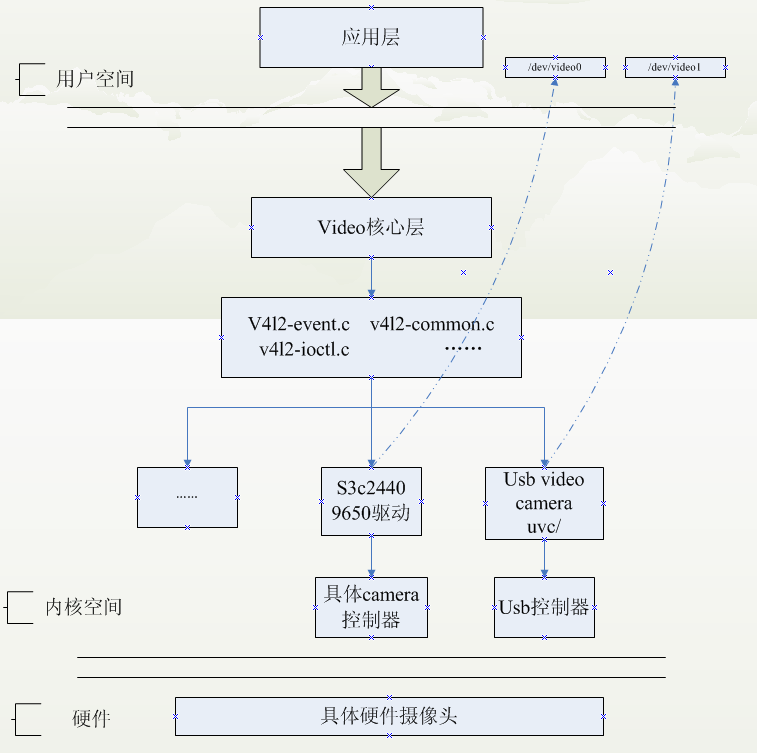
[和菜鸟一起学linux之V4L2摄像头应用流程](http://blog.csdn.net/eastmoon502136/article/details/8190262)

       对于v4l2，上次是在调试收音机驱动的时候用过，其他也就只是用i2c配置一些寄存器就可以了。那时只是粗粗的了解了，把收音机当作v4l2的设备后会在/dev目录下生成一个radio的节点。然后就可以操作了。后来就没怎么接触了。这周，需要调试下usb的摄像头。因为有问题，所以就要跟进，于是也就要开始学习下linux的v4l2了。看到一篇很不错的文章，下面参考这篇文章，加上自己的一些见解，做一些总结把。

       Video for Linuxtwo(Video4Linux2)简称V4L2，是V4L的改进版。V4L2是linux操作系统下用于采集图片、视频和音频数据的API接口，配合适当的视频采集设备和相应的驱动程序，可以实现图片、视频、音频等的采集。在远程会议、可视电话、视频监控系统和嵌入式多媒体终端中都有广泛的应用。

在Linux下，所有外设都被看成一种特殊的文件，成为“设备文件”，可以象访问普通文件一样对其进行读写。一般来说，采用V4L2驱动的摄像头设备文件是/dev/video0。V4L2支持两种方式来采集图像：内存映射方式(mmap)和直接读取方式(read)。V4L2在include/linux/videodev.h文件中定义了一些重要的数据结构，在采集图像的过程中，就是通过对这些数据的操作来获得最终的图像数据。Linux系统V4L2的能力可在Linux内核编译阶段配置，默认情况下都有此开发接口。

       而摄像头所用的主要是capature了，视频的捕捉，具体linux的调用可以参考下图。



应用程序通过V4L2进行视频采集的原理

V4L2支持内存映射方式(mmap)和直接读取方式(read)来采集数据，前者一般用于连续视频数据的采集，后者常用于静态图片数据的采集，本文重点讨论内存映射方式的视频采集。

应用程序通过V4L2接口采集视频数据分为五个步骤：

首先，打开视频设备文件，进行视频采集的参数初始化，通过V4L2接口设置视频图像的采集窗口、采集的点阵大小和格式;

