**视频转码API调用说明**

1. **API数据结构说明**

1.1 视频截图参数数据结构说明(**ExtractImage\_info**)

用途:

设定视频截图的参数。

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| xres | ushort | 设定图片分辩率(水平)  =0，和视频分辩率相同 |
| yres | ushort | 设定图片分辩率(垂直)  =0 和视频分辩率相同 |
| duration\_second | uint | 切分文件的时长设定(以秒为单位) |
| filePathIn | byte[] | 输入文件的路径 |
| fileNameIn | byte[] | 输入文件名 |
| filePathOut | byte[] | 输出文件的路径 |
| fileNameOut | byte[] | 输出文件名 |

1.2 视频转码中视频文件的参数说明**(Vedio\_format**)

用途:

设定转码视频的格式。

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| x\_res | ushort | 视频分辨率(水平方向) |
| y\_res | ushort | 视频分辨率(垂直方向) |
| codec | ushort | 视频codec，设定  Codec=1,即h264 |
| television | ushort | 制式,设定  television=1, 即pal |
| frame\_rate | ushort | 视频帧率，  frame\_rate=24 |
| duration\_sec | ushort | 视频切分文件的时长定义(秒为单位)，这个参数通过配置设定，默认40\*60 秒 |
| hw\_mode | ushort | 硬件加速  = 0,不使用;  =1,intel hd graphics  =2,nvidia gpu |

1.3 视频转码中音频文件的参数说明(**Audio\_format**)

用途:

设定音频文件的格式

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| sample\_rate | uint | 音频采样率(单位hz)  = 44100 |
| code\_rate | uint | 音频的码率(bps)  = 128000 |
| codec | ushort | 音频codec  =0 mp3  =1 aac |
| duration\_sec | ushort | 音频切分文件的时长定义(秒为单位)  默认=40 min |

1.4 转码的音视频 参数定义(**Vedio\_info**)

用途:

转码时的参数定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| fileType | ushort | 文件类型  =0,audio  =1,vedio |
| thread\_num | ushort | 线程数 默认4 |
| audio\_fmt | Audio\_format | 定义见上面  视频文件，需要定义音频格式 |
| vedio\_fmt | Vedio\_format | 定义见上面  音频文件，不需要定义这个结构 |
| filePathIn | byte[] | 输入文件路径 |
| fileNameIn | byte[] | 输入文件名 |
| filePathOut | byte[] | 输出文件的路径 |
| fileNameOut | byte[] | 输出文件名，需要带扩展名 |
|  |  |  |

1.5 回调函数结构的定义（VedioCallback\_info)

|  |  |  |
| --- | --- | --- |
| 字段名 | 数据类型 | 说明 |
| file\_index | ushort | 当前转码的文件下标,保留字段，当前无意义 |
| status | ushort | 当前转码的状态  =1>初始化状态  =2 >转码  =3 >转码后生成新文件  =4>转码完成  =5 >转码终止 |
| cur\_duration | uint | 已转码的时长(单位秒) |
| all\_duration | uint | 总时长(单位秒) |

1. **api函数调用说明**

2.1 **视频截图**

作用:

截取视频的第一帧为jpeg图像。如果视频被转码为分段视频，则

每段视频截一个图。

说明:

函数不支持多线程重入。

函数定义:

int ffmpeg\_extract\_image(IntPtr param);

参数定义：

param:是指向ExtractImage\_info结构

返回值:

<0 :失败

>0 :成功，并返回截图的图片个数.

生成的截图的文件命名:

如果是单个文件: 定义的输入文件名是test.jpeg

生成的截图文件名是test\_\_0.jpeg

如果生成多个截图文件: 定义的输入文件名是test.jpeg

生成的截图文件名是test\_\_0.jpeg

生成的截图文件名是test\_\_1.jpeg

生成的截图文件名是test\_\_2.jpeg

等。

**2.2 音视频转码**

作用:

将一种音视频格式转到另外一种。

说明:

只支持单线程，不支持在多个线程同时调用这个函数。

函数定义:

int ffmpeg\_convert\_start(IntPtr param);

参数定义：

param: 是指向Vedio\_info结构

返回值:

<0 :失败

>0 :成功，并返回成功转码的音视频文件的个数，

如果视频不分段，则返回1,分段则返回分段的个数。

输出文件名说明:

如果不分段，则直接是输出文件名

如果分段， 则在扩展名之前加上\_0,\_1,如test\_0.mp4,

Test\_1.mp4,test\_2.mp4.

