000001的数据，h.264又提出‘防止竞争 emulation prevention"机制，在编码完一个NAL时，
如果检测出有连续两个0x00字节，就在后面插入一个0x03。当解码器在NAL内部检测到0x000003的数据，就把0x03抛弃，恢复原始数据。

49 如何实现视频的倍速播放?
在喇叭扬声器初始化参数不变的条件下， 可以通过重采样的方式。 将输出的采样率降低为原来的一半。 这样导致输出的数据也为原来的一半，
就可以实现倍速播放，但是这种方式会导致声音变快变尖锐。 另外一种方式，可以通过算法将原来的pcm数据每隔一个字节进行均值合并，
或者直接每隔一个字节丢掉，这样pcm数据也会减半，达到不变声倍速播放。

50 如何实现视频倒放?
第一种方式可以将视频全部解码后，按照时间戳保存将解码后的图像定义成一种带时间戳的临时图像文件保存到磁盘上。
播放的时候，按照倒叙，处理后的图片时间戳来播放。抖音剪映app用的这种方式。
第二种，直接按照backward的方式seek最后一帧时间戳，就可以seek到最后一个pts到最后一个I帧之间的数据帧，
然后进行解码后倒叙播放。 不断重复这个过程。

51 select 与 epool的区别是什么

I/O 模型不同：select 使用轮询模型，每次遍历所有的文件描述符集合，效率低下；而 epoll 是基于事件驱动的模型，通过回调函数只处理活跃的文件描述符，效率更高。
文件描述符数量限制不同：select 的文件描述符数量被限制在 1024 左右，对于大规模的连接，需要使用 fd_set 多次扩展或者使用其他的多路复用技术；
而 epoll 可以支持数万个文件描述符，因此适用于高并发场景。
触发方式不同：select 和 epoll 对可读事件和可写事件的触发方式不同。select 只能检测是否可以从文件读取数据或将数据写入文件，
无法区分数据是进入还是出去，无法保证数据完整性；而 epoll 通过边沿触发 (ET) 模式，只有当状态发生变化时才会触发事件，能够确保数据完整性。

内核实现机制不同：select 是系统调用，每次调用都需要把全部描述符集合从用户态传输到内核态，并且需要判断每个描述符是否就绪；
epoll 则是内核空间与用户空间共享一块内存，只需要在内核空间注册一个事件表，然后等待事件的通知即可，不需要对已注册的文件描述符进行遍历。

52 epool 水平触发 与边缘触发的区别

水平触发和边缘触发的最大区别在于事件通知的时机。水平触发会一直通知文件描述符处于就绪状态，直到应用程序处理完为止；

而边缘触发只在状态变化时通知一次，需要应用程序自行保证数据的完整性。边缘触发可以减少事件通知次数，提高效率，但同时要求应用程序更加谨慎地处理事件以确保数据完整性。

水平触发模式适合于使用阻塞 I/O 的情况，而边缘触发模式适合于使用非阻塞 I/O 和基于事件驱动的编程模型。边缘触发模式对应的处理方式更加高效，但对编程模型的要求也更高。