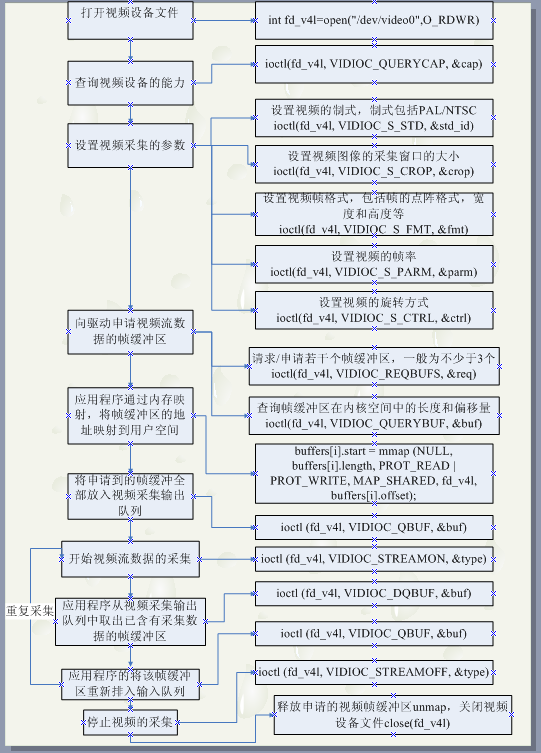
其次，申请若干视频采集的帧缓冲区，并将这些帧缓冲区从内核空间映射到用户空间，便于应用程序读取/处理视频数据;

第三，将申请到的帧缓冲区在视频采集输入队列排队，并启动视频采集;

第四，驱动开始视频数据的采集，应用程序从视频采集输出队列取出帧缓冲区，处理完后，将帧缓冲区重新放入视频采集输入队列，循环往复采集连续的视频数据;

第五，停止视频采集。

具体的程序实现流程可以参考下面的流程图:

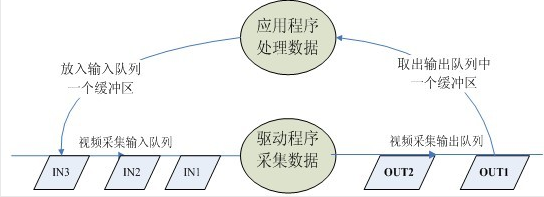


其实其他的都比较简单，就是通过ioctl这个接口去设置一些参数。最主要的就是buf管理。他有一个或者多个输入队列和输出队列。

启动视频采集后，驱动程序开始采集一帧数据，把采集的数据放入视频采集输入队列的第一个帧缓冲区，一帧数据采集完成，也就是第一个帧缓冲区存满一帧数据后，驱动程序将该帧缓冲区移至视频采集输出队列，等待应用程序从输出队列取出。驱动程序接下来采集下一帧数据，放入第二个帧缓冲区，同样帧缓冲区存满下一帧数据后，被放入视频采集输出队列。

应用程序从视频采集输出队列中取出含有视频数据的帧缓冲区，处理帧缓冲区中的视频数据，如存储或压缩。

最后，应用程序将处理完数据的帧缓冲区重新放入视频采集输入队列,这样可以循环采集，如图所示。



每一个帧缓冲区都有一个对应的状态标志变量，其中每一个比特代表一个状态

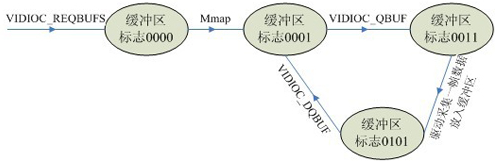
　　V4L2\_BUF\_FLAG\_UNMAPPED 0B0000

　　V4L2\_BUF\_FLAG\_MAPPED 0B0001

　　V4L2\_BUF\_FLAG\_ENQUEUED 0B0010

　　V4L2\_BUF\_FLAG\_DONE 0B0100

　　缓冲区的状态转化如图所示。



下面的程序注释的很好，就拿来参考下：

V4L2 编程

1. 定义

V4L2(Video ForLinux Two) 是内核提供给应用程序访问音、视频驱动的统一接口。

2. 工作流程：

打开设备－> 检查和设置设备属性－>设置帧格式－> 设置一种输入输出方法（缓冲区管理）－> 循环获取数据－> 关闭设备。

3. 设备的打开和关闭：

#include<fcntl.h>

int open(constchar \*device\_name, int flags);

