

视音频数据处理入门：AAC音频码流解析

2016年01月31日 11:55:51 阅读数：56890

视音频数据处理入门系列文章：

[视音频数据处理入门：RGB、YUV像素数据处理](#)

[视音频数据处理入门：PCM音频采样数据处理](#)

[视音频数据处理入门：H.264视频码流解析](#)

[视音频数据处理入门：AAC音频码流解析](#)

[视音频数据处理入门：FLV封装格式解析](#)

[视音频数据处理入门：UDP-RTP协议解析](#)

本文继续上一篇文章的内容，介绍一个音频码流处理程序。音频码流在视频播放器中的位置如下所示。



本文中的程序是一个AAC码流解析程序。该程序可以从AAC码流中分析得到它的基本单元ADTS frame,并且可以简单解析ADTS frame首部的字段。通过修改该程序可以实现不同的AAC码流处理功能。

原理

AAC原始码流（又称为“裸流”）是由一个一个的ADTS frame组成的。他们的结构如下图所示。



其中每个ADTS frame之间通过syncword（同步字）进行分隔。同步字为0xFFF（二进制“11111111111”）。AAC码流解析的步骤就是首先从码流中搜索0xFFF，分离出ADTS frame；然后再分析ADTS frame的首部各个字段。本文的程序即实现了上述的两个步骤。

代码

整个程序位于simplest_aac_parser()函数中，如下所示。

```
[cpp]
1.  /**
2.   * 最简单的视音频数据处理示例
3.   * Simplest MediaData Test
4.   *
5.   * 雷霄骅 Lei Xiaohua
6.   * leixiaohua102@gmail.com
```

```

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. * 本项目包含如下几种视音频测试示例：
12. * (1) 像素数据处理程序。包含RGB和YUV像素格式处理的函数。
13. * (2) 音频采样数据处理程序。包含PCM音频采样格式处理的函数。
14. * (3) H.264码流分析程序。可以分离并解析NALU。
15. * (4) AAC码流分析程序。可以分离并解析ADTS帧。
16. * (5) FLV封装格式分析程序。可以将FLV中的MP3音频码流分离出来。
17. * (6) UDP-RTP协议分析程序。可以将分析UDP/RTP/MPEG-TS数据包。
18. *
19. * This project contains following samples to handling multimedia data:
20. * (1) Video pixel data handling program. It contains several examples to handle RGB and YUV data.
21. * (2) Audio sample data handling program. It contains several examples to handle PCM data.
22. * (3) H.264 stream analysis program. It can parse H.264 bitstream and analysis NALU of stream.
23. * (4) AAC stream analysis program. It can parse AAC bitstream and analysis ADTS frame of stream.
24. * (5) FLV format analysis program. It can analysis FLV file and extract MP3 audio stream.
25. * (6) UDP-RTP protocol analysis program. It can analysis UDP/RTP/MPEG-TS Packet.
26. *
27. */
28. #include <stdio.h>
29. #include <stdlib.h>
30. #include <string.h>
31.
32.
33. int getADTSframe(unsigned char* buffer, int buf_size, unsigned char* data ,int* data_size){
34.     int size = 0;
35.
36.     if(!buffer || !data || !data_size ){
37.         return -1;
38.     }
39.
40.     while(1){
41.         if(buf_size < 7 ){
42.             return -1;
43.         }
44.         //Sync words
45.         if((buffer[0] == 0xff) && ((buffer[1] & 0xf0) == 0xf0) ){
46.             size |= ((buffer[3] & 0x03) <<11); //high 2 bit
47.             size |= buffer[4]<<3; //middle 8 bit
48.             size |= ((buffer[5] & 0xe0)>>5); //low 3bit
49.             break;
50.         }
51.         --buf_size;
52.         ++buffer;
53.     }
54.
55.     if(buf_size < size){
56.         return 1;
57.     }
58.
59.     memcpy(data, buffer, size);
60.     *data_size = size;
61.
62.     return 0;
63. }
64.
65. int simplest_aac_parser(char *url)
66. {
67.     int data_size = 0;
68.     int size = 0;
69.     int cnt=0;
70.     int offset=0;
71.
72.     //FILE *myout=fopen("output_log.txt","wb+");
73.     FILE *myout=stdout;
74.
75.     unsigned char *aacframe=(unsigned char *)malloc(1024*5);
76.     unsigned char *aacbuffer=(unsigned char *)malloc(1024*1024);
77.
78.     FILE *ifile = fopen(url, "rb");
79.     if(!ifile){
80.         printf("Open file error");
81.         return -1;
82.     }
83.
84.     printf("----- ADTS Frame Table -----+\n");
85.     printf(" NUM | Profile | Frequency| Size |\n");
86.     printf("-----+-----+-----+\n");
87.
88.     while(!feof(ifile)){
89.         data_size = fread(aacbuffer+offset, 1, 1024*1024-offset, ifile);
90.         unsigned char* input_data = aacbuffer;
91.
92.         while(1)
93.         {
94.             int ret=getADTSframe(input_data, data_size, aacframe, &size);
95.             if(ret==-1){
96.                 break;
97.             }else if(ret==1){

