Linux 下基于 Video4Linux 的 USB 摄像头视频采集实现

黄睿邦 ， 汤荣江 ， 李文亮

（广东工业大学计算机学院，广州 510006）

摘 要：介绍在 Linux 系统下通过 USB 摄像头实现视频采集的一种方法，利用 Video4Linux 提供的数据结构、编程接口，结合 USB 摄像头驱动，实现了 Linux 环境下 USB 摄像头的视频采集、连续播放的功能。

关键词：Video4Linux；摄像头；USB；视频采集

1 Video4Linux 简介

Video4Linux(简 V4L)是 Linux 中关于视频设备的内核驱动，它为针对视频设备的应用程序编程提供一系列接口函数，这些视频设备包括现今市场上流行 TV 卡、视频捕捉卡和 USB 摄像头等。

对于 USB 口摄像头，其驱动程序中需要提供基本的 I/O 操作接口函数 open、read、write、close 的实现。对中断的处理实现，内存映射功能以及对 I/O 通道的控制接口函数 ioct1 的实现等，并把它们定义在 struct file_operations 中。这样当应用程序对设备文件进行诸如 open、close、read、write 等系统调用操作时，Linux 内核将通过 file_operations 结构访问驱动程序提供的函数。例如，当应用程序对设备文件执行读操作时，内核将调用 file_operations 结构中的 read 函数。

1 加载 USB 摄像头驱动模块

在 Linux 下，所有的外设均被看成是一种特殊文件进行处理，称之为设备文件。系统调用及各种函数库直接或间接地提供了内核和应用程序之间的接口，而设备驱动程序则是内核和外设之间的接口。

驱动程序在 Linux 内核里扮演着特殊的角色，它们是截然不同的“黑盒子”，使硬件的特殊的一部分响应定义好的内部编程接口。它们完全隐藏了设备工作的细节。用户的活动通过一套标准化的调用来进行，这些调用与特别的驱动是独立的；设备驱动的角色就是将这些调用映射到作用于实际硬件的和设备相关的操作上。这个编程接口是这样，驱动可以与内核的其他部分分开建立，并在需要的时候在运行时“插入”。

视频设备也是一种设备文件，所以可以像访问普通文件一样对其进行读写。摄像头设备地址是/dev/video0。

在系统平台上对 USB 口摄像头进行驱动，首先把 USB 控制器驱动模块静态编译进内核，使平台中支持 USB 接口，再在需要使用摄像头采集时，使用 insmode 动态加载其驱动模块，这样摄像头就可正常工作了。

在 Linux 下，ov511 芯片是直接支持 Linux，这种芯片的摄像头有网眼 V2000。笔者使用的是 Z-Star ZC0301 芯片的摄像头。需要 spca5xx 的驱动，这是一个通用驱动。笔者下载的是 gspcav1，比 spca5xx 更高一点的版本。

用户可以在命令行输入：lsusb ； 查看所有 USB 接口的设备参数，接上 USB 摄像头后，这里就会显示相关信息，摄像头设备的参数信息最后字母为 WebCam，参数包括生产商、芯片型号及产品型号。安装驱动模块具体步骤如下：

(1)下载 gspcav1-20070110.tar.gz 包，解压。在终端里进入该文件夹运行：make； 得到 gspca.ko 驱动模块。

(2)可以通过文件自带的程序，用：make install；加载驱动模块；或通过命令：modprobe gspca；加载驱

收稿日期：2009-04-07 修稿日期：2009-06-11

作者简介：黄睿邦(1984-)，男，广东东莞人，在读研究生，研究方向为网络监控与决策

动模块。

加载模块的命令有 insmod 与 modprobe，其差别于 modprobe 能够处理 module 载入的相依问题。比方要载入 a module，但是 a module 要求系统先载入 b module 时，直接用 insmod 挂入通常都会出现错误信息，而 modprobe 能够知道先载入 b module 后才载入 a module，如此相依性就会满足。所以推荐大家使用 modprobe 命令。

加载模块完毕后。输入：lsmodl grep gdpca；就可以看到有 gspca 模块。再输入：ls /dev/video0；就可以检测到设备。这样就成功加载了驱动模块。

