

原 最简单的基于FFMPEG的音频编码器（PCM编码为AAC）

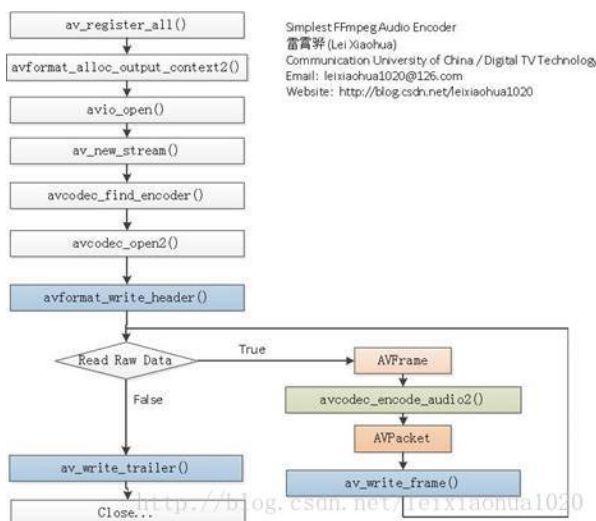
2014年05月11日 00:14:28 阅读数：67244

本文介绍一个最简单的基于FFMPEG的音频编码器。该编码器实现了PCM音频采样数据编码为AAC的压缩编码数据。编码器代码十分简单，但是每一行代码都很重要。通过看本编码器的源代码，可以了解FFMPEG音频编码的流程。

本程序使用最新版的类库（编译时间为2014.5.6），开发平台为VC2010。所有的配置都已经做好，只需要运行就可以了。

流程（2014.9.29更新）

下面附一张使用FFmpeg编码音频的流程图。使用该流程，不仅可以编码AAC的音频，而且可以编码MP3，MP2等等各种FFmpeg支持的音频。图中蓝色背景的函数是实际输出数据的函数。浅绿色的函数是音频编码的函数。



