V4L2编程初体验

内容摘要：

Video for Linux two(Video4Linux2)简称V4L2，是V4L的改进版。V4L2是linux操作系统下用于采集图片、视频和音频数据的API接口，配合适当的视频采集设备和相应的驱动程序，可以实现图片、视频、音频等的采集。在远程会议、可视电话、视频监控系统和嵌入式多媒体终端中都有广泛的应用。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/video2下。

最近想做智能机器人，想加上视频采集这个模块，于是对linux下的视频方面的编程产生了兴趣，首先从入门开始吧！

一、Video for Linux Tow

在Linux下，所有外设都被看成一种特殊的文件，成为“设备文件”，可以象访问普通文件一样对其进行读写。一般来说，采用V4L2驱动的摄像头设备文件是/dev/v4l/video0。为了通用，可以建立一个到/dev/video0的链接。V4L2支持两种方式来采集图像：内存映射方式(mmap)和直接读取方式(read)。V4L2在include/linux/videodev.h文件中定义了一些重要的数据结构，在采集图像的过程中，就是通过对这些数据的操作来获得最终的图像数据。Linux系统V4L2的能力可在Linux内核编译阶段配置，默认情况下都有此开发接口。V4L2从Linux 2.5.x版本的内核中开始出现。

V4L2规范中不仅定义了通用API元素(Common API Elements)，图像的格式(Image Formats)，输入/输出方法(Input/Output)，还定义了Linux内核驱动处理视频信息的一系列接口(Interfaces)，这些接口主要有：

视频采集接口——Video Capture Interface;

视频输出接口—— Video Output Interface;

视频覆盖/预览接口——Video Overlay Interface;

视频输出覆盖接口——Video Output Overlay Interface;

编解码接口——Codec Interface。

二、操作流程

关于V4L2的介绍网上很多，这里简单说下我们经常常用相关结构体：

```
struct v4l2_requestbuffers reqbufs;//向驱动申请帧缓冲的请求，里面包含申请的个数
struct v4l2_capability cap;//这个设备的功能，比如是否是视频输入设备
struct v4l2_standard std;//视频的制式，比如PAL，NTSC
struct v4l2_format fmt;//帧的格式，比如宽度，高度等

struct v4l2_buffer buf;//代表驱动中的一帧
v4l2_std_id stdid;//视频制式，例如：V4L2_STD_PAL_B
struct v4l2_queryctrl query;//查询的控制
struct v4l2_control control;//具体控制的值
```

下面就介绍相关的操作流程：

1.打开设备文件。 int fd=open("/dev/video2",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
3.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
4.向驱动申请帧缓冲，一般不超过5个。struct v4l2_requestbuffers
5.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。mmap
6.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
7.开始视频的采集。VIDIOC_STREAMON
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
9.将缓冲重新入队列尾,这样可以循环采集。VIDIOC_QBUF
10.停止视频的采集。VIDIOC_STREAMOFF
11.关闭视频设备。close(fd);

以下详细介绍操作流程（相信对新手有用）：

1. 定义

V4L2(Video For Linux Two) 是内核提供给应用程序访问音、视频驱动的统一接口。

2. 工作流程：

打开设备－> 检查和设置设备属性－> 设置帧格式－> 设置一种输入输出方法（缓冲区管理）－> 循环获取数据－> 关闭设备。

3. 设备的打开和关闭：

#include <fcntl.h>

int open(const char *device_name, int flags);

#include <unistd.h>

int close(int fd);

例：

int fd=open("/dev/video2",O_RDWR);// 打开设备

close(fd);// 关闭设备

注意：V4L2 的相关定义包含在头文件<linux/videodev2.h> 中.