#include <unistd.h>

int close(intfd);

例：

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. int fd=open(“/dev/video0”,O\_RDWR);// 打开设备
2. close(fd);// 关闭设备

注意：V4L2 的相关定义包含在头文件<linux/videodev2.h>中.

4. 查询设备属性： VIDIOC\_QUERYCAP

相关函数：

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. int ioctl(intfd, int request, struct v4l2\_capability \*argp);

相关结构体：

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_capability
2. {
3. \_\_u8 driver[16];     // 驱动名字
4. \_\_u8 card[32];       // 设备名字
5. \_\_u8bus\_info[32]; // 设备在系统中的位置
6. \_\_u32 version;       // 驱动版本号
7. \_\_u32capabilities;  // 设备支持的操作
8. \_\_u32reserved[4]; // 保留字段
9. };
10. capabilities 常用值:
11. V4L2\_CAP\_VIDEO\_CAPTURE    // 是否支持图像获取

例：显示设备信息

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_capability cap;
2. ioctl(fd,VIDIOC\_QUERYCAP,&cap);
3. printf(“DriverName:%s/nCard Name:%s/nBus info:%s/nDriverVersion:%u.%u.%u/n”,cap.driver,cap.card,cap.bus\_info,(cap.version**>>**16)&0XFF,(cap.version**>>**8)&0XFF,cap.version&OXFF);

5. 帧格式：

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. VIDIOC\_ENUM\_FMT// 显示所有支持的格式
2. int ioctl(intfd, int request, struct v4l2\_fmtdesc \*argp);
3. structv4l2\_fmtdesc
4. {
5. \_\_u32 index;   // 要查询的格式序号，应用程序设置
6. enumv4l2\_buf\_type type;     // 帧类型，应用程序设置
7. \_\_u32 flags;    // 是否为压缩格式
8. \_\_u8       description[32];      // 格式名称
9. \_\_u32pixelformat; // 格式
10. \_\_u32reserved[4]; // 保留
11. };

例：显示所有支持的格式

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_fmtdesc fmtdesc;
2. fmtdesc.index=0;
3. fmtdesc.type=V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
4. printf("Supportformat:/n");
5. while(ioctl(fd,VIDIOC\_ENUM\_FMT,&fmtdesc)!=-1)
6. {
7. printf("/t%d.%s/n",fmtdesc.index+1,fmtdesc.description);
8. fmtdesc.index++;
9. }

// 查看或设置当前格式

VIDIOC\_G\_FMT,VIDIOC\_S\_FMT

// 检查是否支持某种格式

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. VIDIOC\_TRY\_FMT
2. int ioctl(intfd, int request, struct v4l2\_format \*argp);
3. structv4l2\_format
4. {
5. enumv4l2\_buf\_type type;// 帧类型，应用程序设置
6. union fmt
7. {
8. structv4l2\_pix\_format pix;// 视频设备使用
9. structv4l2\_window win;
10. structv4l2\_vbi\_format vbi;
11. structv4l2\_sliced\_vbi\_format sliced;
12. \_\_u8raw\_data[200];
13. };
14. };

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_pix\_format
2. {
3. \_\_u32 width;  // 帧宽，单位像素
4. \_\_u32 height;  // 帧高，单位像素
5. \_\_u32pixelformat; // 帧格式
6. enum v4l2\_fieldfield;
7. \_\_u32bytesperline;
8. \_\_u32 sizeimage;
9. enumv4l2\_colorspace colorspace;
10. \_\_u32 priv;
11. };

