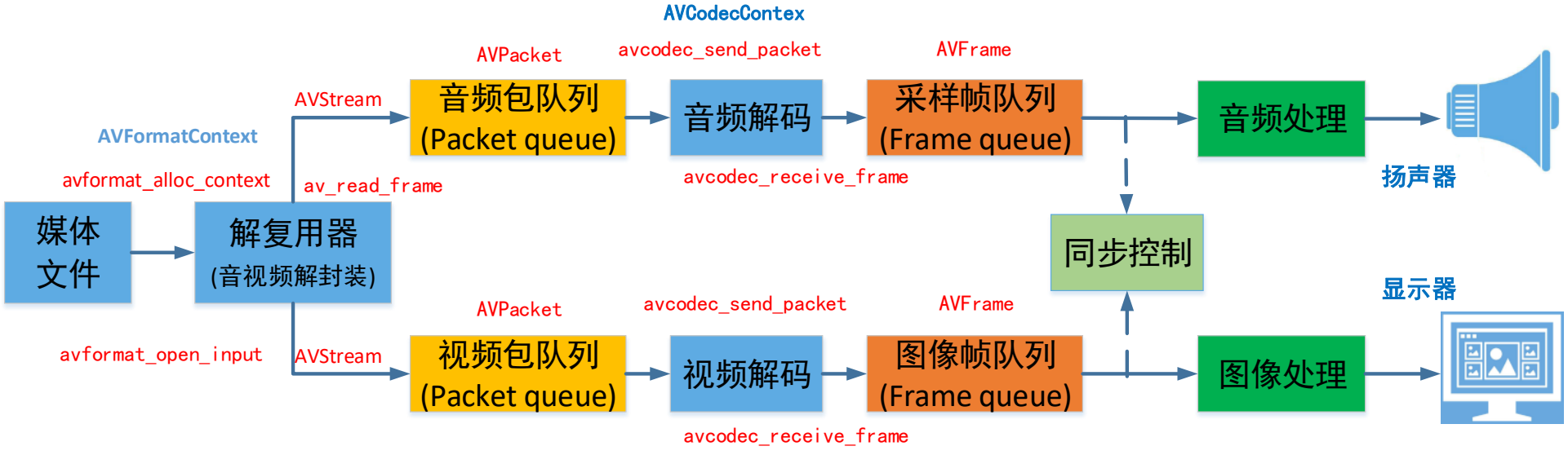
C/C++音视频高级开发 FFmpeg内存模型

FFmpeg内存模型




```
int avcodec_send_packet(AVCodecContext *avctx, const AVPacket *avpkt);
int avcodec_receive_frame(AVCodecContext *avctx, AVFrame *frame);
```

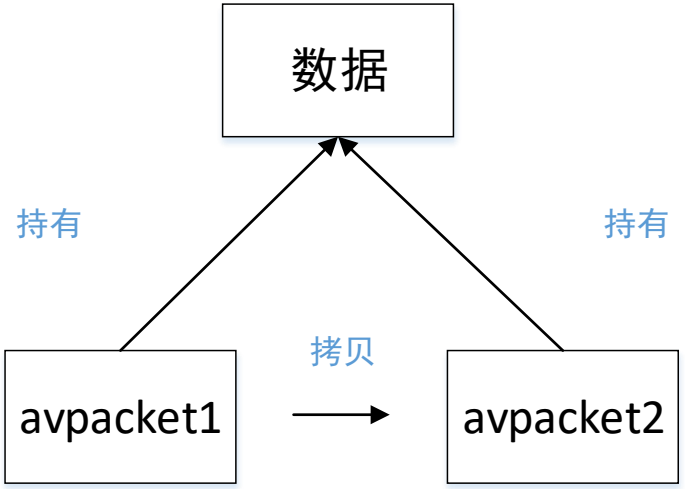
问题：

（1）从av_read_frame读取到一个AVPacket后怎么放入队列？

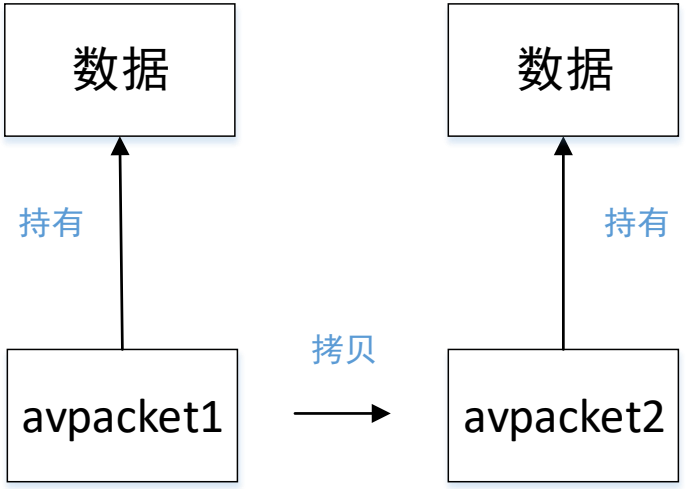
（2）从avcodec_recevice_frame读取到一个AVFrame后又怎么放入队列？