2 程序模块设计

视频图像采集流程图如图 1：

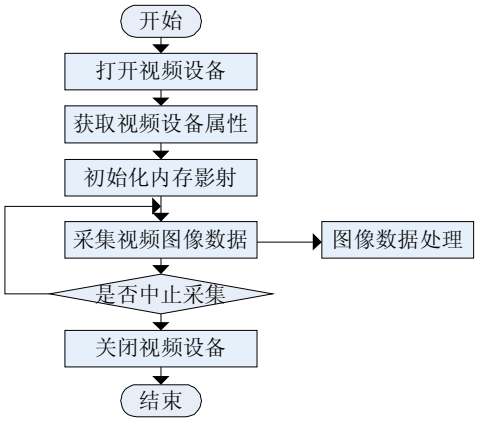

图1 视频图像采集流程

在 Linux 系统下视频设备的图像数据采集是基于 Video4Linux 应用编程接口的，通过使用接口提供的有个数据结构定义和函数库，对摄像头进行图像采集。

这里先定义一个描述设备的数据结构，它包含了 v4l 中定义的所有数据结构，在本文后续部分用到的变量也以此为依据。

```
struct _v4l_struct
{ int fd;
  struct video_capability capability;
  struct video_channel channel[8];
  struct video_picture picture;
   struct video_mmap mmap;
   struct video_mbuf mbuf;
   unsigned char *map;
   unsigned char *buffer;
   int frame_current;
   int frame_using[2];
};
typedef struct _v4l_struct v4l_device;
v4l_device  vd;
```

这些数据结构的用途如下：

●video_capability 包含摄像头的基本信息，例如设备名称、支持的最大最小分辨率、信号源信息；

●video_picture 包含设备采集图象的各种属性：亮度、色调、对比度、色度、深度、调色版；

●video_channel 包含信号通道的各种属性，例如信号源的编、名字、类型等；

●video_mmap 用于内存映射；

●video_mbuf 利用 mmap 进行映射的帧信息，包括帧大小、最多支持帧数、每帧相对基址的偏移；

●Map 是 mmap 方式获取数据时，数据的首地址；

●Buffer 是图像数据存放区，即帧地址；

●frame_current 是保存当前使用的是那一帧；

●frame_using[2]是保存这一帧的状态：0 表示可期用，1 表示不可用。

ioctl 函数大部分驱动除了读写设备的能力，还需要通过设备驱动进行各种硬件控制的能力。大部分设备可进行超出简单的数据传输之外的操作。而 ioctl 函数就是设备驱动程序中对设备的 I/O 通道进行管理的函数。所谓对 I/O 通道进行管理，就是对设备的一些特性进行控制，例如串口的传输波特率、马达的转速或者自我销毁等。函数原型：int ioctl (int fd, ind cmd, …)；其中 fd 就是用户程序打开设备时使用 open 函数返回的文件标示符，cmd 就是用户程序对设备的控制命令，至于后面的省略号，那是一些补充参数，一般最多一个，有或没有是和 cmd 的意义相关的。

ioctl 函数是文件结构中的一个属性分量，就是说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。还有，用户程序所作的只是通过命令码告诉驱动程序它想做什么，至于怎么解释这些命令和怎么实现这些命令，这都是驱动程序要做的事情。

2.1 打开视频设备文件

```
if( (vd->fd = open( '/dev/video', O_RDWR ) ) < 0 )
{   perror( "v4l_open error" );
      return -1;
}
```

用 Open 函数打开视频设备，可以获取相应的文

件描述符.若打开失败。就返回错误信息。在实际的程序设计中,加入类似的出错处理,可以方便程序调试。

2.2 获取视频设备属性

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

fd 就是用户程序打开设备时使用 Open 函数返回的文件标示符,VIDIOCGCAP 参数告诉函数取得视频设备文件的性能参数,包括名称、类型、通道数、图像宽度、图像高度等,并存放到 video_capability 的结构里。

2.3 获取图像属性信息

```
ioctl( vd->fd,VIDIOCGPICT,&( vd->picture )
```

VIDIOCGPICT 参数告诉 ioctl 函数返回采集图像帧的属性,包括图像亮度、色彩、对比度等,比较重要的是图像帧的调色板参数 palette,并存放到数据结构 videoqoieture 里。

2.4 内存映射初始化

设置 video_mmap 类变量的值:

```
vd->mmap.width = MAX_WIDTH;
vd->mmap.height = MAX_HEIGHT;
vd->mmap.format = vd->picture.palette;
```

通过对 mmap 结构填人适当的值,初始化设备的图像采集格式、大小等。如设置图像格式为 RGB32、大小为 640×480 或者 320×240。

映射设备文件到内存

截取图像有两种方法:直接读取设备方式和 mmap 内存映射方式。直接读取设备方式通过内核缓冲区来读取数据;而 mmap()通过把设备文件映射到内存中,绕过了内核缓冲区,最快的磁盘访问往往还是慢于最慢的内存访问,所以 mmap()方式加速了 I/O 访问。这里本程序采用前一种方法。另外,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,进程可以像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。

```
ioctl ( vd->fd,VIDIOCGMBUF,&( vd->mbuf )
vd ->map =(unsignedchar*)mmap (0,vd ->mbuf.size,
PROT_READ|PROT_WRITE,MAP_SHARED, vd->fd, 0 )
```

函数原型 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。

2.5 图像采集

图像的采集主要是调用了两次 ioctl()函数,用到的命令参数是 VIDIOCMCAPTURE 和 VIDIOCSYNC。

```
ioctl(vd->fd, VIDIOCMCAPTURE, &(vd->mmap )
```