例：显示当前帧的相关信息

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_format fmt;
2. fmt.type=V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
3. ioctl(fd,VIDIOC\_G\_FMT,&fmt);
4. printf(“Currentdata format information:
5. /n/twidth:%d/n/theight:%d/n”,fmt.fmt.width,fmt.fmt.height);
6. structv4l2\_fmtdesc fmtdesc;
7. fmtdesc.index=0;
8. fmtdesc.type=V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
9. while(ioctl(fd,VIDIOC\_ENUM\_FMT,&fmtdesc)!=-1)
10. {
11. if(fmtdesc.pixelformat& fmt.fmt.pixelformat)
12. {
13. printf(“/tformat:%s/n”,fmtdesc.description);
14. break;
15. }
16. fmtdesc.index++;
17. }

例：检查是否支持某种帧格式

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_format fmt;
2. fmt.type=V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
3. fmt.fmt.pix.pixelformat=V4L2\_PIX\_FMT\_RGB32;
4. if(ioctl(fd,VIDIOC\_TRY\_FMT,&fmt)==-1)
5. if(errno==EINVAL)
6. printf(“notsupport format RGB32!/n”);

6. 图像的缩放

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. VIDIOC\_CROPCAP
2. int ioctl(int fd,int request, struct v4l2\_cropcap \*argp);
3. structv4l2\_cropcap
4. {
5. enumv4l2\_buf\_type type;// 应用程序设置
6. struct v4l2\_rectbounds;//     最大边界
7. struct v4l2\_rectdefrect;// 默认值
8. structv4l2\_fract pixelaspect;
9. };

// 设置缩放

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. VIDIOC\_G\_CROP,VIDIOC\_S\_CROP
2. int ioctl(intfd, int request, struct v4l2\_crop \*argp);
3. int ioctl(intfd, int request, const struct v4l2\_crop \*argp);
4. struct v4l2\_crop
5. {
6. enumv4l2\_buf\_type type;// 应用程序设置
7. struct v4l2\_rectc;
8. }

7. 申请和管理缓冲区，应用程序和设备有三种交换数据的方法，直接read/write ，内存映射(memorymapping) ，用户指针。这里只讨论 memorymapping.

// 向设备申请缓冲区

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. VIDIOC\_REQBUFS
2. int ioctl(intfd, int request, struct v4l2\_requestbuffers \*argp);
3. structv4l2\_requestbuffers
4. {
5. \_\_u32 count;  // 缓冲区内缓冲帧的数目
6. enumv4l2\_buf\_type type;     // 缓冲帧数据格式
7. enum v4l2\_memorymemory;       // 区别是内存映射还是用户指针方式
8. \_\_u32 reserved[2];
9. };
11. enum v4l2\_memoy{V4L2\_MEMORY\_MMAP,V4L2\_MEMORY\_USERPTR};
12. //count,type,memory都要应用程序设置

例：申请一个拥有四个缓冲帧的缓冲区

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_requestbuffers req;
2. req.count=4;
3. req.type=V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
4. req.memory=V4L2\_MEMORY\_MMAP;
5. ioctl(fd,VIDIOC\_REQBUFS,&req);

获取缓冲帧的地址，长度：

VIDIOC\_QUERYBUF

int ioctl(intfd, int request, struct v4l2\_buffer \*argp);

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_buffer
2. {
3. \_\_u32 index;   //buffer 序号
4. enumv4l2\_buf\_type type;     //buffer 类型
5. \_\_u32 byteused;     //buffer 中已使用的字节数
6. \_\_u32 flags;    // 区分是MMAP 还是USERPTR
7. enum v4l2\_fieldfield;
8. struct timevaltimestamp;// 获取第一个字节时的系统时间
9. structv4l2\_timecode timecode;
10. \_\_u32 sequence;// 队列中的序号
11. enum v4l2\_memorymemory;//IO 方式，被应用程序设置
12. union m
13. {
14. \_\_u32 offset;// 缓冲帧地址，只对MMAP 有效
15. unsigned longuserptr;
16. };
17. \_\_u32 length;// 缓冲帧长度
18. \_\_u32 input;
19. \_\_u32 reserved;
20. };