FFmpeg内存模型

- **从现有的Packet拷贝一个新Packet的时候，有两种情况：**
  - ①两个Packet的buf引用的是同一数据缓存空间，这时候要注意数据缓存空间的释放问题；
  - ②两个Packet的buf引用不同的数据缓存空间，每个Packet都有数据缓存空间的copy；

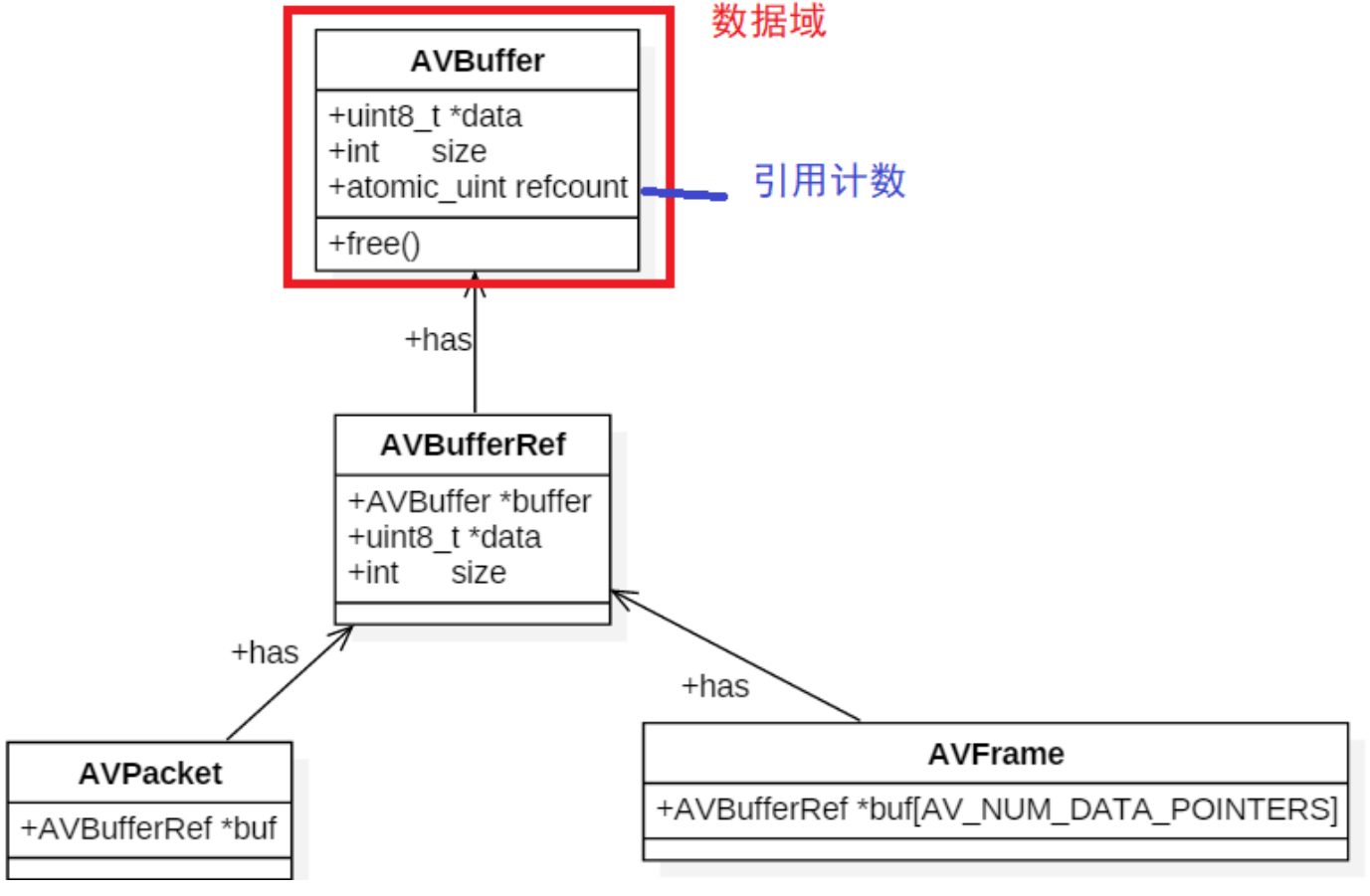


① 数据共享



② 数据独立

FFmpeg内存模型

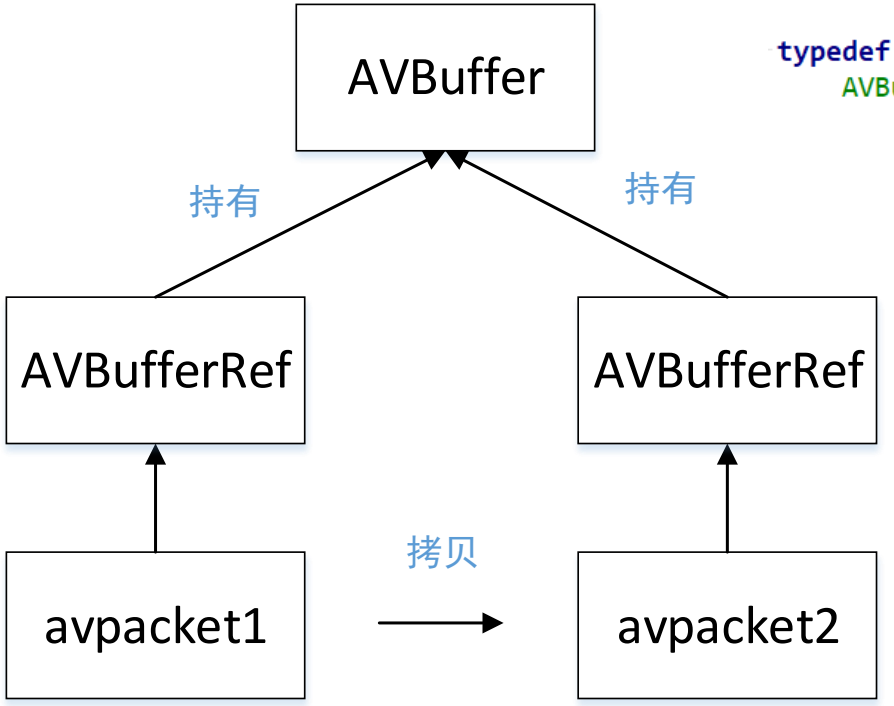

数据域
AVBuffer
+uint8_t *data
+int size
+atomic_uint refcount
+free()
引用计数
+has
AVBufferRef
+AVBuffer *buffer
+uint8_t *data
+int size
+has
+has
AVPacket
+AVBufferRef *buf
AVFrame
+AVBufferRef *buf[AV_NUM_DATA_POINTERS]

FFmpeg内存模型

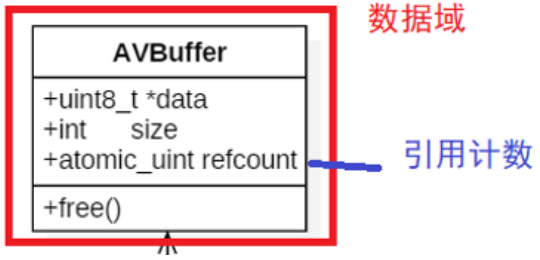
- 更为精确的模型

```
typedef struct AVPacket {
    /**
     * A reference to the reference-counted buffer where the packet data is
     * stored.
     * May be NULL, then the packet data is not reference-counted.
     */
    AVBufferRef *buf;
    /**
```

```
typedef struct AVBufferRef {
    AVBuffer *buffer;
```



实际共同持有的是AVBuffer



FFmpeg内存模型–引用计数

- 对于多个AVPacket共享同一个缓存空间，FFmpeg使用的引用计数的机制（reference-count）：
  - 初始化引用计数为0，只有真正分配AVBuffer的时候，引用计数初始化为1;
  - 当有新的Packet引用共享的缓存空间时，就将引用计数+1;
