基于 Video4Linux 的 USB 摄像头图像采集实现

Write by daily3（戴小鼠） 著作权：戴丽（合肥工业大学）
(email:daily3@126.com)

做了一段时间的摄像头图像采集，有了一些心得。在论坛上开的2410摄像头问题专贴（http://www.hhcn.com/cgi-bin/topic.cgi?forum=1&topic=247&show=0）也得到了大家的关注。在此，我将这一阶段遇到的问题，解决方法等做个总结，希望对您有所帮助。

Linux本身自带了采用ov511芯片的摄像头，而市场上应用最广泛的是采用中芯微公司生产的zc301芯片的摄像头，下面我将针对这两大系列的摄像头分别做介绍。（注：所有的开发都是在华恒HHARM-2410-EDU上完成，ov511摄像头采用的是网眼webeye3000，zc301摄像头采用的是ANC奥尼S888）。

一 驱动加载

1.1 ov511 驱动

1.静态加载

（1）在arm linux的kernel目录下make menuconfig。

（2）首先(*)选择Multimedia device->下的Video for linux。加载video4linux模块，为视频采集设备提供了编程接口；

（3）然后在usb support->目录下(*)选择support for usb和usb camera ov511 support。这使得在内核中加入了对采用OV511接口芯片的USB数字摄像头的驱动支持。

（4）保存配置退出。

（5）make dep；make zImage

此时在/tftpboot 下就生成了带有 ov511 驱动的内核。

2.动态加载

（1）在arm linux的kernel目录下make menuconfig。

（2）首先<*>选择Multimedia device->下的Video for linux。

（3）然后在usb support->目录下<*>选择support for usb和<M>选择usb camera

ov511 support。

（4）保存退出。

（5）Make dep;make zImage;make modules然后就在/driver/usb下生成ov511.o，同时生成的zImage自动放在/tftpboot下。

（6）然后用新内核启动板子后insmod ov511.o就可以成功加载。

动态方式与静态方式相比，测试时要简单的多。不需要下载整个内核，只需通过nfs，加载驱动即可测试。在测试成功后就可以编译进内核。

模块加载中出现的问题：

1．insmod 和 modprobe 间的一个区别试后者不会在当前目录中查找模块，它只在/lib/modules 下的缺省目录下查找，这是因为该程序只是一个系统实用例程，不是一个交互工具。可以通过在/etc/modules.conf 中指定自己的目录，来把它们加到缺省目录集中。

2．如果插入模块 ov511.o 时，出现以下信息：

Ov511.o:unresolved symbol video********之类的，说明还有其它模块 videodev.o 没有加。

3．出现错误：ov511.o:couldn’t find the kernel version this modules was compiled for。这是试图插入一个不是可装入模块的目标文件。因为在内核配置阶段，是把 ov511 模块静态加到内核中的，虽然看起来和可装入模块的文件名 ov511.o 完全一样，但是不能用 insmod 命令加入。

4．如果出现 Ov511.o:unresolved symbol video********，那就<M>选中 video for linux,用新生成的内核启动系统，再 insmod videodev.o,insmod ov511.o 就可以啦。

1.2 zc301 驱动

摄像头的驱动是从http://mxhaard.free.fr/下的针对embeded环境，有专门的patch，我用的是usb-2.4.31LE06.patch。

（1）把它放到/HHARM9-EDU/kernel/driver/usb下，解压，打补丁。就会在此目录下看到spca5xx文件夹了。可能会有一些错误，我的错误是在Makefile和config.in文件中，根据它的提示，进行相应的修改即可。Patch时会将修改方法写到Makefile.rej和config.in.rej文件中，把这两个文件里的内容加到Makefile和config.in中就行了。

（2）编译内核，进入/HHARM9-EDU/kernel，make menuconfig。我采用和上面介绍的ov511驱动的方法一样，动态加载。（M）选中SPCA5XX这一项。

（3）make dep；make zImage；make modules。就会在/HHARM9-EDU/kernel/driver/usb/spca5xx 中生成spca5xx.o,spcadecoder.o,spca_core.o啦。这就是我们要的驱动。
（4）用新内核启动，insmod这三个.o文件（可以不用加载spcadecoder.o），摄像头就加载成功啦。