VIDIOCMCAPTURE 参数告诉 ioctl(),将图像数据采集到 mmap()所映射的内存中。若调用成功,就开始一帧的截取, 是非阻塞的, 是否截取完一帧留给 VIDIOCSYNC 来判断。

```
ioctl(vd->fd, VIDIOCSYNC, &(vd->frame_current)
```

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

单帧采集和连续帧采集:

●单帧采集:摄像头存储缓冲区帧信息中,最多可支持的帧数一般为两帧。对于单帧采集只需设置 vd->mmap.frame=0, 即采集其中的第一帧,然后调用 ioctl(vd->fd, VID10CMCAPTuR E, &(vd->mmap))函数, 再调用 ioctl (vd->fd, VIDIOCSYNC, & (vd->frame_current)) 之后就可把图像数据保存成文件的形式。

●连续帧采集:在单帧的基础上,利用 vd->mbuf.frames 值确定采集完摄像头帧缓冲区帧数据进行循环的次数。在循环语句中, 也是使用 VIDIOCMCCAPTURE 和 VIDIOCSYNC ioctl 函数完成每帧截取,但要给采集到的每帧图像赋地址,利用语句:

vd ->buffer =vd ->map +vd ->mbuf.offsets [vd ->frame_current],然后保存成文件的形式。

部分代码如下:

```
vd->mmap.frame = vd->frame_current;
ioctl(vd->fd, VIDIOCMCAPTURE, &(vd->mmap )
vd->frame_using[vd->frame_current] = 1;
ioctl(vd->fd, VIDIOCSYNC, &(vd->frame_current)
vd->frame_using[vd->frame_current] = 0;
vd->buffer=vd->map+vd->mbuf.offsets [vd->frame_cur-
```

```
rent];  //得到这一帧的地址
    vd->frame_current = (v4l_dev.frame_current+1)%2; //下一帧的 frame
```

本程序通过设置一个定时器：timerid = startTimer (50)；在定时器事件里不断循环以包括以上代码的子程序，来连续采集图像数据。

2.6 图像显示

通过读取保存在内存中的图像数据，并把它填充到空图像中，以显示一帧的图像在客户端上。部分程序如下，这部分程序代码也是放在定时器的事件中，以达到连续播放，实时监控。

```
if(img.create(MAX_WIDTH, MAX_HEIGHT, 32, 0, QImage::IgnoreEndian))
  { for(y=0; y<MAX_HEIGHT; y++)
    { for(x=0; x<MAX_WIDTH; x++)
      { r=(int)bit[i+2];
        g=(int)bit[i+1];
        b=(int)bit[i];
        point= (QRgb *)(img).scanLine(y)+ x;
        *point = qRgb(r,g,b);
        i+=3;
        } } }
paint.drawImage(5, 5, (img));
```

3 运行结果

操作系统是：Fadora 8，内核是：2.6.23.1。运行程序后，可实时显示 USB 摄像头的画面。如图 2、图 3：

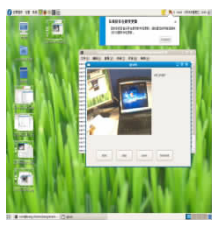

图 2 运行结果一

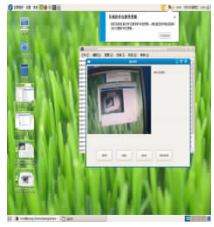

图 3 运行结果二

4 结 语

Linux 作为一种新的操作系统，其发展前景是无法估量的，本程序是在 Linux 操作环境下实现了 USB 摄像头图像的采集，并连续播放功能。主要涉及的有 Video4Linux 应用编程和 QT 库。本程序结构、功能都比较简单，在此基础上可以扩展到一个实时监控系统，又或者是在此基础上进行一些图像处理分析方面的研究。

参考文献

[1]肖踞雄，翁铁成，宋中庆. USB 技术及应用设计[M]. 北京：清华大学出版社，2003

[2]魏永明. Rubini A. Linux 设备驱动程序[M]. 北京：中国电力出版社，2002

[3]Robert Love. Linux 的内核开发[M]. USA：O Reilly，2004

[4]Neil Matthew，Richard Stones，陈健等. Linux 程序设计[M]. 3 版. 北京：人民邮电出版社，2007

Implementation of USB WebCam Display in Linux Based on Video4Linux

HUANG Rui-bang , TANG Rong-jian , LI Wen-liang

(School of Computer, Guangdong University of Technology, Guangzhou 510006)

Abstract: Introduces a method of video capture through USB WebCam based on Linux system, uses the Video4Linux's data structure and programming interface with USB WebCam driver to capture the USB WebCam's image date and display it in Linux environment.

Keywords: Video4Linux; WebCam; USB; Image Acquisition