1. F；webrtc的安装和vs2015安装介绍
2. 下载depot\_tools，解压后把depot\_tools添加到环境变量Path。
3. 运行命令行，键入gclient，会自动下载svn，python等。
4. 从百度云网盘、360网盘等国内资源 (由于数据量很大，且需要翻墙) 下载webrtc2015源码。
5. 修改控制面板--》语言--》区域--》管理--》非Unicode程序为English。(否则编译报错)
6. 运行命令行，

键入set DEPOT\_TOOLS\_WIN\_TOOLCHAIN=0，键入python webrtc\build\gyp\_webrtc.py。(webrtc使用gyp构建项目，而实际编译则是由ninja完成)

注意:此处若需要编译成64位的库，则需要添加键入set GYP\_DEFINES=target\_arch=x64。

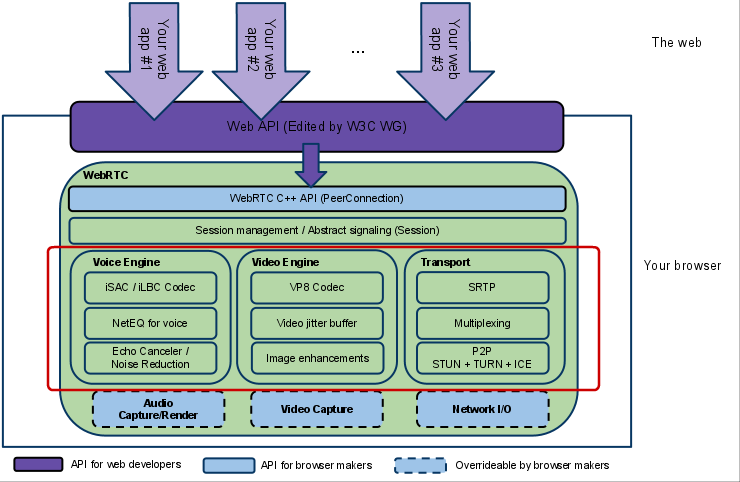
1. 键入ninja -C out/Debug，完成编译。
2. 安装vs2015。
3. 产生vs2015项目。

键入set GYP\_GENERATORS=ninja,msvs-ninja，set GYP\_MSVS\_VERSION=2015，python webrtc/build/gyp\_webrtc.py，产生了vs工程，但整个项目的编译还是由ninja完成。

