[Video4linux 解析](http://blog.chinaunix.net/uid-25909619-id-3380662.html) 2012-10-21 21:48:39

分类：

原文地址：[Video4linux 解析](http://blog.chinaunix.net/uid-8786412-id-2454489.html) 作者：[wangchunyi](http://blog.chinaunix.net/uid/8786412.html)

•Video4linux（简称V4L),是linux中关于视频设备的内核驱动。

•现在已有Video4linux2，还未加入linux内核，使用需自己下载补丁

•在Linux中，视频设备是设备文件，可以像访问普通文件一样对其进行读写

•摄像头在/dev/video0下

1.打开视频设备：

2. 读取设备信息

3.更改设备当前设置（如果有必要）

4.进行视频采集，两种方法: (都没成功）L

                            1.内存映射

                            2.直接从设备读取

5.对采集的视频进行处理（没做）

6.关闭视频设备。

为程序定义的数据结构

•typedef struct v4l\_struct

•{

•   int fd;

•   struct video\_capability capability;

•   struct video\_channel channel[4];

•   struct video\_picture picture;

•   struct video\_window window;

•   struct video\_capture capture;

•   struct video\_buffer buffer;

•   struct video\_mmap mmap;

•   struct video\_mbuf mbuf;

•   unsigned char \*map;

•   int frame;

•   int framestat[2];

•}vd;

•1. **video\_capability**

•包含设备的基本信息（设备名称、支持的最大最小分辨率、信号源信息等）

•包含的分量：

•**name[32]   //**设备名称

•**maxwidth ，maxheight，minwidth，minheight**

•**Channels    //**信号源个数

•**type    //是否能capture，彩色还是黑白，是否          能裁剪等等。                                      值如VID\_TYPE\_CAPTURE等**

2 •video\_picture

•设备采集的图象的各种属性

•**brightness** 0~65535

•**hue**

•**colour**

•**contrast**

•**whiteness**

•**depth** // 24

•**palette** //**VIDEO\_PALETTE\_RGB24**

3•video\_channel         关于各个信号源的属性

**Channel //信号源的编号**

**name**

**tuners**

**Type     VIDEO\_TYPE\_TV  |  IDEO\_TYPE\_CAMERA**

**Norm  制式**

**4**

•**video\_window**  //包含关于capture area的信息

**x  x windows 中的坐标**.

**y     x windows 中的坐标**.

**width**    The width of the image capture.

**height**   The height of the image capture.

**chromakey** A host order RGB32 value for the chroma key.

**flags**      Additional capture flags.

**clips**      A list of clipping rectangles. *(Set only)*

**clipcount**    The number of clipping rectangles. *(Set only)*

5 •video\_mbuf

       利用mmap进行映射的帧的信息

**size**  //每帧大小

**Frames //最多支持的帧数**

**Offsets  //每帧相对基址的偏移**

6 •video\_buffer   最底层对buffer的描述

**void \*base**Base   physical address of the buffer

**int height**Height of the frame buffer

**int width**Width    of the frame buffer

**int depth**Depth   of the frame buffer

**int bytesperline**Number of bytes of memory between the start of two adjacent lines

    实际显示的部分一般比它描述的部分小

7 •video\_mmap //用于mmap

关键步骤介绍

•初始化阶段做的工作

•**int ioctl(int fd, ind cmd, …)   i**nput **o**utput **c**on**t**ro**l** 的缩写

•用于和设备进行“对话”。

•如果驱动程序提供了对ioctl的支 持，用户就可以在用户程序中使用ioctl函数控制设备的I/O通道。

•fd：设备的文件描述符，cmd:用户程序对设备的控制命令 ，.省略号一般是一个表示类型长度的参数，也可以没有。

•1.打开视频：

•Open（”/dev/video0”，vdàfd);

•关闭视频设备用close（”/dev/video0”，vdàfd);

•2. 读**video\_capability** 中信息

•ioctl(vd->fd, VIDIOCGCAP, &(vd->capability))

•成功后可读取vd->capability各分量  eg.

•Printf（”maxwidth = %d”vd->capability .maxwidth）;

•3.读video\_picture中信息

•ioctl(vd->fd, VIDIOCGPICT, &(vd->picture))；

•

•

•4.改变video\_picture中分量的值

•先为分量赋新值，再调用VIDIOCSPICT

•Eg.

