**C++实现RTMP协议发送H.264编码及AAC编码的音视频**

　　RTMP（Real Time Messaging Protocol）是专门用来传输音视频数据的流媒体协议，最初由Macromedia 公司创建，后来归Adobe公司所有，是一种私有协议，主要用来联系Flash Player和RtmpServer，如**FMS**, **Red5**, **crtmpserver**等。RTMP协议可用于实现直播、点播应用，通过**FMLE（Flash Media Live Encoder）**推送音视频数据至RtmpServer，可实现摄像头实时直播。不过，毕竟FMLE应用范围有限，想要把它嵌入到自己的程序中，还是要自己来实现RTMP协议的推送。本人实现了一个**RTMPLiveEncoder**，通过采集摄像头视频和麦克风音频，并进行H.264和AAC编码，然后发送到FMS和crtmpserver上，实现实时直播，可以通过flash player正常观看，目前效果良好，延迟时间在**2秒**左右。本文就介绍一下RTMPLiveEncoder的主要思路和关键点，以期对需要这方面技术的朋友有所帮助。

**技术分析**

　　要实现**RTMPLiveEncoder**，需要以下四种关键技术：

* 采集摄像头视频和麦克风音频
* H264编码和AAC编码
* 视频和音频数据封装为可被流媒体服务器识别的可播放流
* RTMP协议实现报文发送

　　其中，前两项技术在我之前的文章“[采集音频和摄像头视频并实时H264编码和AAC编码](http://www.cnblogs.com/haibindev/archive/2011/11/10/2244442.html)”中已经介绍过了，这里就不再啰嗦了。

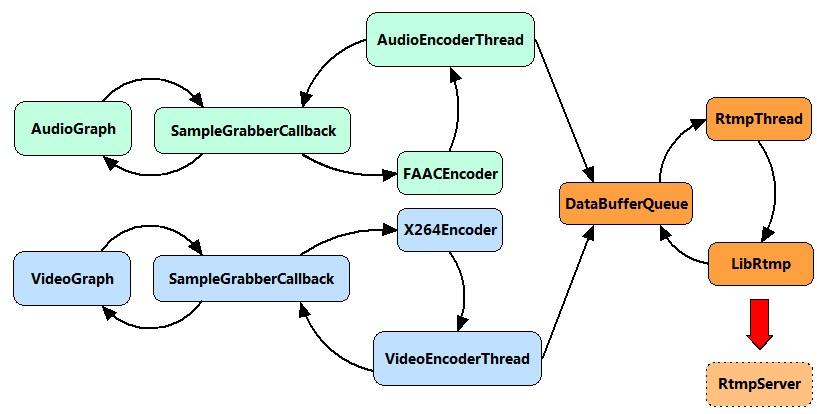
　　把音视频数据封装为可播放流，这个是一个难点。仔细研究一下，你会发现，RTMP Packet中封装的音视频数据流，其实和**FLV**封装音频和视频数据的方式是相同的，所以，我们只需要按照FLV封装H264和AAC的方式，即可生成可播放流。

　　我们再看一下RTMP协议。Adobe曾经发布过一份文档《RTMP Specification》，不过[wikipedia](http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol)指出这份文档隐藏了很多细节，单独根据它是无法正确实现RTMP的。不过，它还是有参考意义的。其实Adobe发布之前，RTMP协议就已经被破解的差不多了，现在也已经有比较完善的实现，比如：[RTMPDump](http://rtmpdump.mplayerhq.hu/)，它提供的是C语言的接口，这意味着可以很方便的在其他语言中调用。

**程序框架**

　　与我之前写的“[采集音频和摄像头视频并实时H264编码和AAC编码](http://www.cnblogs.com/haibindev/archive/2011/11/10/2244442.html)”这篇文章相同，采用DirectShow技术来实现音视频采集，音频编码和视频编码，在各自线程（AudioEncoderThread和VideoEncoderThread）中循环进行，RTMP的推送另起一个线程（RtmpThread）。两个编码线程实时编码音视频数据后，将数据交与Rtmp线程，由Rtmp线程循环封装为Rtmp Packet，然后发出去。

