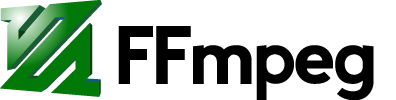
FFmpeg



内容目录

[1. ffmpeg简介 2](#__RefHeading___Toc449_891268285)

[2. ffmpeg相关资料 2](#__RefHeading___Toc451_891268285)

[3. ffmpeg项目组成 2](#__RefHeading___Toc453_891268285)

[3.1 FFmpeg的封装模块AVFormat 3](#__RefHeading___Toc455_891268285)

[3.2 FFmpeg的编解码板块AVCodec 3](#__RefHeading___Toc457_891268285)

[3.3 FFmpeg的滤镜模块AVFilter 3](#__RefHeading___Toc459_891268285)

[3.4 FFmpeg的视频图像转换计算模块swscale 3](#__RefHeading___Toc461_891268285)

[3.5 FFmpeg的音频转换计算模块swresample 3](#__RefHeading___Toc463_891268285)

[3.6 FFmpeg的编解码工具ffmpeg 3](#__RefHeading___Toc465_891268285)

[3.7 FFmpeg的播放器ffplay 5](#__RefHeading___Toc467_891268285)

[3.8 FFmpeg的多媒体分析器ffprobe 5](#__RefHeading___Toc469_891268285)

[4. ffmpeg RTMP 相关命令 6](#__RefHeading___Toc993_222442618)

# 1. ffmpeg简介

FFmpeg是一套可以用来记录、转换数字音频、视频，并能将其转化为流的开源计算机程序。采用LGPL或GPL许可证。它提供了录制、转换以及流化音视频的完整解决方案。

FFmpeg在Linux平台下开发，但它同样也可以在其它操作系统环境中编译运行，包括Windows、Mac OS X等。这个项目最早由Fabrice Bellard发起，2004年至2015年间由Michael Niedermayer主要负责维护。许多FFmpeg的开发人员都来自MPlayer项目，而且当前FFmpeg也是放在MPlayer项目组的服务器上。项目的名称来自MPEG视频编码标准。

FFMpeg提供了多种媒体格式的封装和解封装，包括多种音视频编码、多种协议的流媒体、多种色彩格式转换、多种采样率转换、多种码率转换等；FFmpeg框架提供了多种丰富的插件模块，包含封装与解封装的插件、编码与解码的插件等。

FFmpeg中的“FF”指的是“Fast Forward”，FFmpeg中的“mpeg”则是“Moving Picture Experts Group（动态图像专家组）”。

# 2. ffmpeg相关资料

官方文档资料

FFmpeg官方文档：https://ffmpeg.org/documentation.html

FFmpeg官方wiki：http://trac.ffmpeg.org/wiki

中文经典资料

雷霄骅博士csdn链接：https://blog.csdn.net/leixiaohua1020

罗索实验室：http://www.rosoo.net/

ChinaFFmpeg：http://bbs.chinaffmpeg.com

除了以上这些信息，还可以通过Google、百度等搜索引擎获得大量相关资料。

# 3. ffmpeg项目组成

FFmpeg框架的基本组成包括AVFormat、AVFilter、AVDevice、AVUtil等模块库。

下面针对这些模块做一个大概的介绍。

## 3.1 FFmpeg的封装模块AVFormat

AVFormat中实现了目前多媒体领域中的绝大多数媒体封装格式，包括封装和解封装，如MP4、FLV、KV、TS等文件封装格式，RTMP、RTSP、MMS、HLS等网络协议封装格式。

FFmpeg是否支持某种媒体封装格式，取决于编译时是否包含了该格式的封装库。根据实际需求，可进行媒体封装格式的拓展，增加自己定制的封装格式，即在AVFormat中增加自己的封装处理模块。

## 3.2 FFmpeg的编解码板块AVCodec

AVCodec中实现了目前多媒体领域绝大多数常用的编解码格式，既支持编码，也支持解码。AVCodec除了支持MPEG4、AAC、MJPEG等自带的媒体编解码格式之外，还支持第三方的编解码器，如H.264（AVC）编码，需要使用x264编码器；H.265（HEVC）编码，需要使用X265编码器；MP3（mp3lame）编码，需要使用libmp3lame编码器。如果希望增加自己的编码格式，或者硬件编解码，则需要在AVCodec中增加相应的编解码模块。