4. 查询设备属性： VIDIOC_QUERYCAP

相关函数：

int ioctl(int fd, int request, struct v4l2_capability *argp);

相关结构体：

struct v4l2_capability

{

__u8 driver[16]; // 驱动名字

__u8 card[32]; // 设备名字

__u8 bus_info[32]; // 设备在系统中的位置

__u32 version; // 驱动版本号

__u32 capabilities; // 设备支持的操作

__u32 reserved[4]; // 保留字段

};

capabilities 常用值:

V4L2_CAP_VIDEO_CAPTURE // 是否支持图像获取

例：显示设备信息

struct v4l2_capability cap;

ioctl(fd,VIDIOC_QUERYCAP,&cap);

printf("Driver Name:%s/nCard Name:%s/nBus info:%s/nDriver Version:%u.%u.%u/n",cap.driver,cap.card,cap.bus_info,(cap.version>>16)&0XFF, (cap.version>>8)&0XFF,cap.version&OXFF);

5. 帧格式：

VIDIOC_ENUM_FMT // 显示所有支持的格式

int ioctl(int fd, int request, struct v4l2_fmtdesc *argp);

struct v4l2_fmtdesc

{

__u32 index; // 要查询的格式序号，应用程序设置

enum v4l2_buf_type type; // 帧类型，应用程序设置

__u32 flags; // 是否为压缩格式

__u8 description[32]; // 格式名称

__u32 pixelformat; // 格式

__u32 reserved[4]; // 保留

};

例：显示所有支持的格式

struct v4l2_fmtdesc fmtdesc;

fmtdesc.index=0;

fmtdesc.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;

单片机(3)
工作心情(5)
旅游日记(1)
嵌入式(16)
嵌入式Linux驱动(1)
求职之旅(11)
人生感悟(3)
笑傲职场
眼球中的影视
杂记(2)
智能机器人之旅

随笔档案

2014年4月 (2)
2014年3月 (1)
2014年2月 (1)
2014年1月 (1)
2013年11月 (2)
2013年9月 (1)
2013年8月 (1)
2013年7月 (4)
2013年6月 (1)
2013年5月 (2)
2013年4月 (1)
2013年3月 (5)
2012年12月 (1)
2012年11月 (2)
2012年10月 (4)
2012年9月 (4)
2012年8月 (5)
2012年3月 (1)
2011年12月 (3)
2011年10月 (2)

相册

峨眉山之旅
我的作品
自己

Source File

最新评论 XML

1. Re:S3C6410嵌入式应用平台构建（一）

我当时只看了一半，的确其中有某些错误，但理解其中意思就好了，不要硬搬书上的代码，否则会被误导。这本书是我上学时候看的，现在都没去看那个了。

--We'll be better off

2. Re:S3C6410嵌入式应用平台构建（一）

你好，请问你那本《轻松自编小型嵌入式系统》看完吗？我试着编写，发现里面很多错误呢？能交流下吗？ luyuncheng@sina.com

--求ac

3. Re:移植属于自己的6410开发板的U-Boot

@xitry那就用2010版的嘛，其实我按照牛人的移植还出现其他问题，不过基本可以把linux引导起来。...

--We'll be better off

4. Re:移植属于自己的6410开发板的U-Boot

@We'll be better off我用2009.08的真是问题多

```
printf("Support format:/n");
while(ioctl(fd,VIDIOC_ENUM_FMT,&fmtdesc)!=-1)
{
printf("/t%d.%s/n",fmtdesc.index+1,fmtdesc.description);
fmtdesc.index++;
}
// 查看或设置当前格式
VIDIOC_G_FMT, VIDIOC_S_FMT
// 检查是否支持某种格式
VIDIOC_TRY_FMT
int ioctl(int fd, int request, struct v4l2_format *argp);
struct v4l2_format
{
enum v4l2_buf_type type;// 帧类型，应用程序设置
union fmt
{
struct v4l2_pix_format pix;// 视频设备使用
struct v4l2_window win;
struct v4l2_vbi_format vbi;
struct v4l2_sliced_vbi_format sliced;
__u8 raw_data[200];
};
};
struct v4l2_pix_format
{
__u32 width; // 帧宽，单位像素
__u32 height; // 帧高，单位像素
__u32 pixelformat; // 帧格式
enum v4l2_field field;
__u32 bytesperline;
__u32 sizeimage;
enum v4l2_colorspace colorspace;
__u32 priv;
};
例：显示当前帧的相关信息
struct v4l2_format fmt;
fmt.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;
ioctl(fd,VIDIOC_G_FMT,&fmt);
printf("Current data format information:/n/twidth:%d/n/theight:%d/n",fmt.fmt.width,fmt.fmt.height);
struct v4l2_fmtdesc fmtdesc;
fmtdesc.index=0;
fmtdesc.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;
while(ioctl(fd,VIDIOC_ENUM_FMT,&fmtdesc)!=-1)
{
if(fmtdesc.pixelformat & fmt.fmt.pixelformat)
{
printf("/tformat:%s/n",fmtdesc.description);
break;
}
```