MMAP ，定义一个结构体来映射每个缓冲帧。

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. Struct buffer
2. {
3. void\* start;
4. unsigned intlength;
5. }\*buffers;

#include<sys/mman.h>

void \*mmap(void\*addr, size\_t length, int prot, int flags, int fd, off\_t offset);

//addr 映射起始地址，一般为NULL ，让内核自动选择

//length 被映射内存块的长度

//prot 标志映射后能否被读写，其值为PROT\_EXEC,PROT\_READ,PROT\_WRITE,PROT\_NONE

//flags 确定此内存映射能否被其他进程共享，MAP\_SHARED,MAP\_PRIVATE

//fd,offset, 确定被映射的内存地址

返回成功映射后的地址，不成功返回MAP\_FAILED ((void\*)-1);

int munmap(void\*addr, size\_t length);// 断开映射

//addr 为映射后的地址，length 为映射后的内存长度

例：将四个已申请到的缓冲帧映射到应用程序，用buffers 指针记录。

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. buffers =(buffer\*)calloc (req.count, sizeof (\*buffers));
2. if (!buffers) {
3. fprintf (stderr,"Out of memory/n");
4. exit(EXIT\_FAILURE);
5. }

// 映射

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. for (unsignedint n\_buffers = 0; n\_buffers **<** **req.count**; ++n\_buffers) {
2. struct v4l2\_bufferbuf;
3. memset(&buf,0,sizeof(buf));
4. buf.type =V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
5. buf.memory =V4L2\_MEMORY\_MMAP;
6. buf.index =n\_buffers;
7. // 查询序号为n\_buffers 的缓冲区，得到其起始物理地址和大小
8. if (-1 == ioctl(fd, VIDIOC\_QUERYBUF, &buf))
9. exit(-1);
10. buffers[n\_buffers].length= buf.length;
11. // 映射内存
12. buffers[n\_buffers].start=mmap (NULL,buf.length,PROT\_READ | PROT\_WRITE ,MAP\_SHARED,fd, buf.m.offset);
13. if (MAP\_FAILED== buffers[n\_buffers].start)
14. exit(-1);
15. }

8. 缓冲区处理好之后，就可以开始获取数据了

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. // 启动/ 停止数据流
2. VIDIOC\_STREAMON,VIDIOC\_STREAMOFF
3. int ioctl(intfd, int request, const int \*argp);
4. //argp 为流类型指针，如V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE.
5. 在开始之前，还应当把缓冲帧放入缓冲队列：
6. VIDIOC\_QBUF// 把帧放入队列
7. VIDIOC\_DQBUF// 从队列中取出帧
8. int ioctl(intfd, int request, struct v4l2\_buffer \*argp);

例：把四个缓冲帧放入队列，并启动数据流

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. unsigned int i;
2. enum v4l2\_buf\_typetype;
3. // 将缓冲帧放入队列
4. for (i = 0; i**<** **4**; ++i)
5. {
6. structv4l2\_buffer buf;
7. buf.type =V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
8. buf.memory =V4L2\_MEMORY\_MMAP;
9. buf.index = i;
10. ioctl (fd,VIDIOC\_QBUF, &buf);
11. }
12. type =V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
13. ioctl (fd,VIDIOC\_STREAMON, &type);

例：获取一帧并处理

**[html]** [view plain](http://blog.csdn.net/eastmoon502136/article/details/8190262) [copy](http://blog.csdn.net/eastmoon502136/article/details/8190262)

1. structv4l2\_buffer buf;
2. CLEAR (buf);
3. buf.type =V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;
4. buf.memory =V4L2\_MEMORY\_MMAP;
5. // 从缓冲区取出一个缓冲帧
6. ioctl (fd,VIDIOC\_DQBUF, &buf);
7. // 图像处理
8. process\_image(buffers[buf.index].start);
9. // 将取出的缓冲帧放回缓冲区
10. ioctl (fd, VIDIOC\_QBUF,&buf);

至于驱动的实现，可以参考内核中，我是用usb摄像头的，所以，其实现都是好的。主要就是应用程序的实现了。驱动都哦在uvc目录下面，这个待理解。