## 3.3 FFmpeg的滤镜模块AVFilter

AVFilter库提供了一个通用的音频、视频、字幕等滤镜处理框架。在AVFilter中，滤镜框架可以有多个输入或多个输出。

## 3.4 FFmpeg的视频图像转换计算模块swscale

swscale模块提供了高级别的图像转换API，例如它允许进行图像缩放和像素格式转换，常见于将图像从1080p转换成720p或者480p等的缩放，或者将图像数据从YUV420P转换成YUYV，或者YUV转RGB等图像格式转换。

## 3.5 FFmpeg的音频转换计算模块swresample

swresample模块提供了高级别的音频重采样API。例如它允许操作音频采样、音频通道布局转换与布局调整。

## 3.6 FFmpeg的编解码工具ffmpeg

ffmpeg是FFmpeg源代码编译后生成一个可执行程序，其可以作为命令行工具使用。

首先列举一个简单的例子：

./ffmpeg -i input.mp4 output.avi

这是一条简单的ffmpeg命令，可以看到，ffmpeg通过-i参数将input.mp4作为输入源输入，然后进行转码与转封装操作，输出到output.avi中，这条命令主要做了如下工作：

1）获得输入源input.mp4

2）转码

3）输出文件output.avi

看似简单的两步主要的工作，其实远远不止是从后缀名为MP4的文件输出成后缀名为AVI的文件，因为在ffmpeg中，MP4与AVI是两种文件封装格式，并不是后缀名就可以决定的，例如上面的命令行同样可以写成这样：

./ffmpeg -i input.mp4 -f avi output.dat

这条ffmpeg命令相当于前面的那条命令做了一些改变，加了一个“-f”进行约束，“-f"参数的工作非常重要，它制定了输出文件的容器格式，所以可以看到输出的文件为output.dat，文件后缀名为.dat，但是其主要工作依然与之前的指令相同。

ffmpeg的主要工作流程相对比较简单，具体如下：

1）解封装

2）解码

3）编码

4）封装

其中需要经过6个步骤，具体如下：

1）读取输入源

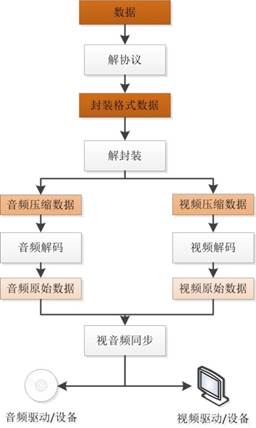
2）进行音视频的解封装（调用libavformat中的接口实现）

3）解码每一帧音视频数据（调用libavcodec中的接口实现）

4）编码每一帧音视频数据（调用libavcodec中的接口实现）

5）进行音视频的重新封装（调用libavformat中的接口实现）

6）输出到目标



具体的使用方法可以参考：

详细的使用说明（英文）：http://ffmpeg.org/ffmpeg.html

## 3.7 FFmpeg的播放器ffplay

FFmpeg不但可以提供转码、转封装等功能，同时还提供了播放器相关功能，使用FFmpeg的avformat与avcodec，可以播放各种媒体文件或者流。

ffplay是FFmpeg源代码编译后生成的另一个可执行程序，与ffmpeg在FFmpeg项目中充当的角色基本相同，可以作为测试工具进行使用，ffplay提供了音视频显示和播放相关的图像信息，也可以作为很多音视频数据的图形化分析工具，通过ffplay可以看到视频图像的运动估计方向、音频数据的波形等。

详细的使用说明（英文）：http://ffmpeg.org/ffplay.html

## 3.8 FFmpeg的多媒体分析器ffprobe

ffprode也是FFmpeg源码编译后生成的一个可执行程序。ffprode是一个非常强大的多媒体分析工具，可以从媒体文件或者媒体流中获得你想要了解的媒体信息，比如音频的参数、视频的参数、媒体容器的参数信息等。