不过这种LE的驱动有许多问题，比如运行到设置图像格式(RGB565或RGB24)时出错，说不支持此参数。原因在于：(摘自驱动程序主页http://mxhaard.free.fr/spca5le.html)
The spca5xx-LE design is very different from the spca5xx full package(LE版的驱动和完全版的差很多)。
The memory in use are the most smaller as possible(LE版的驱动会尽量减少内存的使用)
The spcadecoder is reduce and only raw jpeg webcam are used.(驱动模块只支持输出原始jpeg格式)。

还有一种方法，从 http://mxhaard.free.fr/download.html 下载最新的驱动 spca5xx-20060402.tar.gz。这个可独立编译，无需放到linux内核里面，编译生成一个spca5xx.o即可，不要三个.o做驱动了。因为这个驱动是针对2.6的，编译时会出现很多错误，修改CFLAGS即可。华恒的群里已经有编译好的驱动提供大家下载。

模块加载中出现的问题：

1．运行./servfox时出现Error Opening V4L interface.
我测试一下，是没有加载驱动。虽然内核中（M）选中了驱动，但是启动后要手工加进去。insmod一下啦。

2．insmod spcadecoder.o时，出现错误：spcadecoder.o:couldn’t find the kernel version this modules was compiled for。如果你insmod spca5xx.o成功的话就不需要再insmod其他模块了。
3．insmod video.o时却说can't find the kernel version the modules was compiled for。这是因为video for linux一般是直接编译到内核中去的.不需要加载的。

二 Video4linux 编程

2.1 Video4linux 简介

Video4Linux是为市场现在常见的电视捕获卡和并口及USB口的摄像头提供统一的编程接口。同时也提供无线电通信和文字电视广播解码和垂直消隐的数据接口。本文主要针对USB摄像头设备文件/dev/video0，进行视频图像采集方面的

程序设计。

2.2 Video4linux 编程指南

1.视频编程的流程

（1）打开视频设备：

（2）读取设备信息

（3）更改设备当前设置（可以不做）

（4）进行视频采集，两种方法:

a.内存映射

b.直接从设备读取

（5）对采集的视频进行处理

（6）关闭视频设备。

定义的数据结构及使用函数

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

extern int v4l_open(char *, v4l_device *);
extern int v4l_close(v4l_device *);
extern int v4l_get_capability(v4l_device *);
extern int v4l_set_norm(v4l_device *, int);
extern int v4l_get_picture(v4l_device *);
extern int v4l_grab_init(v4l_device *, int, int);
extern int v4l_grab_frame(v4l_device *, int);
extern int v4l_grab_sync(v4l_device *);
extern int v4l_mmap_init(v4l_device *);
```

```
extern int v4l_get_mbuf(v4l_device *);
extern int v4l_get_picture(v4l_device *);
extern int v4l_grab_picture(v4l_device *, unsigned int);
extern int v4l_set_buffer(v4l_device *);
extern int v4l_get_buffer(v4l_device *);
extern int v4l_switch_channel(v4l_device *, int);
```

3.Video4linux支持的数据结构及其用途

（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）

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 8 16 24 32
palette VIDEO_PALETTE_RGB24 | VIDEO_PALETTE_RGB565| VIDEO_PALETTE_JPEG| VIDEO_PALETTE_RGB32

（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 PAL|NSTC|SECAM|AUTO

（4）video_window 包含关于capture area的信息

x x windows 中的坐标.
y y 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_mmap 用于mmap

4.关键步骤介绍

【注】接多个摄像头。方法如下：买一个usb hub接到开发板的usb host上。cat /proc/devices可以知道video capture device的major是81，再ls －l /dev看到video0的次设备号是0。两个摄像头当然要两个设备号，所以mknod /dev/video1 c 81 1，如果接4个，就mknod /dev/video2 c 81 2,mknod /dev/video3 c 81 3。依次类推。

（1）打开视频：

```
int v4l_open(char *dev, v4l_device *vd)
{
    if (!dev)
        dev = ”/dev/video0”;
     if ((vd ->fd = open(dev, O_RDWR)) < 0) {
        perror("v4l_open:");
        return   -1;
    }
    if (v4l_get_capability(vd))
        return   -1;
     if (v4l_get_picture(vd))
        retu rn -1;
     return 0;
}
```

（2）读video_capability 中信息

