# ffmedia介绍

ffmedia是生产者、消费者模式，一个生产者可以有多个消费者。vi路径下主要是输入模块，vp下主要是中间处理模块，vo下主要是输出模块。具体使用可参考demo

## 各个模块简单使用说明

输入模块不需要设置生产者；非输入模块必须设置生产者（setProductor()），以接收来自生产者的输出数据。各个模块需要在调用init之前设置模块的buffer数量（setBufferCount()）。具体可参考demo.cpp

### vi:

#### 相机模块ModuleCam:

ModuleCam(const char\* dev);

dev 相机设备节点。

#### 文件读入模块 ModuleFileReader:

ModuleFileReader(const char\* path, bool loop\_play);

path 为读取文件路径，loopplay 将决定是否循环读取文件。如果读取的是yuv等没有图像参数的裸流文件，需要手动创建图像描述然后设置进模块（可参考demo.cpp），目前支持mkv、mp4、h265、h264及其他裸流文件读入。

const uint8\_t\* audioExtraData(); 在模块init()之后便可调用该接口获取音频附加数据；

unsigned audioExtraDataSize(); 获取音频附加数据的大小；

const uint8\_t\* videoExtraData(); 同上；

unsigned videoExtraDataSize(); 同上；

int64\_t getFileReaderMaxSeek(); 如果是mp4 mkv获取视频播放时长；裸流获取文件最大大小；

int setFileReaderSeek(int64\_t ms\_time); 如果是📛mp4 mkv 切换到该时间点获取帧，单位毫秒。

#### rtsp客户端模块 ModuleRtspClient：

ModuleRtspClient(const char\* url, int \_stream\_type = 0);

url是传入rtsp地址，stream\_type是0为udp协议，1为tcp协议，2为多播协议。

const uint8\_t\* videoExtraData(); 同文件输入模块。

unsigned videoExtraDataSize();

const uint8\_t\* audioExtraData();

unsigned audioExtraDataSize();

使用参考demo.cpp

### vp:

#### mpp解码模块 ModuleMppDec:

ModuleMppDec(ImagePara& input\_para);

ModuleMppDec(ImagePara& input\_para, DecodeType type);

input\_para是上个模块的output\_para, type是输入图像的格式，不设置则从input\_para中获取。

#### mpp编码模块 ModuleMppEnc:

ModuleMppEnc(EncodeType type, ImagePara& input\_para, int fps, int gop, int bps, EncodeRcMode mode, EncodeQuality quality, EncodeProfile profile);

type是编码类型，input\_para是上个模块的output\_para，fps是视频帧数，gop是I帧间隔， bps是码率， mode是速率控制模式， quality是图像质量参数，profile 是 profile\_idc 参数；

ModuleMppEnc(EncodeType type, ImagePara& input\_para);

该接口默认fps = 30, gop = 60, bps = 2048, mode = ENCODE\_RC\_MODE\_CBR, quality = ENCODE\_QUALITY\_BEST, profile = ENCODE\_PROFILE\_HIGH。

#### RGA模块 ModuleRga：

ModuleRga(ImagePara& input\_para, ImagePara& output\_para, RgaRotate rotate);

input\_para上个模块的输出图像格式， output\_para需要输出的图像格式， rotate输出图像旋转方式。

MediaBuffer\* newModuleMediaBuffer(VideoBuffer::BUFFER\_TYPE buffer\_type = VideoBuffer::BUFFER\_TYPE::DRM\_BUFFER\_CACHEABLE);

申请一块和rga内部一样的buf出来。

MediaBuffer\* exportUseMediaBuffer(MediaBuffer\* match\_buffer, MediaBuffer\* input\_buffer, int flag);

match\_buffer 传入rga内部模块buf（一般可在rga回调函数拿到），input\_buffer 为 newModuleMediaBuffer得到的buf或为NULL在内部申请，flag 为0 使用input\_buffer将match\_buffer在rga内存池替换出来（0拷贝）；为1则将match\_buffer的数据拷贝到input\_buffer。函数失败返回NULL,成功返回buf。

#### aac 解码模块 ModuleAacDec:

ModuleAacDec(const uint8\_t\* \_extradata, unsigned \_extradata\_size, int \_sample\_rate, int \_nb\_channels = -1);

extradata 是音频附加数据， extradata\_size 音频附加数据大小（从文件读入模块或rtsp客户端模块拿到）， \_sample\_rate和\_nb\_channels都可从附加数据解析出来。

void setAlsaDevice(string dev) { a\_dev = dev; }

设置alsa设备名称，设置则进行音频播放。

#### aac 编码模块 ModuleAacEnc：

### 输出VO：

#### drm显示模块 ModuleDrmDisplay：

ModuleDrmDisplay(ImagePara& input\_para);

input\_para为上个模块的图像输出。

int move(uint32\_t x, uint32\_t y);

可在显示时改变显示位置。

void setPlanePara(uint32\_t fmt, uint32\_t plane\_id, PLANE\_TYPE plane\_type, uint32\_t plane\_zpos, uint32\_t plane\_linear);

设置图层，fmt 为图像格式；plane\_id 为图层id；plane\_type 为图层类型；plane\_zpos 为图层显示z轴位置；

void setPlaneSize(uint32\_t x, uint32\_t y, uint32\_t w, uint32\_t h);

设置图层显示位置及大小。

void setWindowSize(uint32\_t x, uint32\_t y, uint32\_t w, uint32\_t h);

设置窗口显示位置及大小。

void getDisplayPlaneSize(uint32\_t\* h, uint32\_t\* v);

获取图层大小。

#### 文件写入模块 ModuleFileWriter：

ModuleFileWriter(ImagePara& para, string path);

para为上个模块的图像输出；path 为保存文件的路径，如果后缀有.mp4则封装成mp4文件保存，如果有.mkv则封装陈mkv文件保存，其他则保存成裸流文件。

int restart(string file\_name);

在模块init之后可更换另一个文件名保存，不改变封装方式。

void setMaxFrameCount(uint32\_t frame\_count);

设置保存的文件的最大帧数，超过则在原文件名后加1重新打开文件保存。

void setVideoParameter(int width, int height, media\_codec\_t type);

设置视频参数。一般不需要设置，会从para解析。

void setVideoExtraData(const uint8\_t\* extra\_data, unsigned extra\_size);

设置视频附加数据。他的生产者是编码模块可以不用设置，反之需要设置。

void setAudioParameter(int channel\_count, int bit\_per\_sample, int sample\_rate, media\_codec\_t type);

void setAudioExtraData(const uint8\_t\* extra\_data, unsigned extra\_size);

同上。

#### rtsp服务端模块 ModuleRtspServer：

ModuleRtspServer(ImagePara& para, const char\* path, int port);

para 为生产者模块的图像输出；path 为推流的路径；port为推流的端口；

### 其他模块：

#### 音视频同步模块 Synchronize：

Synchronize(SynchronizeType \_type);

type 选择基于音频同步、视频同步及系统世界同步。

如果没有同步模块那么播放、推流速度就取决与解码器及文件获取的速度了，并且有音视频不同步问题等，所以需要为输出模块设置同步模块（module->setSynchronize(sync);）

## python的支持：

在dist目录下安装最新版本的.whl，即可使用demo.py.

python使用的接口和c++版本一致，可参考demo.py和demo.cpp编写。