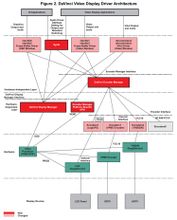
V4L2-百度百科

**V4L2(video 4 linux 2)**

V4L2有一段历史了。大约在1998的秋天，它的光芒第一次出现在Bill Dirks 的眼中。经过长足的发展，它于2002年11 月，发布2.5.46 时，融入了内核主干之中。然而直到今天，仍有一部分内核驱不支持新的API，这种新旧API 的转换工作仍在进行。同时，V4L2 API也在发展，并在2.6.18 版本中进行了一些重大的改变。支持V4L2的应用依旧相对较少。V4L2在设计时，是要支持很多广泛的设备的，它们之中只有一部分在本质上是真正的视频设备：

[](http://baike.baidu.com/picview/5494174/5531619/0/8b13632762d0f703074f377f08fa513d2797c590.html)

v4l2在linux内的系统图[1]

可以支持多种设备,它可以有以下几种接口:

1. [视频采集](http://baike.baidu.com/view/189678.htm)接口(video capture interface):这种应用的设备可以是高频头或者摄像头.V4L2的最初设计就是应用于这种功能的.

2. 视频输出接口(video output interface):可以驱动计算机的外围视频图像设备--像可以输出电视信号格式的设备.

3. 直接传输视频接口(video overlay interface):它的主要工作是把从[视频采集](http://baike.baidu.com/view/189678.htm" \t "_blank)设备采集过来的信号直接输出到[输出设备](http://baike.baidu.com/view/632676.htm)之上,而不用经过系统的CPU.

4. 视频间隔消隐信号接口(VBI interface):它可以使应用可以访问传输消隐期的视频信号.

5. 收音机接口(radio interface):可用来处理从AM或FM高频头设备接收来的音频流.

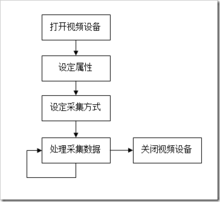
[编辑本段](http://baike.baidu.com/view/5494174.htm)**主要功能**

使程序有发现设备和操作设备的能力.它主要是用一系列的回调函数来实现这些功能。像设置摄像头的频率、帧频、视频压缩格式和图像参数等等。当然也可以用于其他多媒体的开发，如音频等。

但是此框架只能运行在Linux操作系统之上。

[编辑本段](http://baike.baidu.com/view/5494174.htm)**v4L2编程**

v4L2是针对uvc免驱usb设备的编程框架[2]，主要用于采集usb摄像头等，编程模式如下：

[](http://baike.baidu.com/picview/5494174/5531619/0/838ba61ea8d3fd1fecbe91d1304e251f94ca5fa1.html)

编程

**设置采集方式**

打开视频设备后，可以设置该视频设备的属性，例如裁剪、缩放等。这一步是可选的。在Linux编程中，一般使用ioctl函数来对设备的I/O通道进行管理：

extern int ioctl (int \_\_fd, unsigned long int \_\_request, …) \_\_THROW;

\_\_fd：设备的ID，例如刚才用open函数打开视频通道后返回的cameraFd；

\_\_request：具体的命令标志符。

在进行V4L2开发中，一般会用到以下的命令标志符：

VIDIOC\_REQBUFS：分配内存

VIDIOC\_QUERYBUF：把VIDIOC\_REQBUFS中分配的数据缓存转换成物理地址

VIDIOC\_QUERYCAP：查询驱动功能

VIDIOC\_ENUM\_FMT：获取当前驱动支持的视频格式

VIDIOC\_S\_FMT：设置当前驱动的频捕获格式

VIDIOC\_G\_FMT：读取当前驱动的频捕获格式

VIDIOC\_TRY\_FMT：验证当前驱动的显示格式

VIDIOC\_CROPCAP：查询驱动的修剪能力

VIDIOC\_S\_CROP：设置视频信号的边框

VIDIOC\_G\_CROP：读取视频信号的边框

VIDIOC\_QBUF：把数据从缓存中读取出来

VIDIOC\_DQBUF：把数据放回缓存队列

VIDIOC\_STREAMON：开始视频显示函数

VIDIOC\_STREAMOFF：结束视频显示函数

VIDIOC\_QUERYSTD：检查当前视频设备支持的标准，例如PAL或NTSC。

这些IO调用，有些是必须的，有些是可选择的。

**一般操作流程（视频设备）**

1. 打开设备文件。

int fd=open(”/dev/video0″,O\_RDWR);

2. 取得设备的capability，看看设备具有什么功能，比如是否具有视频输入,或者音频输入输出等。VIDIOC\_QUERYCAP,struct v4l2\_capability

v4l2\_std\_id std;

do {

ret= ioctl(fd, VIDIOC\_QUERYSTD, &std);

} while (ret == -1 && errno == EAGAIN);

switch (std) {

case V4L2\_STD\_NTSC:

//……

case V4L2\_STD\_PAL:

//……

}

3. 选择视频输入，一个视频设备可以有多个视频输入。VIDIOC\_S\_INPUT,struct v4l2\_input(可不要)

4. 设置视频的制式和帧格式，制式包括PAL，NTSC，帧的格式个包括宽度和高度等。

VIDIOC\_S\_STD,VIDIOC\_S\_FMT,struct v4l2\_std\_id,struct v4l2\_format

struct v4l2\_format fmt;