例如它可以帮助分析某个媒体容器中的音频是什么编码格式、视频是什么编码格式，同时还可以得到媒体文件中媒体的总时长、复合码率等信息。使用ffprode可以分析媒体文件中每个包的长度、包的类型、帧的信息等。

详细的使用说明（英文）：<http://ffmpeg.org/ffprobe.html>

# 4. ffmpeg RTMP 相关命令

**4.1 将文件当做直播送至live**

ffmpeg -re -i localFile.mp4 -c copy -f flv rtmp://server/live/streamName

**4.2、将直播媒体保存至本地文件**

ffmpeg -i rtmp://server/live/streamName -c copy dump.flv

**4.3、将其中一个直播流，视频改用h264压缩，音频不变，送至另外一个直播服务流**

ffmpeg -i rtmp://server/live/originalStream -c:a copy -c:v libx264 -vpre slow -f flv rtmp://server/live/h264Stream

**4.4、将其中一个直播流，视频改用h264压缩，音频改用faac压缩，送至另外一个直播服务流**

ffmpeg -i rtmp://server/live/originalStream -c:a libfaac -ar 44100 -ab 48k -c:v libx264 -vpre slow -vpre baseline -f flv rtmp://server/live/h264Stream

**4.5、将其中一个直播流，视频不变，音频改用faac压缩，送至另外一个直播服务流**

ffmpeg -i rtmp://server/live/originalStream -acodec libfaac -ar 44100 -ab 48k -vcodec copy -f flv rtmp://server/live/h264\_AAC\_Stream

**4.6、将一个高清流，复制为几个不同视频清晰度的流重新发布，其中音频不变**

ffmpeg -re -i rtmp://server/live/high\_FMLE\_stream -acodec copy -vcodec x264lib -s 640×360 -b 500k -vpre medium -vpre baseline rtmp://server/live/baseline\_500k -acodec copy -vcodec x264lib -s 480×272 -b 300k -vpre medium -vpre baseline rtmp://server/live/baseline\_300k -acodec copy -vcodec x264lib -s 320×200 -b 150k -vpre medium -vpre baseline rtmp://server/live/baseline\_150k -acodec libfaac -vn -ab 48k rtmp://server/live/audio\_only\_AAC\_48k

**4.7、功能一样，只是采用-x264opts选项**

ffmpeg -re -i rtmp://server/live/high\_FMLE\_stream -c:a copy -c:v x264lib -s 640×360 -x264opts bitrate=500:profile=baseline:preset=slow rtmp://server/live/baseline\_500k -c:a copy -c:v x264lib -s 480×272 -x264opts bitrate=300:profile=baseline:preset=slow rtmp://server/live/baseline\_300k -c:a copy -c:v x264lib -s 320×200 -x264opts bitrate=150:profile=baseline:preset=slow rtmp://server/live/baseline\_150k -c:a libfaac -vn -b:a 48k rtmp://server/live/audio\_only\_AAC\_48k

**4.8、将当前摄像头及音频通过DSSHOW采集，视频h264、音频faac压缩后发布**

ffmpeg -r 25 -f dshow -s 640×480 -i video=”video source name”:audio=”audio source name” -vcodec libx264 -b 600k -vpre slow -acodec libfaac -ab 128k -f flv rtmp://server/application/stream\_name

**4.9、将一个JPG图片经过h264压缩循环输出为mp4视频**

ffmpeg.exe -i INPUT.jpg -an -vcodec libx264 -coder 1 -flags +loop -cmp +chroma -subq 10 -qcomp 0.6 -qmin 10 -qmax 51 -qdiff 4 -flags2 +dct8x8 -trellis 2 -partitions +parti8x8+parti4x4 -crf 24 -threads 0 -r 25 -g 25 -y OUTPUT.mp4

**4.10、将普通流视频改用h264压缩，音频不变，送至高清流服务(新版本FMS live=1)**

ffmpeg -i rtmp://server/live/originalStream -c:a copy -c:v libx264 -vpre slow -f flv “rtmp://server/live/h264Stream live=1″