转码时硬件加速的说明:

调用转码函数时，先指定intel hd graphics的硬件加速，如果

失败，再调用,nvidia gpu的硬件加速，如果失败，调用不带硬件

加速的参数设定。

**2.3 停止转码**

作用:

停止正在转码的操作，并返回。

函数定义:

int ffmpeg\_convert\_cancel();

返回值:

<0 :失败

=0：成功。

**2.4 回调函数定义:**

作用:

用于转码进度的显示，应用程序应该实现这个函数，函数中，

不能写耗时的操作，保存值后立即返回，否则，影响转码

函数定义:

public static int ffmpegCallBack(int nCurSecond, int nAllSecond, int nStatus)

参数定义:

nCurSecond: 当前已完成转码的时长(秒为单位)

nAllSecond: 总时长

nStatus : 当前转码的状态

=1: 初始化状态

=2: 转码

=3: 转码后生成新文件

=4 转码完成

=5 转码终止

**2.5 设定回调函数:**

作用：

设定回调函数的指针

void ffmpeg\_setCallback(ffmpegCallbackHander loopHander);

1. **C#示例**

**3.1 回调函数**

public static int ffmpegCallBack(int nCurSecond, int nAllSecond, int nStatus)

{

Console.WriteLine("callback：" + nCurSecond.ToString());

return 0;

}

**3.2 转码视频**

static void test\_vedio()

{

string strInputFilePath = "e:\\work\\test\\in";

string strInputFileName = "test1.flv";

string strOutputFilePath = "e:\\work\\test\\out";

string strOutputFileName = "test.mp4";

Vedio\_info vedioInfo;

Vedio\_format vedioFmt;

Audio\_format audioFmt;

vedioInfo.filePathIn = new byte[200];

byte[] filePathIn1 = Encoding.UTF8.GetBytes(strInputFilePath);

Array.Copy(filePathIn1, vedioInfo.filePathIn, filePathIn1.Length);

vedioInfo.fileNameIn = new byte[100];

byte[] fileNameIn1 = Encoding.UTF8.GetBytes(strInputFileName);

Array.Copy(fileNameIn1, vedioInfo.fileNameIn, fileNameIn1.Length);

vedioInfo.filePathOut = new byte[200];

byte[] filePathOut1 = Encoding.UTF8.GetBytes(strOutputFilePath);

Array.Copy(filePathOut1, vedioInfo.filePathOut, filePathOut1.Length);

vedioInfo.fileNameOut = new byte[100];

byte[] fileNameOut1 = Encoding.UTF8.GetBytes(strOutputFileName);

Array.Copy(fileNameOut1, vedioInfo.fileNameOut, fileNameOut1.Length);

vedioInfo.fileType = 1;

vedioFmt.codec = 1;

vedioFmt.duration\_sec = 40 \* 60;

vedioFmt.frame\_rate = 24;

vedioFmt.television = 1;

vedioFmt.x\_res = 1280;

vedioFmt.y\_res = 720;

vedioFmt.hw\_mode = 1;

vedioInfo.vedio\_fmt = vedioFmt;

//----

audioFmt.codec = 1;

audioFmt.code\_rate = 128\*1000;

audioFmt.duration\_sec = 40 \* 60;

audioFmt.sample\_rate = 44100;

vedioInfo.audio\_fmt = audioFmt;

vedioInfo.thread\_num = 4;

int sizeCur = Marshal.SizeOf(vedioInfo);

IntPtr pIntPtr = Marshal.AllocHGlobal(sizeCur);