简单介绍一下流程中各个函数的意义：

av_register_all()：注册FFmpeg所有编解码器。

avformat_alloc_output_context2()：初始化输出码流的AVFormatContext。

avio_open()：打开输出文件。

av_new_stream()：创建输出码流的AVStream。

avcodec_find_encoder()：查找编码器。

avcodec_open2()：打开编码器。

avformat_write_header()：写文件头（对于某些没有文件头的封装格式，不需要此函数。比如说MPEG2TS）。

avcodec_encode_audio2()：编码音频。即将AVFrame（存储PCM采样数据）编码为AVPacket（存储AAC，MP3等格式的码流数据）。

av_write_frame()：将编码后的视频码流写入文件。

av_write_trailer()：写文件尾（对于某些没有文件头的封装格式，不需要此函数。比如说MPEG2TS）。

代码

```
[cpp]
1.  /**
2.   *最简单的基于FFmpeg的音频编码器
3.   *Simplest FFmpeg Audio Encoder
4.   *
5.   *雷霄骅 Lei Xiaohua
6.   *leixiaohua1020@126.com
7.   *中国传媒大学/数字电视技术
8.   *Communication University of China / Digital TV Technology
9.   *http://blog.csdn.net/leixiaohua1020
10.  *
11.  *本程序实现了音频PCM采样数据编码为压缩码流（MP3，WMA，AAC等）。
12.  *是最简单的FFmpeg音频编码方面的教程。
13.  *通过学习本例子可以了解FFmpeg的编码流程。
14.  *This software encode PCM data to AAC bitstream
```

```

24.  *This software encode PCM data to AAC bitstream.
15.  *It's the simplest audio encoding software based on FFmpeg.
16.  *Suitable for beginner of FFmpeg
17.  */
18.
19.  #include <stdio.h>
20.
21.  #define __STDC_CONSTANT_MACROS
22.
23.  #ifdef _WIN32
24.  //Windows
25.  extern "C"
26.  {
27.  #include "libavcodec/avcodec.h"
28.  #include "libavformat/avformat.h"
29.  };
30.  #else
31.  //Linux...
32.  #ifdef __cplusplus
33.  extern "C"
34.  {
35.  #endif
36.  #include <libavcodec/avcodec.h>
37.  #include <libavformat/avformat.h>
38.  #ifdef __cplusplus
39.  };
40.  #endif
41.  #endif
42.
43.
44.  int flush_encoder(AVFormatContext *fmt_ctx,unsigned int stream_index){
45.  int ret;
46.  int got_frame;
47.  AVPacket enc_pkt;
48.  if (!(fmt_ctx->streams[stream_index]->codec->capabilities &
49.  CODEC_CAP_DELAY))
50.  return 0;
51.  while (1) {
52.  enc_pkt.data = NULL;
53.  enc_pkt.size = 0;
54.  av_init_packet(&enc_pkt);
55.  ret = avcodec_encode_audio2 (fmt_ctx->streams[stream_index]->codec, &enc_pkt,
56.  NULL, &got_frame);
57.  av_frame_free(NULL);
58.  if (ret < 0)
59.  break;
60.  if (!got_frame){
61.  ret=0;
62.  break;
63.  }
64.  printf("Flush Encoder: Succeed to encode 1 frame!\tsize:%5d\n",enc_pkt.size);
65.  /* mux encoded frame */
66.  ret = av_write_frame(fmt_ctx, &enc_pkt);
67.  if (ret < 0)
68.  break;
69.  }
70.  return ret;
71.  }
72.
73.  int main(int argc, char* argv[])
74.  {
75.  AVFormatContext* pFormatCtx;
76.  AVOutputFormat* fmt;
77.  AVStream* audio_st;
78.  AVCodecContext* pCodecCtx;
79.  AVCodec* pCodec;
80.
81.  uint8_t* frame_buf;
82.  AVFrame* pFrame;
83.  AVPacket pkt;
84.
85.  int got_frame=0;
86.  int ret=0;
87.  int size=0;
88.
89.  FILE *in_file=NULL; //Raw PCM data
90.  int framenum=1000; //Audio frame number
91.  const char* out_file = "tdjm.aac"; //Output URL
92.  int i;
93.
94.  in_file= fopen("tdjm.pcm", "rb");
95.
96.  av_register_all();
97.
98.  //Method 1.
99.  pFormatCtx = avformat_alloc_context();
100.  fmt = av_guess_format(NULL, out_file, NULL);
101.  pFormatCtx->oformat = fmt;
102.
103.
104.  //Method 2.
105.  //avformat_alloc_outout context2(&pFormatCtx, NULL, NULL, out file):

```

```

106. //fmt = pFormatCtx->oformat;
107.
108. //Open output URL
109. if (avio_open(&pFormatCtx->pb,out_file, AVIO_FLAG_READ_WRITE) < 0){
110.     printf("Failed to open output file!\n");
111.     return -1;
112. }
113.
114. audio_st = avformat_new_stream(pFormatCtx, 0);
115. if (audio_st==NULL){
116.     return -1;
117. }
118. pCodecCtx = audio_st->codec;
119. pCodecCtx->codec_id = fmt->audio_codec;
120. pCodecCtx->codec_type = AVMEDIA_TYPE_AUDIO;
121. pCodecCtx->sample_fmt = AV_SAMPLE_FMT_S16;
122. pCodecCtx->sample_rate= 44100;
123. pCodecCtx->channel_layout=AV_CH_LAYOUT_STEREO;
124. pCodecCtx->channels = av_get_channel_layout_nb_channels(pCodecCtx->channel_layout);
125. pCodecCtx->bit_rate = 64000;
126.
127. //Show some information
128. av_dump_format(pFormatCtx, 0, out_file, 1);
129.
130. pCodec = avcodec_find_encoder(pCodecCtx->codec_id);
131. if (!pCodec){
132.     printf("Can not find encoder!\n");
133.     return -1;
134. }
135. if (avcodec_open2(pCodecCtx, pCodec,NULL) < 0){
136.     printf("Failed to open encoder!\n");
137.     return -1;
138. }
139. pFrame = av_frame_alloc();
140. pFrame->nb_samples= pCodecCtx->frame_size;
141. pFrame->format= pCodecCtx->sample_fmt;
142.
143. size = av_samples_get_buffer_size(NULL, pCodecCtx->channels,pCodecCtx->frame_size,pCodecCtx->sample_fmt, 1);
144. frame_buf = (uint8_t *)av_malloc(size);
145. avcodec_fill_audio_frame(pFrame, pCodecCtx->channels, pCodecCtx->sample_fmt,(const uint8_t*)frame_buf, size, 1);
146.
147. //Write Header
148. avformat_write_header(pFormatCtx,NULL);
149.
150. av_new_packet(&pkt,size);
151.
152. for (i=0; i<framenum; i++){
153.     //Read PCM
154.     if (fread(frame_buf, 1, size, in_file) <= 0){
155.         printf("Failed to read raw data! \n");
156.         return -1;
157.     }else if(feof(in_file)){
158.         break;
159.     }
160.     pFrame->data[0] = frame_buf; //PCM Data
161.
162.     pFrame->pts=i*100;
163.     got_frame=0;
164.     //Encode
165.     ret = avcodec_encode_audio2(pCodecCtx, &pkt,pFrame, &got_frame);
166.     if(ret < 0){
167.         printf("Failed to encode!\n");
168.         return -1;
169.     }
170.     if (got_frame==1){
171.         printf("Succeed to encode 1 frame! \tsize:%5d\n",pkt.size);
172.         pkt.stream_index = audio_st->index;
173.         ret = av_write_frame(pFormatCtx, &pkt);
174.         av_free_packet(&pkt);
175.     }
176. }
177.
178. //Flush Encoder
179. ret = flush_encoder(pFormatCtx,0);
180. if (ret < 0) {
181.     printf("Flushing encoder failed\n");
182.     return -1;
183. }
184.
185. //Write Trailer
186. av_write_trailer(pFormatCtx);
187.
188. //Clean
189. if (audio_st){
190.     avcodec_close(audio_st->codec);
191.     av_free(pFrame);
192.     av_free(frame_buf);
193. }
194. avio_close(pFormatCtx->pb);
195. avformat_free_context(pFormatCtx);
196.