/\*

v4l2\_format 结构如下：

struct v4l2\_format

{

enum v4l2\_buf\_type type; // 数据流类型，必须永远是V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE

union

{

struct v4l2\_pix\_format pix;

struct v4l2\_window win;

struct v4l2\_vbi\_format vbi;

\_\_u8 raw\_data[200];

} fmt;

};

struct v4l2\_pix\_format

{

\_\_u32 width; // 宽，必须是16 的倍数

\_\_u32 height; // 高，必须是16 的倍数

\_\_u32 pixelformat; // 视频数据存储类型，例如是YUV 4 ：2 ：2 还是RGB

enum v4l2\_field field;

\_\_u32 bytesperline;

\_\_u32 sizeimage;

enum v4l2\_colorspace colorspace;

\_\_u32 priv;

};

\*/

样例：

memset ( &fmt, 0, sizeof(fmt) );

fmt.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

fmt.fmt.pix.width = 320;

fmt.fmt.pix.height = 240;

fmt.fmt.pix.pixelformat = V4L2\_PIX\_FMT\_JPEG;

if (ioctl(fd, VIDIOC\_S\_FMT, &fmt) < 0）

{

printf("set format failed\n");

//return 0;

}

5. 向驱动申请帧缓冲，一般不超过5个。struct v4l2\_requestbuffers

struct v4l2\_requestbuffers

{

\_\_u32 count; // 缓存数量，也就是说在缓存队列里保持多少张照片

enum v4l2\_buf\_type type; // 数据流类型，必须永远是V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE

enum v4l2\_memory memory; // V4L2\_MEMORY\_MMAP 或 V4L2\_MEMORY\_USERPTR

\_\_u32 reserved[2];

};

样例：

struct v4l2\_requestbuffers req;

memset(&req, 0, sizeof (req));

req.count = 4;

req.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

req.memory = V4L2\_MEMORY\_MMAP;

if (ioctl(fd,VIDIOC\_REQBUFS,&req) == -1)

{

perror("VIDIOC\_REQBUFS error \n");

//return -1;

}

6.申请物理内存

将申请到的帧缓冲映射到用户空间，这样就可以直接操作采集到的帧了，而不必去复制。将申请到的帧缓冲全部入队列，以便存放采集到的数据.VIDIOC\_QBUF,struct v4l2\_buffer

VideoBuffer\* buffers = calloc( req.count, sizeof(VideoBuffer) );

printf("sizeof(VideoBuffer) is %d\n",sizeof(VideoBuffer));

struct v4l2\_buffer buf;

for (numBufs = 0; numBufs < req.count; numBufs++)

{

memset( &buf, 0, sizeof(buf) );

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

buf.index = numBufs;

if (ioctl(fd, VIDIOC\_QUERYBUF, &buf) < 0)

{

printf("VIDIOC\_QUERYBUF error\n");

//return -1;

}

printf("buf len is %d\n",sizeof(buf));

//内存映射

buffers[numBufs].length = buf.length;

buffers[numBufs].offset = (size\_t) buf.m.offset;

buffers[numBufs].start = mmap (NULL, buf.length,PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, buf.m.offset);

printf("buffers.length = %d,buffers.offset = %d ,buffers.start[0] = %d\n",buffers[numBufs].length,buffers[numBufs].offset,buffers[numBufs].start[0]);

printf("buf2 len is %d\n",sizeof(buffers[numBufs].start));

if (buffers[numBufs].start == MAP\_FAILED)

{

perror("buffers error\n");

//return -1;

}

if (ioctl (fd, VIDIOC\_QBUF, &buf) < 0)

{

printf("VIDIOC\_QBUF error\n");

//return -1;

}

}

7. 开始视频的采集。

enum v4l2\_buf\_type type;

type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

if (ioctl (fd, VIDIOC\_STREAMON, &type) < 0)

{

printf("VIDIOC\_STREAMON error\n");

// return -1;

}

8. 出队列以取得已采集数据的帧缓冲，取得原始采集数据。VIDIOC\_DQBUF, 将缓冲重新入队列尾,这样可以循环采集。VIDIOC\_QBUF

if (ioctl(fd, VIDIOC\_DQBUF, &buf) < 0)

{

perror("VIDIOC\_DQBUF failed.\n");

//return -1;

}

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

unsigned char \*ptcur = buffers[numBufs].start;

DEBUG("buf.bytesused = %d \n",buf.bytesused);

int i1;

for(i1=0; i1<buf.bytesused; i1++)

{

if((buffers[numBufs].start[i1] == 0x000000FF) && (buffers[numBufs].start[i1+1] == 0x000000C4))

{

DEBUG("huffman table finded! \nbuf.bytesused = %d\nFFC4 = %d \n",buf.bytesused,i1);

break;

}

}

if(i1 == buf.bytesused)printf("huffman table don't exist! \n");

int i;

for(i=0; i<buf.bytesused; i++)

{

if((buffers[numBufs].start[i] == 0x000000FF) && (buffers[numBufs].start[i+1] == 0x000000D8)) break;

ptcur++;

}

DEBUG("i=%d,FF=%02x,D8=%02x\n",i,buffers[numBufs].start[i],buffers[numBufs].start[i+1]);

int imagesize =buf.bytesused - i;

DEBUG("buf.bytesused = %d \n",buf.bytesused);

DEBUG ("imagesize = %d \n",imagesize);

9. 停止视频的采集。VIDIOC\_STREAMOFF

10. 关闭视频设备。close(fd);