//编译webrtc2014出现好多问题//

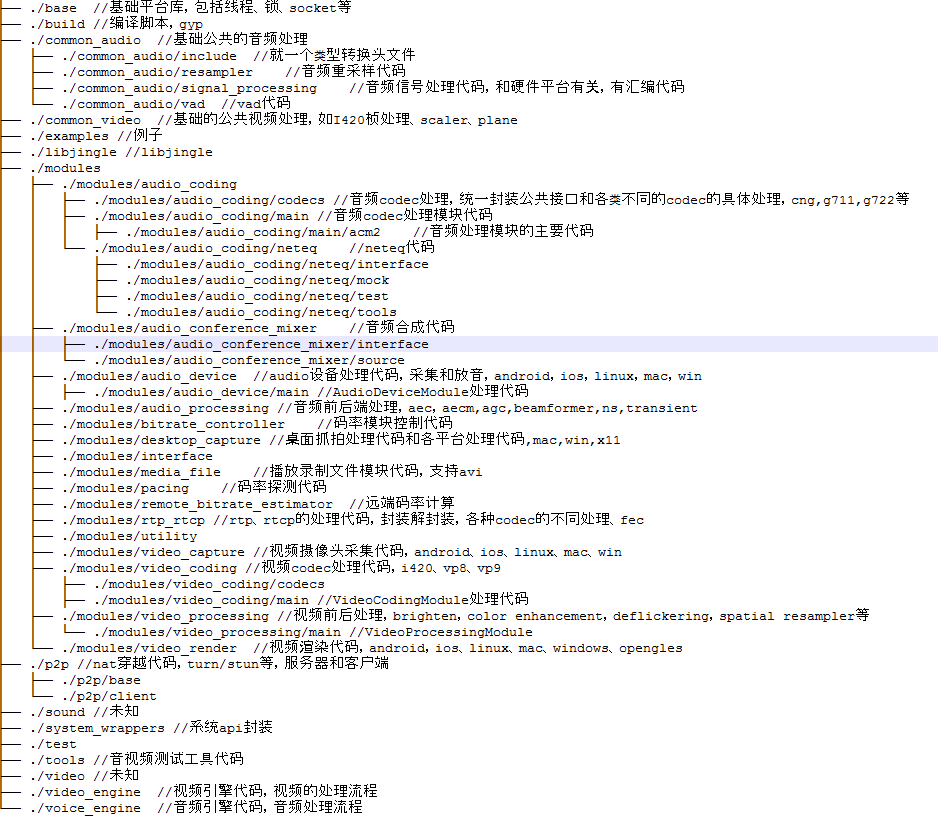
1. webrtc整体结构介绍

1.整体框架图



**Transport / Session**(传输/会话层):会话层组件采用了libjingle库的部分组件实现。

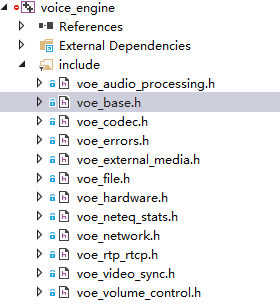
2.代码目录结构



方便后期查阅代码。

1. 从webrtc的voiceengine的demo学习voiceengine
2. Voice\_engin对外接口

1）对外暴露的头文件：



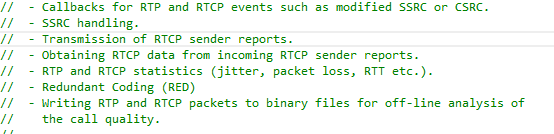
Voe\_base.h:支持G.711编码的、RTP传输的全双工VoIP应用，若要支持其他编码器，则需要VoECodec的支持，初始化和销毁VoiceEngine实例，通过文本或回调函数记录trace信息，支持多channel（mixing或发送至多目标地址）。

Voe\_codec.h:vad监测。支持其他编码。

voe\_audio\_processing.h:噪声抑制(Noise Suppression)，自动增益控制(AGC)，回声控制(EC)，接收端的VAD、NS、AGC，语音、噪声、回音level的测量，audio processing调试信息的生成与记录，检测键盘动作。

Voe\_neteq\_stats.h: 获取网络信息和音频解码信息。

voe\_network.h:额外协议的支持，Packet timeout notification，Dead-or-Alive connection observations。

voe\_rtp\_rtcp.h: 

2）头文件中对外主要接口:

(1)VoiceEngine：负责引擎的所有接口查询，存储共享数据信息ShareData。

(2)VoEBase：负责音频处理的基本操作。

(3)VoEAudioProcessing：音频信号处理接口，设置各个音频处理项的参数。

(4)VoECodec：音频编解码接口，提供支持的编解码器查询，音频编解码设置。

(5)VoEHardware：音频硬件设备接口，负责音频硬件设备的设置。

除了以上接口，还有VoENetEqStats, VoENetwork, VoERTP\_RTCP, VoEVideoSync, VoEVolumeControl, VoEFile, VoECallReport, VoEDtmf, VoEMeidaProcess 和VoEEncryption接口。

WebRTC使用继承实现接口转换和查询，接口之间的数据共享是通过ShareData完成，首先VoiceEngineImpl继承各个对外接口的实 现，所以可以从VoiceEngineImpl很容易获取其他对外接口。而VoiceEngineImpl本身也继承ShareData，当从 VoiceEngineImpl获取其他对外接口的同时，隐式的传递了ShareData指针，因此各个接口可以很方便的获取到ShareData的数据信息。