　　线程之间的数据交换，通过一个队列DataBufferQueue来实现。AudioEncoderThread和VideoEncoderThread把数据指针post到DataBufferQueue之后，立即返回，这样就可以避免因为发送Rtmp报文的而影响到编码线程的正常执行时间。



　　RtmpThread的主要工作就是发送音频数据流的解码信息头和视频数据流的解码信息头，并不断从DataBufferQueue中取出数据，封装为RTMP Packet，发送出去。流程如下列代码所示：（process\_buf\_queue\_，即是上图中的DataBufferQueue）



**librtmp**

**一、编译librtmp**

　　下载rtmpdump的代码，你会发现，它是一个地道的linux项目，除了一个简单的Makefile，其他什么都没有。好像librtmp不依赖于系统，我们可以不用费太多功夫，把它在windows上编译。不过，librtmp依赖于openssl和zlib，我们需要首先编译好它们。

**1. 编译openssl1.0.0e**

　　a) 下载并安装ActivePerl

　　b) 下载并安装nasm(http://nasm.sourceforge.net/)

　　c) 解压openssl压缩包

　　d) 运行cmd命令行，切到openssl目录，分别执行以下命令

>perl Configure VC-WIN32 --prefix=c:\some\dir  
>ms\do\_nasm

　　e) 运行Visual Studio Command Prompt(2010)，切到openssl目录，分别执行以下命令。

>nmake -f ms\nt.mak  
>nmake -f ms\nt.mak install

　　f) 编译完毕后，即可在第一个命令所指定的目录下发现编译好的sdk。

**2. 编译zlib**

　　a) 解压zlib压缩包

　　b) 运行Visual Studio Command Prompt(2010)，切到openssl目录，分别执行以下命令

>cd contrib\masmx86  
>bld\_ml32.bat

　　c) 回到zlib目录，进入contrib\vstudio\vc10目录，打开vs2010解决方案文件，

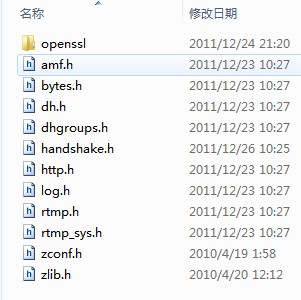
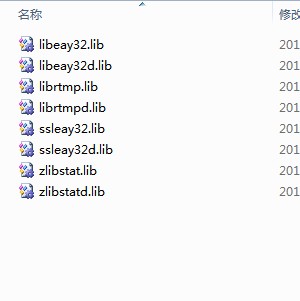
　　   在zlibstat工程属性中，去掉预编译宏 ZLIB\_WINAPI