```
int v4l_get_capability(v4l_device *vd)
{
    if (ioctl(vd ->fd, VIDIOCGCAP, &(vd->capability)) < 0) {
        perror("v4l_get_capability:");
        return   -1;
    }
    return 0;
}
```

成功后可读取vd->capability各分量

（3）读video_picture中信息

```
int v4l_get_picture(v4l_device *vd)
{
    if (ioctl(vd ->fd, VIDIOCGPICT, &(vd->picture)) < 0) {
        perror("v4l_get_picture:");
        return   -1;
    }
    return 0;
}
```

成功后可读取图像的属性

（4）改变video_picture中分量的值 （可以不做的）

先为分量赋新值，再调用VIDIOCSPICT

```
vd->picture.colour = 65535;
if(ioctl(vd->fd, VIDIOCSPICT, &(vd->picture)) < 0)
     {
perror("VIDIOCSPICT");
return -1;
     }
```

（5）初始化channel （可以不做的）

必须先做得到vd->capability中的信息

```
int v4l_get_channels(v4l_device *vd)
{
    int i;
    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;
        }
    }
    return 0;
}
```

（6）关闭设备

```
int v4l_close(v4l_device *vd)
{
    close(vd ->fd);
    return 0;
}
```

重点：截取图象的两种方法

一、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

（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

```
int v4l_mmap_init(v4l_device *vd)
{
    if (v4l_get_mbuf(vd) < 0)
    return    -1;
    if ((vd ->map = mmap(0, vd->mbuf.size, PROT_READ|PROT_WRITE,
MAP_SHARED, vd->fd, 0)) < 0) {
        perror("v4 l_mmap_init:mmap");
        return    -1;
    }
    return 0;
}
```

（5）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

（6）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)

在实际应用时还可以采用缓冲队列等方式。

二、视频截取的第二种方法：直接读设备

关于缓冲大小，图象等的属性须由使用者事先设置

调用read（）；

int read (要访问的文件描述符；指向要读写的信息的指针；应该读写的字符数);

返回值为实际读写的字符数

```
int len ;
unsigned char
*vd->map=
  (unsigned char *) malloc(vd◊capability.maxwidth*vd◊capability.maxheight );
len = read(vd◊fd,vd◊ vd->map,
                        vd◊capability.maxwidth*vd◊capability.maxheight*3 );
```

2.3 编程实例（mouse_capture）

不管是ov511还是zc301的摄像头，它们采集的方式都是相同的，只不过采集到的数据有所差异，ov511的就是rgb的位流，而zc301是jpeg编码的位流。mouse_capture是根据servfox改编的一个专门从zc301摄像头获取一张jpeg图片，用来测试摄像头是否加载成功的小程序。这样就可以不用cat /dev/video0>1.jpg来测试摄像头是否正常。cat命令一运行，就源源不断地采集jpeg流。但是采到的图片只能显示第一个jpeg头和jpeg尾之间的数据。mouse_capture仅仅获得一张完整的jpeg。 可以从

（http://www.hhcn.com/cgi-bin/topic.cgi?forum=1&topic=247&start=144&show=0）处下载参考。

现将主要函数的功能介绍如下：

static int GetVideoPict (struct vdIn *vd);//获取图片属性信息。
static int SetVideoPict (struct vdIn *vd);//设置图片属性。
static int isSpcaChip (const char *BridgeName);//测试芯片类型
static int GetStreamId (const char *BridgeName); //测试输出数据的格式
static int GetDepth (int format);//获取颜色深度。
void exit_fatal(char *messages);//错误显示。
int init_videoIn(struct vdIn *vd,char *device,int width,int height,int format,int grabmethod);//初始化设备。
int convertframe(unsigned char *dst,unsigned char *src, int width,int height, int formatIn, int size);//把共享缓冲区中的数据放到一个变量中，通知系统已获得一帧。
int v4lGrab (struct vdIn *vd,char *filename);//从摄像头采集图片。
int close_v4l (struct vdIn *vd);//关闭摄像头。
int get_jpegsize (unsigned char *buf, int insize);//获取jpeg图片大小。

三 实例程序

3.1 LCD 实时显示从 ov511 上采集的图像

参考HHARM9-EDU/applications/usbcam2lcd。从摄像头获取bmp位流直接显示在framebuffer中。此程序图像的采集采用read的方式，注意由于lcd液晶屏显示的是16bits的RGB图片，所以，ov511输出的图片格式也应该是16bits的RGB图片数据，宏VIDEO_PALETTE_RGB565定义的就是16bits的RGB数据图片。而linux自带的ov511驱动中图像采集是32位的，这样采集到的图片显示在lcd上是雪花点。因此需要修改驱动。 在kernet/driver/usb/目录下有ov511芯片的驱动ov511.c，驱

动里的ov51x_set_default_params函数是设置芯片默认的输出图片的格式，该函数中的