多啊，至今还无法启动内核，环境变量也保存不成功...

--xitry

5. Re:移植属于自己的6410开发板的U-Boot

@xitry可以的，需要和你要引导的linux而定，如果linux内核入口地址是50008000，那么这儿就要改为50008000了...

--We'll be better off

阅读排行榜

1. V4L2编程初体验(2499)
2. 移植属于自己的6410开发板的U-Boot(1750)
3. Dnw下载工具还是Linux下的好（For OK6410）(631)
4. 开源视频服务软件MJPG-streamer移植(481)
5. STC15F2K60S2应用笔记(446)

评论排行榜

1. 移植属于自己的6410开发板的U-Boot(10)
2. 求职第八面——瑞昱半导体（深圳）(5)
3. 求职第五面——龙芯中科(4)
4. STC15F2K60S2应用笔记(3)
5. V4L2编程初体验(2)

推荐排行榜

1. 移植属于自己的6410开发板的U-Boot(1)
2. V4L2编程初体验(1)
3. Dnw下载工具还是Linux下的好（For OK6410）(1)
4. STC15F2K60S2应用笔记(1)

```
fmtdesc.index++;
}
```

例：检查是否支持某种帧格式

```
struct v4l2_format fmt;
fmt.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;
fmt.fmt.pix.pixelformat=V4L2_PIX_FMT_RGB32;
if(ioctl(fd,VIDIOC_TRY_FMT,&fmt)==-1)
if(errno==EINVAL)
printf("not support format RGB32!/n");
```

6. 图像的缩放

```
VIDIOC_CROPCAP
int ioctl(int fd, int request, struct v4l2_cropcap *argp);
struct v4l2_cropcap
{
enum v4l2_buf_type type;// 应用程序设置
struct v4l2_rect bounds;// 最大边界
struct v4l2_rect defrect;// 默认值
struct v4l2_fract pixelaspect;
};
// 设置缩放
VIDIOC_G_CROP,VIDIOC_S_CROP
int ioctl(int fd, int request, struct v4l2_crop *argp);
int ioctl(int fd, int request, const struct v4l2_crop *argp);
struct v4l2_crop
{
enum v4l2_buf_type type;// 应用程序设置
struct v4l2_rect c;
}
```

7. 申请和管理缓冲区，应用程序和设备有三种交换数据的方法，直接 read/write ，内存映射(memory mapping) ，用户指针。这里只讨论 memory mapping.

```
// 向设备申请缓冲区
VIDIOC_REQBUFS
int ioctl(int fd, int request, struct v4l2_requestbuffers *argp);
struct v4l2_requestbuffers
{
__u32 count; // 缓冲区内缓冲帧的数目
enum v4l2_buf_type type; // 缓冲帧数据格式
enum v4l2_memory memory; // 区别是内存映射还是用户指针方式
__u32 reserved[2];
};
enum v4l2_memoy {V4L2_MEMORY_MMAP,V4L2_MEMORY_USERPTR};
//count,type,memory 都要应用程序设置
```

例：申请一个拥有四个缓冲帧的缓冲区

```
struct v4l2_requestbuffers req;
req.count=4;
req.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;
req.memory=V4L2_MEMORY_MMAP;
ioctl(fd,VIDIOC_REQBUFS,&req);
```