```

```

98.         memcpy(aacbuffer,input_data,data_size);
99.         offset=data_size;
100.        break;
101.    }
102.
103.    char profile_str[10]={0};
104.    char frequency_str[10]={0};
105.
106.    unsigned char profile=aacframe[2]&0xC0;
107.    profile=profile>>6;
108.    switch(profile){
109.    case 0: sprintf(profile_str,"Main");break;
110.    case 1: sprintf(profile_str,"LC");break;
111.    case 2: sprintf(profile_str,"SSR");break;
112.    default:sprintf(profile_str,"unknown");break;
113.    }
114.
115.    unsigned char sampling_frequency_index=aacframe[2]&0x3C;
116.    sampling_frequency_index=sampling_frequency_index>>2;
117.    switch(sampling_frequency_index){
118.    case 0: sprintf(frequency_str,"96000Hz");break;
119.    case 1: sprintf(frequency_str,"88200Hz");break;
120.    case 2: sprintf(frequency_str,"64000Hz");break;
121.    case 3: sprintf(frequency_str,"48000Hz");break;
122.    case 4: sprintf(frequency_str,"44100Hz");break;
123.    case 5: sprintf(frequency_str,"32000Hz");break;
124.    case 6: sprintf(frequency_str,"24000Hz");break;
125.    case 7: sprintf(frequency_str,"22050Hz");break;
126.    case 8: sprintf(frequency_str,"16000Hz");break;
127.    case 9: sprintf(frequency_str,"12000Hz");break;
128.    case 10: sprintf(frequency_str,"11025Hz");break;
129.    case 11: sprintf(frequency_str,"8000Hz");break;
130.    default:sprintf(frequency_str,"unknown");break;
131.    }
132.
133.
134.    fprintf(myout,"%5d| %8s|  %8s| %5d|\n",cnt,profile_str ,frequency_str,size);
135.    data_size -= size;
136.    input_data += size;
137.    cnt++;
138.    }
139.
140.    }
141.    fclose(ifile);
142.    free(aacbuffer);
143.    free(aacframe);
144.
145.    return 0;
146.    }

```

上文中的函数调用方法如下所示。

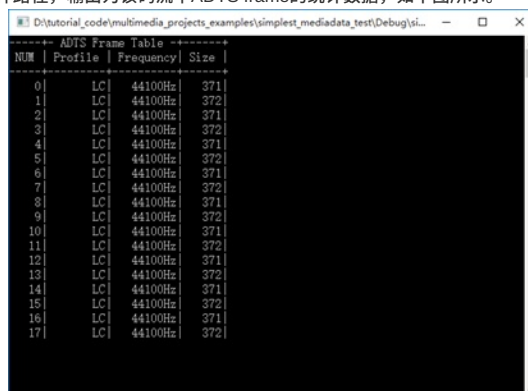
```

1.  simplest_aac_parser("nocturne.aac");

```

结果

本程序的输入为一个AAC原始码流（裸流）的文件路径，输出为该码流中ADTS frame的统计数据，如下图所示。



Num	Profile	Frequency	Size
0	LC	44100Hz	371
1	LC	44100Hz	372
2	LC	44100Hz	371
3	LC	44100Hz	372
4	LC	44100Hz	371
5	LC	44100Hz	372
6	LC	44100Hz	371
7	LC	44100Hz	372
8	LC	44100Hz	371
9	LC	44100Hz	372
10	LC	44100Hz	371
11	LC	44100Hz	372
12	LC	44100Hz	371
13	LC	44100Hz	372
14	LC	44100Hz	371
15	LC	44100Hz	372
16	LC	44100Hz	371
17	LC	44100Hz	372

下载

Simplest mediadata test

项目主页

SourceForge：<https://sourceforge.net/projects/simplest-mediadata-test/>

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

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

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/9422409>

本项目包含如下几种视音频数据解析示例：

- (1)像素数据处理程序。包含RGB和YUV像素格式处理的函数。
- (2)音频采样数据处理程序。包含PCM音频采样格式处理的函数。
- (3)H.264码流分析程序。可以分离并解析NALU。
- (4)AAC码流分析程序。可以分离并解析ADTS帧。
- (5)FLV封装格式分析程序。可以将FLV中的MP3音频码流分离出来。
- (6)UDP-RTP协议分析程序。可以将分析UDP/RTP/MPEG-TS数据包。

雷霄骅 (Lei Xiaohua)

leixiaohua1020@126.com

<http://blog.csdn.net/leixiaohua1020>

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/50535042>

文章标签：[AAC](#) [音频](#) [码流分析](#)

个人分类：[我的开源项目](#)

此PDF由[spygg](#)生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com