<http://www.linuxidc.com/Linux/2011-03/33022.htm>

**Linux下V4L2程式設計小結**

**一．設置採集方式，格式 常用命令標誌**

    打開視頻設備後，可以設置該視頻設備的屬性，例如裁剪、縮放等。這一步是可選的。在Linux程式設計中，一般使用ioctl函數來對設備的I/O通道進行管理：

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

\_\_fd：設備的ID，例如剛才用open函數打開視頻通道後返回的cameraFd；

\_\_request：具體的命令標誌符。

在進行V4L2開發中，一般會用到以下的命令標誌符：

1. VIDIOC\_REQBUFS：分配記憶體
2. VIDIOC\_QUERYBUF：把VIDIOC\_REQBUFS中分配的資料緩存轉換成物理位址
3. VIDIOC\_QUERYCAP：查詢驅動功能
4. VIDIOC\_ENUM\_FMT：獲取當前驅動支援的視頻格式
5. VIDIOC\_S\_FMT：設置當前驅動的頻捕獲格式
6. VIDIOC\_G\_FMT：讀取當前驅動的頻捕獲格式
7. VIDIOC\_TRY\_FMT：驗證當前驅動的顯示格式
8. VIDIOC\_CROPCAP：查詢驅動的修剪能力
9. VIDIOC\_S\_CROP：設置視訊訊號的邊框
10. VIDIOC\_G\_CROP：讀取視訊訊號的邊框
11. VIDIOC\_QBUF：把數據從緩存中讀取出來
12. VIDIOC\_DQBUF：從視頻緩衝區的輸出隊列中取得一個已經保存有一幀視頻數據的視頻緩衝區；
13. VIDIOC\_STREAMON：開始視頻顯示函數
14. VIDIOC\_STREAMOFF：結束視頻顯示函數
15. VIDIOC\_QUERYSTD：檢查當前視頻設備支援的標準，例如PAL或NTSC。

這些IO調用，有些是必須的，有些是可選擇的。

**二.一般操作流程（視頻設備）**

1. 打開設備檔。

int fd=open(”/dev/video0″,O\_RDWR);  
2. 取得設備的capability，看看設備有什麼功能，比如是否具有視頻或者音訊輸入輸出等。(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. 選擇視頻輸入，一個視頻設備可以有多個視頻輸入(可不要)。(VIDIOC\_S\_INPUT, struct v4l2\_input)。

4. 設置視頻的制式和框架格式，制式包括PAL，NTSC，幀的格式個包括寬度和高度等。(VIDIOC\_S\_STD,VIDIOC\_S\_FMT, struct v4l2\_std\_id, struct v4l2\_format)。

   struct v4l2\_format    fmt;

  v4l2\_format 結構如下：

  struct v4l2\_format {

    enum v4l2\_buf\_type type;    // 資料流程類型，必須永遠是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;         // 寬，必須是16 的倍數

    \_\_u32                   height;        // 高，必須是16 的倍數

    \_\_u32                   pixelformat;   // 視頻資料存儲類型，例如是YUV 4 ：2 ：2 還是RGB

    enum v4l2\_field         field;

    \_\_u32                   bytesperline;

    \_\_u32                   sizeimage;

    enum v4l2\_colorspace    colorspace;

    \_\_u32                   priv;

  };

樣例：

  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）{

    //return 0;

  }

注意：如果該視頻設備驅動不支持你所設定的圖像格式，視頻驅動會重新修改struct v4l2\_format結構體變量的值為該視頻設備所支持的圖像格式，所以在程序設計中，設定完所有的視頻格式後，要獲取實際的視頻格式，要重新讀取struct v4l2\_format結構體變量。  
5. 向驅動申請幀緩衝，一般不超過5個。(struct v4l2\_requestbuffers)。

struct v4l2\_requestbuffers {

\_\_u32 count; // 緩存數量，也就是說在緩存佇列裡保持多少張照片

enum v4l2\_buf\_type type;   // 資料流程類型，必須永遠是V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE

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

\_\_u32 reserved[2];

};

樣例：

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) {

    return -1;

  }

注意：VIDIOC\_REQBUFS請求V4L2驅動分配視頻緩衝區(申請V4L2視頻驅動分配內存)，V4L2是視頻設備的驅動層，位於內核空間，所以通過VIDIOC\_REQBUFS控制命令字申請的內存位於內核空間，應用程序不能直接訪問，需要通過調用mmap內存映射函數把內核空間內存映射到用戶空間後，應用程序通過訪問用戶空間地址來訪問內核空間。

注意：VIDIOC\_REQBUFS會修改tV4L2\_reqbuf的count值，tV4L2\_reqbuf的count值返回實際申請成功的視頻緩衝區數目；

6. 申請實體記憶體，並將申請到的幀緩衝映射到用戶空間，可直接操作採集到幀，而不必去複製。將申請到的幀緩衝全部入佇列，以便存放採集到的資料。VIDIOC\_QBUF將緩衝重新入佇列尾，這樣可以迴圈採集。(VIDIOC\_QBUF, struct v4l2\_buffer)。

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

    }

    //記憶體映射

   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);

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

     //return -1;

   }

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

     //return -1;

   }

}

控制命令VIDIOC\_QUERYBUF的功能是查詢已經分配的V4L2的視頻緩衝區的相關信息，包括視頻緩衝區的使用狀態、在內核空間的偏移地址、緩衝區長度等。在應用程序設計中通過調VIDIOC\_QUERYBUF來獲取內核空間的視頻緩衝區信息，然後調用函數mmap把內核空間地址映射到用戶空間，這樣應用程序才能夠訪問位於內核空間的視頻緩衝區。

控制命令VIDIOC\_QBUF是投放一個空的視頻緩衝區到視頻緩衝區輸入隊列中；函數執行成功後，指令(指定)的視頻緩衝區進入視頻輸入隊列，在啟動視頻設備拍攝圖像時，相應的視頻數據被保存到視頻輸入隊列相應的視頻緩衝區中。

7. 開始視頻的採集。VIDIOC\_STREAMON

 enum v4l2\_buf\_type type;

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

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

     // return -1;

     }

8. 出佇列以取得已採集資料的幀緩衝，取得原始採集資料。

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

        //return -1;

      }

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

      buf.memory = V4L2\_MEMORY\_MMAP;

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

      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);

函數執行成功後，相應的內核視頻緩衝區中保存有當前拍攝到的視頻數據，應用程序可以通過訪問用戶空間來讀取該視頻數據。（前面已經通過調用函數mmap做了用戶空間和內核空間的內存映射）。

VIDIOC\_DQBUF命令結果使從隊列刪除的緩衝幀信息傳給了此tVL2buf。V4L2\_buffer結構體的作用就相當於申請的緩衝幀的代理，找緩衝幀的都要先問問它，通過它來聯繫緩衝幀，起了中間橋樑的作用。

9. 停止視頻的採集。VIDIOC\_STREAMOFF

10. 關閉視頻設備。close(fd);