```
for (i = 0; i < OV511_NUMFRAMES; i++)
{
ov511->frame[i].width = ov511->maxwidth;
ov511->frame[i].height = ov511->maxheight;
ov511->frame[i].bytes_read = 0;
if (force_palette)
ov511->frame[i].format = force_palette;
else
ov511->frame[i].format = VIDEO_PALETTE_RGB24;
ov511->frame[i].depth = ov511_get_depth(ov511->frame[i].format);
}
```

部分语句是主要设置ov511默认输出图片格式的，其中maxwidth和maxheight设置了图片的最大的宽度和高度。Ifelse语句设置了图片的格式，作如下的修改：

```
for (i = 0; i < OV511_NUMFRAMES; i++)
{
ov511->frame[i].width = ov511->maxwidth;
ov511->frame[i].height = ov511->maxheight;
ov511->frame[i].bytes_read = 0;
ov511->frame[i].format = VIDEO_PALETTE_RGB565;
ov511->frame[i].depth = ov511_get_depth(ov511->frame[i].format);
}
```

如果需要，也可以改变图片的默认输出大小。

3.2 LCD 实时显示从 zc301 上采集的图像

编程思想：从摄像头采集到的图片存放在本地文件夹，通过minigui加载jpeg来实现显示。

具体过程：

1．从网上下载jpegsrc-6b的jpeg库，交叉编译。

（1）./configure –enable-static –enable-shared –prefix=.libs

（2）修改Makefile，将编译器改成交叉编译器。

例如：我改成/opt/host/armv4l/bin/armv4l-unknown-linux-gcc

（3）make 后即在.libs目录中生成for arm的

libjpeg.a, libjpeg.la,libjpeg.so,libjpeg.so.62,libjpeg.so.62.0.0。将这些文件拷贝到系

统库文件目录，我的是/usr/lib中。

2．因为看从zc301采集的图片的二进制位流，jpeg头是ff d8 ff db。而在minigui库文件libminigui的源文件src/mybmp/jpeg.c中，load_jpg和check_jpg的时候测试的头位EXIF和JFIF两种格式的jpeg图片。这两种对应的二进制分别是ff d8 ff e1和ff d8 ff e0。所以我们minigui通过判断认为这是错误的jpeg格式而不加载，故无法显示。实际上通过测试，在源码中去掉这两个判断就能正确加载。

3.交叉编译minigui

（1）编译库：./configure --host=arm-unknown-linux --enable-jpgsupport=yes --enable-pngsupport=no --enable-gifsupport=no --disable-lite --prefix=/HHARM9-EDU/applications/minigui-free/nfsroot
--enable-smdk2410ial=yes
make
make install

（2）编译实例程序时，要加上jpeg库的支持，即在Makefile中加上-ljpeg。此时将在nfsroot生成的库文件和可执行文件移到ramdisk.image.gz相应的目录下。（具体参考华恒的2410开发手册）。

3．Minigui程序的编写

编程小技巧，我采取的方法是一刻不停地从摄像头采集到图片存储在/tmp/1.jpg中，在minigui中通过loadbitmap函数来加载图片。而图片加载后不会自动更新，不能自动根据1.jpg的改变自动变化。因此，我在程序中设定一个timer。每隔100ms刷新屏幕，基本上实现实时更新了。而出现另外一个问题，刷新时会以背景色来填充桌面，导致屏幕闪烁严重。故想到采用MSG_ERASEBKGND的方式，用前一张图片做为刷新屏幕时的填充背景图片。这样就保证了lcd上图像的连续性啦。

Minigui程序如下：其中一些自定义的函数跟mouse_capture中的一样，只是变采集单幅到采集多幅。具体您可以自己改一下：)。也可以向我索取源码。