```

```
197.         fclose(in_file);
198.
199.         return 0;
200.     }
```

结果

程序运行完成后，会将一个PCM采样数据文件 (*.pcm) 编码为AAC码流文件 (*.aac)。

下载

simplest ffmpeg audio encoder

项目主页

SourceForge：<https://sourceforge.net/projects/simplestffmpegaudioencoder/>

Github：https://github.com/leixiaohua1020/simplest_ffmpeg_audio_encoder

开源中国：http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_audio_encoder

CSDN工程下载地址：

<http://download.csdn.net/detail/leixiaohua1020/7324091>

PUDN工程下载地址：

<http://www.pudn.com/downloads644/sourcecode/multimedia/detail2605236.html>

更新-1.1 (2015.2.13)=

这次考虑到了跨平台的要求，调整了源代码。经过这次调整之后，源代码可以在以下平台编译通过：

VC++：打开sln文件即可编译，无需配置。

cl.exe：打开compile_cl.bat即可命令行下使用cl.exe进行编译，注意可能需要按照VC的安装路径调整脚本里面的参数。编译命令如下。

```
[plain]
1.  ::VS2010 Environment
2.  call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
3.  ::include
4.  @set INCLUDE=include;%INCLUDE%
5.  ::lib
6.  @set LIB=lib;%LIB%
7.  ::compile and link
8.  cl simplest_ffmpeg_audio_encoder.cpp /link avcodec.lib avformat.lib avutil.lib ^
9.  avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib /OPT:NOREF
```

MinGW：MinGW命令行下运行compile_mingw.sh即可使用MinGW的g++进行编译。编译命令如下。

```
[cpp]
1.  g++ simplest_ffmpeg_audio_encoder.cpp -g -o simplest_ffmpeg_audio_encoder.exe \
2.  -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

GCC：Linux或者MacOS命令行下运行compile_gcc.sh即可使用GCC进行编译。编译命令如下。

```
[cpp]
1.  gcc simplest_ffmpeg_audio_encoder.cpp -g -o simplest_ffmpeg_audio_encoder.out \
2.  -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

PS：相关的编译命令已经保存到了工程文件夹中

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8445209>

SourceForge上已经更新。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/25430449>

文章标签：[ffmpeg](#) [pcm](#) [aac](#) [编码](#) [音频](#)

个人分类：[FFMPEG](#) [我的开源项目](#)

所属专栏：[FFmpeg](#)

此PDF由[spygg](#)生成,请尊重原作者版权!!!
我的邮箱:liushidc@163.com