•vd->picture.colour = 65535;

•if(ioctl(vd->fd, VIDIOCSPICT, &(vd->picture)) < 0)

•     {

•            perror("VIDIOCSPICT");

•            return -1;

•     }

••5.初始化channel

•必须先做得到vd->capability中的信息

•for (i = 0; i < vd->capability.channels; i++)

•   {

•      vd->channel[i].channel = i;

•      if (ioctl(vd->fd, VIDIOCGCHAN, &(vd->channel[i])) < 0)

•       {

•          perror("v4l\_get\_channel:");

•         return -1;

•         }

•   }

•截取图象的第一种方法：用mmap（内存映射）方式截取视频

•mmap( )系统调用使得进程之间通过映射同一个普通文件实现共享内存。普通文件被映射到进程地址空间后，进程可以向访问普通内存一样对文件进行访问，不必再调用read()，write（）等操作。

•两个不同进程A、B共享内存的意思是，同一块物理内存被映射到进程A、B各自的进程地址空间。进程A可以即时看到进程B对共享内存中数据的更新，反之亦然

•采用共享内存通信的一个显而易见的好处是效率高，因为进程可以直接读写内存，而不需要任何数据的拷贝

•1.设置picture的属性

•2. 初始化video\_mbuf，以得到所映射的buffer的信息

–ioctl(vd->fd, VIDIOCGMBUF, &(vd->mbuf))

••3.可以修改video\_mmap和帧状态的当前设置

•    Eg.     vd->mmap.format = VIDEO\_PALETTE\_RGB24

•              vd->framestat[0] = vd->framestat[1] = 0;  vd->frame = 0;

•

•4.将mmap与video\_mbuf绑定

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

•len //映射到调用进程地址空间的字节数，它从被映射文件开头   offset个字节开始算起

•Prot //指定共享内存的访问权限  PROT\_READ（可 读） ,    PROT\_WRITE （可写）, PROT\_EXEC        （可执行）

•flags  // MAP\_SHARED   MAP\_PRIVATE中必选一个

•         // MAP\_ FIXED不推荐使用

•

••addr  //共内存享的起始地址，一般设0，表示由系统分配

•Mmap( )  返回值是系统实际分配的起始地址

•if((vd->map = (unsigned char\*)mmap(0, vd->mbuf.size, PROT\_READ|PROT\_WRITE, MAP\_SHARED, vd->fd, 0)) < 0)

•     {

•            perror("v4l\_mmap mmap:");

•            return -1;

•     }

•

•4.Mmap方式下真正做视频截取的 VIDIOCMCAPTURE

•ioctl(vd->fd, VIDIOCMCAPTURE, &(vd->mmap)) ;

•若调用成功，开始一帧的截取，是非阻塞的，

•是否截取完毕留给VIDIOCSYNC来判断

••5. 调用VIDIOCSYNC等待一帧截取结束

•if(ioctl(vd->fd, VIDIOCSYNC, &frame) < 0)

•     {

•            perror("v4l\_sync:VIDIOCSYNC");

•            return -1;

•     }

•若成功，表明一帧截取已完成。可以开始做下一次 VIDIOCMCAPTURE

•frame是当前截取的帧的序号。

••关于双缓冲：

•video\_bmuf  bmuf.frames = 2;

•一帧被处理时可以采集另一帧

•   int frame; //当前采集的是哪一帧

•   int framestat[2]; //帧的状态  没开始采集|等待                         采集结束

•帧的地址由

•vd->map + vd->mbuf.offsets[vd->frame]得到

•

•采集工作结束后调用munmap取消绑定

•munmap(vd->map, vd->mbuf.size)

•关于mmap过程的总结：

•1.得到图象的信息

•2.初始化video\_mbuf                      VIDIOCGMBUF

•3. video\_mbuf与mmap绑定           mmap（）

•4. 可以修改video\_mmap和帧状态的当前设置

•    Eg.     vd->mmap.format = VIDEO\_PALETTE\_RGB24

•              vd->framestat[0] = vd->framestat[1] = 0;  vd->frame = 0;

•Loop：

•{

•     if（framestat[帧号] = 0）

     VIDIOCMCAPTURE( );  // 开始 截取，置framestat[帧号] = 1

     if（framestat[帧号] = 1）

         VIDIOCSYNC( );  //等待截取完成，置framestat[帧号] = 0

       对采集的帧做处理；//帧地址是vd->map + vd->mbuf.offsets[vd->frame]

      帧号 = 帧号 ^ 1;

}