```
#include <minigui/common.h>
#include <minigui/minigui.h>
#include <minigui/gdi.h>
#include <minigui/window.h>
#include <minigui/control.h>
#include    "spcav4l.h"

#define IDTIMER 100
static BITMAP bmp;
```

```
static int LoadBmpWinProc(HWND hWnd, int message, WPARAM wParam, LPARAM lParam)
{
	HDC hdc;
	RECT rc={0,0,240,320};
	switch (message) {
		case MSG_CREATE:
		SetTimer(hWnd,IDTIMER,100);
			return 0;

	case MSG_ERASEBKGND:
	{
		RECT rcTemp;
		 if( LoadBitmap(HDC_SCREEN,&bmp,"/tmp/1.jpg"))
		{
		printf("load wrong!\n");
		return -1;
		}
			GetClientRect(hWnd, &rcTemp);
			  hdc = BeginPaint (hWnd);
   FillBoxWithBitmap (hdc, rcTemp.left, rcTemp.top, rcTemp.right-rcTemp.left, rcTemp.bottom-rcTemp.top, &bmp);
			EndPaint(hWnd, hdc);
		return 0;
	}
	case MSG_TIMER:
	InvalidateRect(hWnd,&rc,TRUE);
	return 0;
		case MSG_CLOSE:
			UnloadBitmap (&bmp);
			DestroyMainWindow (hWnd);
			PostQuitMessage (hWnd);
			return 0;
	}
	return DefaultMainWinProc(hWnd, message, wParam, lParam);
}

int MiniGUIMain (int argc, const char* argv[])
{
	MSG Msg;
	HWND hMainWnd;
	MAINWINCREATE CreateInfo;
	char videodevice[] = "/dev/video0";
	char jpegfile[] = "/tmp/1.jpg";
	int grabmethod = 0;
```

```
    int format = VIDEO_PALETTE_JPEG;
    int width = 240;
    int height = 320;
    int i;
#ifdef _LITE_VERSION
     SetDesktopRect(0, 0, 1024, 768);
#endif
     CreateInfo.dwStyle = WS_VISIBLE | WS_BORDER | WS_CAPTION;
     CreateInfo.dwExStyle = WS_EX_NONE;
     CreateInfo.spCaption = "Load and display a bitmap";
     CreateInfo.hMenu = 0;
     CreateInfo.hCursor = GetSystemCursor(0);
     CreateInfo.hIcon = 0;
     CreateInfo.MainWindowProc = LoadBmpWinProc;
     CreateInfo.lx = 0;
     CreateInfo.ty = 0;
     CreateInfo.rx = 240;
     CreateInfo.by = 320;
     CreateInfo.iBkColor = PIXEL_lightwhite;
     CreateInfo.dwAddData = 0;
     CreateInfo.hHosting = HWND_DESKTOP;
     hMainWnd = CreateMainWindow (&CreateInfo);
      if (hMainWnd == HWND_INVALID)
          return -1;
     ShowWindow (hMainWnd, SW_SHOWNORMAL);
    memset(&videoIn, 0, sizeof (struct vdIn));
    if(init_videoIn(&videoIn, videodevice, width, height, format,grabmethod) == 0)
        {
    printf("init is ok!\n");
    }
    else printf("init is wrong!\n");
     while (GetMessage(&Msg, hMainWnd)) {
          TranslateMessage(&Msg);
        v4lGrab(&videoIn, jpegfile);
          DispatchMessage(&Msg);
     }
    close_v4l (&videoIn);
     MainWindowThreadCleanup (hMainWnd);
     return 0;
}
#ifndef _LITE_VERSION
#include <minigui/dti.c>
#endif
```

先写到这里吧，呵呵，希望能对您有所帮助。如果您在阅读的过程中发现问题，欢迎和我交流。

2006-7-7 晚

参考文献

1．HHARM2410摄像头调试记录 华恒科技

2．基于video4linux的视频设备编程 Lingzhi_Shi Apr 7 2004

3．《video4linux programming》 Alan Cox

4．《video streaming 探讨》 陈俊宏

5．《Video4Linux Kernel API Reference 》

6．http://www.hhcn.com/cgi-bin/topic.cgi?forum=1&topic=247&show=0