2.声音demo流程

以一个Channel为例，应用程序中将包含三个活动线程，录音线程，音频接收线程和播放线程。

录音流程：AudioDeviceWinCore负责采集音频数据，传递到AudioDeviceBuffer中缓 存，AudioDeviceBuffer则将数据送入TransmixMixer，首先交给AudioProcess进行近端音频处理，完成后分发到各个 Channel中，Channel则通过AudioCodingModule进行编码，编码后再交付到RtpRtcp中经由RTPSender发送出去。

接收流程：RTPReceiver负责接收音频RTP包，接收到RTP包后交给Channel，Channel转交给AudioCodingModule中的ACMNetEQ模块，进行解码缓存。

播放流程：Channel从ACMNetEQ模块中取出缓存的解码音频数据，如果需要进行远端数据处理的话，传递给AudioProcess处理。最后 所有Channel都汇入到OutputMixer中进行混音，混音后再传递到AudioProcess进行远端数据处理。最后送入 AudioDeviceModule中的AudioDevceWinCore播放。

3.voice\_engine需要的模块

主要由五大模块组成：AudioDeviceModule音频设备模块，AudioProcess音频处理模块，AudioCodingModule音频编码模块，AudioConferenceMixer混音模块和RtpRtcp传输模块。ShareData用于粘合各个模块之间的关系，负责管理全局的对象，包括AudioDeviceModule，TransmitMixer，OutputMixer，ChannelManager和AudioProcess。

以voice\_engine编解码模块介绍

neteq

VoEcodecImpl

channel

AudioCodingImpl

AudioEncoder

iLBC编码器

G711编码器

G722编码器

AudioDecoder

对声音的处理

--蓝色是继承关系

--红色是调用关系

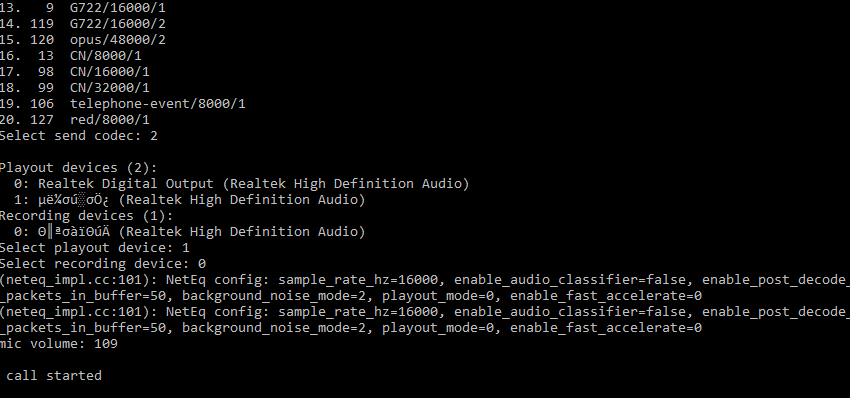
(1)G722位于webrtc/modules/g722

(2)AudioEncoder位于webrtc/modules/audio\_encoder\_interface

(3)Channel和VoEcodecImpl位于voice\_engine

(4)Neteq位于webrtc/modules/audio\_coding/neteq，用于解码。解码端的其他一些内容，诸如抖动缓冲区等，都在这个目录下。解码端的另一项重要功能，即混音功能，则在webrtc\modules\audio\_conference\_mixer目录下。

4.测试



本地测试，介入话筒和耳机，说话，可以从耳机听见从话筒传来的声音。

测试代码步骤说明：

1).创建VoiceEngine实例，并获得相关接口。如VoEBase、VoECodec、VoEVolumeControl等。

2).创建channel。

3).设置编解码。

4).设置播放、记录设备。

5).设置音频处理。如vad、ns等。

6).开始接受、播放、发送、记录(发送和接受的声音数据到本地文件)。