  - 当释放了引用共享空间的Packet，就将引用计数-1；引用计数为0时，就释放掉引用的缓存空间AVBuffer。
- AVFrame也是采用同样的机制。

AVPacket常用API

函数原型	说明
AVPacket *av_packet_alloc(void);	分配AVPacket 这个时候和buffer没有关系
void av_packet_free(AVPacket **pkt);	释放AVPacket 和_alloc对应
void av_init_packet(AVPacket *pkt);	初始化AVPacket 只是单纯初始化pkt字段
int av_new_packet(AVPacket *pkt, int size);	给AVPacket的buf分配内存，引用计数初始化为1
int av_packet_ref(AVPacket *dst, const AVPacket *src);	增加引用计数
void av_packet_unref(AVPacket *pkt);	减少引用计数
void av_packet_move_ref(AVPacket *dst, AVPacket *src);	转移引用计数
AVPacket *av_packet_clone(const AVPacket *src);	等于 av_packet_alloc()+av_packet_ref()

具体见代码：6-01-buffer

AVFrame常用API

函数原型	说明
AVFrame *av_frame_alloc(void);	分配AVFrame
void av_frame_free(AVFrame **frame);	释放AVFrame
int av_frame_ref(AVFrame *dst, const AVFrame *src);	增加引用计数
void av_frame_unref(AVFrame *frame);	减少引用计数
void av_frame_move_ref(AVFrame *dst, AVFrame *src);	转移引用计数
int av_frame_get_buffer(AVFrame *frame, int align);	根据AVFrame分配内存
AVFrame *av_frame_clone(const AVFrame *src);	等于 av_frame_alloc()+av_frame_ref()

具体见代码：6-01-buffer