Marshal.StructureToPtr(vedioInfo, pIntPtr, true);

ffmpegCallbackHander pCallback = new ffmpegCallbackHander(Program.ffmpegCallBack);

ffmpeg\_setCallback(pCallback);

ffmpeg\_convert\_start(pIntPtr);

}

**3.3转码音频**

static void test\_audio()

{

string strInputFilePath = "e:\\work\\test\\in"；

string strInputFileName = "test3.mp3";

string strOutputFilePath = "e:\\work\\test\\out";

string strOutputFileName = "test.m4a";

Vedio\_info vedioInfo;

Vedio\_format vedioFmt;

Audio\_format audioFmt;

vedioInfo.filePathIn = new byte[200];

byte[] filePathIn1 = Encoding.UTF8.GetBytes(strInputFilePath);

Array.Copy(filePathIn1, vedioInfo.filePathIn, filePathIn1.Length);

vedioInfo.fileNameIn = new byte[100];

byte[] fileNameIn1 = Encoding.UTF8.GetBytes(strInputFileName);

Array.Copy(fileNameIn1, vedioInfo.fileNameIn, fileNameIn1.Length);

vedioInfo.filePathOut = new byte[200];

byte[] filePathOut1 = Encoding.UTF8.GetBytes(strOutputFilePath);

Array.Copy(filePathOut1, vedioInfo.filePathOut, filePathOut1.Length);

vedioInfo.fileNameOut = new byte[100];

byte[] fileNameOut1 = Encoding.UTF8.GetBytes(strOutputFileName);

Array.Copy(fileNameOut1, vedioInfo.fileNameOut, fileNameOut1.Length);

vedioInfo.fileType = 0;

//----

audioFmt.codec = 1;

audioFmt.code\_rate = 128 \* 1000;

audioFmt.duration\_sec = 40 \* 60;

audioFmt.sample\_rate = 44100;

vedioInfo.audio\_fmt = audioFmt;

vedioInfo.thread\_num = 4;

int sizeCur = Marshal.SizeOf(vedioInfo);

IntPtr pIntPtr = Marshal.AllocHGlobal(sizeCur);

Marshal.StructureToPtr(vedioInfo, pIntPtr, true);

ffmpegCallbackHander pCallback = new ffmpegCallbackHander(Program.ffmpegCallBack);

ffmpeg\_setCallback(ffmpegCallBack);

ffmpeg\_convert\_start(pIntPtr);}

**3.4 截图**

static void test\_extract1()

{

string strInputFilePath = "e:\\work\\test\\in";

string strInputFileName = "test6.avi";

string strOutputFilePath = "e:\\work\\test\\out";

string strOutputFileName = "test.jpeg";

ExtractImage\_info vedioInfo;

vedioInfo.filePathIn = new byte[200];

byte[] filePathIn1 = Encoding.UTF8.GetBytes(strInputFilePath);

Array.Copy(filePathIn1, vedioInfo.filePathIn, filePathIn1.Length);

vedioInfo.fileNameIn = new byte[100];

byte[] fileNameIn1 = Encoding.UTF8.GetBytes(strInputFileName);

Array.Copy(fileNameIn1, vedioInfo.fileNameIn, fileNameIn1.Length);

vedioInfo.filePathOut = new byte[200];

byte[] filePathOut1 = Encoding.UTF8.GetBytes(strOutputFilePath);

Array.Copy(filePathOut1, vedioInfo.filePathOut, filePathOut1.Length);

vedioInfo.fileNameOut = new byte[100];

byte[] fileNameOut1 = Encoding.UTF8.GetBytes(strOutputFileName);

Array.Copy(fileNameOut1, vedioInfo.fileNameOut, fileNameOut1.Length);

vedioInfo.duration\_second = 10;

vedioInfo.xres = 800;

vedioInfo.yres = 600;

int sizeCur = Marshal.SizeOf(vedioInfo);

IntPtr pIntPtr = Marshal.AllocHGlobal(sizeCur);

Marshal.StructureToPtr(vedioInfo, pIntPtr, true);

ffmpeg\_extract\_image(pIntPtr);}