　　d) 选择debug或release编译即可

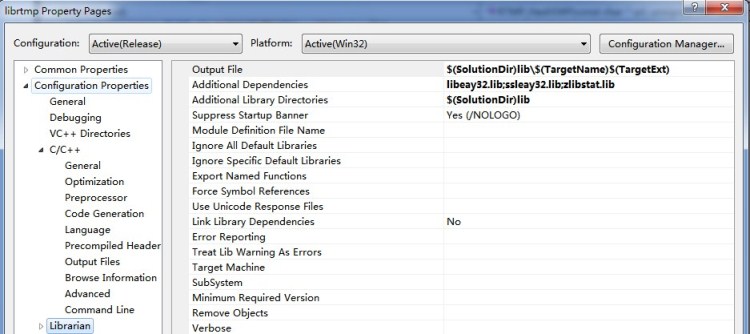
**3. 编译librtmp**

　　a) 首先打开visual studio 2010，新建一个win32 console工程，指定为静态链接库

　　b) 将librtmp的代码导入工程，把openssl、zlib的头文件和librtmp放在一起，把编译好的openssl和zlib的静态库放在一起

　　c) 在工程设置中，添加之前编译好的openssl和zlib的库，编译即可。

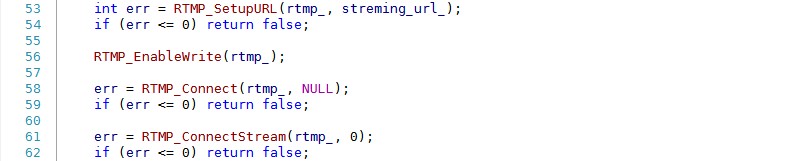


**二、librtmp的使用**

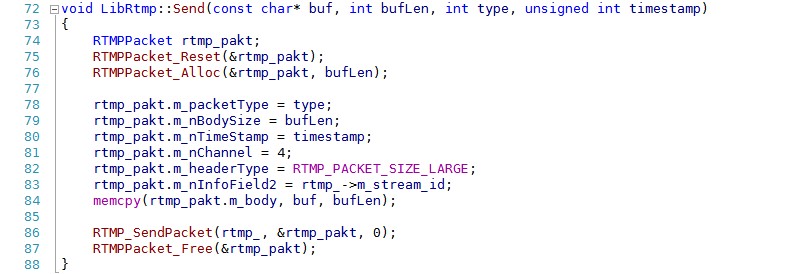
　　首先初始化RTMP结构

https://pic002.cnblogs.com/images/2011/254714/2011122911025738.jpg

　　开始之后，就要向RTMP Server发起握手连接报文



　　连接成功，就可以开始循环发送报文了，这里需要指定时戳和数据类型（Audio、Video、Metadata）。这里有一点需要注意的是，在调用Send之前，buf中的数据，必须是已经封装好的H264或AAC数据流。



　　关闭

https://pic002.cnblogs.com/images/2011/254714/2011122911061415.jpg

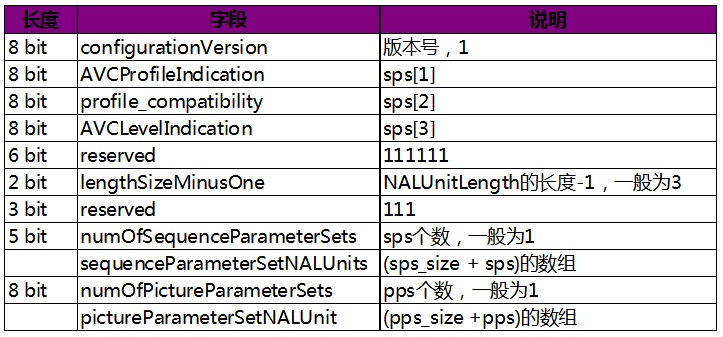
　　最后是释放

https://pic002.cnblogs.com/images/2011/254714/2011122911062315.jpg

**H264和AAC数据流**

　　本文提到过，RTMP推送的音视频流的封装形式和FLV格式相似，由此可知，向FMS推送H264和AAC直播流，需要首先发送"**AVC sequence header**"和"**AAC sequence header**"，这两项数据包含的是重要的编码信息，没有它们，解码器将无法解码。

　　AVC sequence header就是**AVCDecoderConfigurationRecord**结构，该结构在标准文档“**ISO-14496-15 AVC file format**”中有详细说明。



　　AAC sequence header存放的是**AudioSpecificConfig**结构，该结构则在“**ISO-14496-3 Audio**”中描述。AudioSpecificConfig结构的描述非常复杂，这里我做一下简化，事先设定要将要编码的音频格式，其中，选择"AAC-LC"为音频编码，音频采样率为44100，于是AudioSpecificConfig简化为下表：



　　这样，AVC sequence header和AAC sequence header的内容可以基本确定了，更详细的信息，大家可以去翻阅相关文档。

**运行效果**

　　RtmpLiveEncoder开始运行



　　用FMS自带的一个flash播放器播放