获取缓冲帧的地址，长度：

VIDIOC_QUERYBUF

int ioctl(int fd, int request, struct v4l2_buffer *argp);

struct v4l2_buffer

{

__u32 index; //buffer 序号

enum v4l2_buf_type type; //buffer 类型

__u32 byteused; //buffer 中已使用的字节数

__u32 flags; // 区分是MMAP 还是USERPTR

enum v4l2_field field;

struct timeval timestamp;// 获取第一个字节时的系统时间

struct v4l2_timecode timecode;

__u32 sequence; // 队列中的序号

enum v4l2_memory memory;//IO 方式，被应用程序设置

union m

{

__u32 offset;// 缓冲帧地址，只对MMAP 有效

unsigned long userptr;

};

__u32 length;// 缓冲帧长度

__u32 input;

__u32 reserved;

};

MMAP ，定义一个结构体来映射每个缓冲帧。

Struct buffer

{

void* start;

unsigned int length;

}*buffers;

#include <sys/mman.h>

void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);

//addr 映射起始地址，一般为NULL ，让内核自动选择

//length 被映射内存块的长度

//prot 标志映射后能否被读写，其值为PROT_EXEC,PROT_READ,PROT_WRITE, PROT_NONE

//flags 确定此内存映射能否被其他进程共享，MAP_SHARED,MAP_PRIVATE

//fd,offset, 确定被映射的内存地址

返回成功映射后的地址，不成功返回MAP_FAILED ((void*)-1);

int munmap(void *addr, size_t length);// 断开映射

//addr 为映射后的地址，length 为映射后的内存长度

例：将四个已申请到的缓冲帧映射到应用程序，用buffers 指针记录。

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

if (!buffers) {

fprintf (stderr, "Out of memory/n");

exit (EXIT_FAILURE);

}

// 映射

for (unsigned int n_buffers = 0; n_buffers < req.count; ++n_buffers) {

struct v4l2_buffer buf;

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

buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;

```
buf.memory = V4L2_MEMORY_MMAP;
buf.index = n_buffers;
// 查询序号为n_buffers 的缓冲区，得到其起始物理地址和大小
if (-1 == ioctl (fd, VIDIOC_QUERYBUF, &buf))
exit(-1);
buffers[n_buffers].length = buf.length;
// 映射内存
buffers[n_buffers].start =mmap (NULL,buf.length,PROT_READ | PROT_WRITE ,MAP_SHARED,fd, buf.m.offset);
if (MAP_FAILED == buffers[n_buffers].start)
exit(-1);
}
```

8. 缓冲区处理好之后，就可以开始获取数据了

```
// 启动/ 停止数据流
VIDIOC_STREAMON,VIDIOC_STREAMOFF
int ioctl(int fd, int request, const int *argp);
//argp 为流类型指针，如V4L2_BUF_TYPE_VIDEO_CAPTURE.
在开始之前，还应当把缓冲帧放入缓冲队列：
VIDIOC_QBUF// 把帧放入队列
VIDIOC_DQBUF// 从队列中取出帧
int ioctl(int fd, int request, struct v4l2_buffer *argp);
例：把四个缓冲帧放入队列，并启动数据流
unsigned int i;
enum v4l2_buf_type type;
// 将缓冲帧放入队列
for (i = 0; i < 4; ++i)
{
struct v4l2_buffer buf;
buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
buf.memory = V4L2_MEMORY_MMAP;
buf.index = i;
ioctl (fd, VIDIOC_QBUF, &buf);
}
type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
ioctl (fd, VIDIOC_STREAMON, &type);
// 这有个问题，这些buf 看起来和前面申请的buf 没什么关系，为什么呢?
例：获取一帧并处理
struct v4l2_buffer buf;
CLEAR (buf);
buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
buf.memory = V4L2_MEMORY_MMAP;
// 从缓冲区取出一个缓冲帧
ioctl (fd, VIDIOC_DQBUF, &buf);
// 图像处理
process_image (buffers[buf.index].start);
// 将取出的缓冲帧放回缓冲区
ioctl (fd, VIDIOC_QBUF, &buf);
```

关于视频采集方式

操作系统一般把系统使用的内存划分成用户空间和内核空间，分别由应用程序管理和操作系统管理。应用程序可以直

接访问内存的地址，而内核空间存放的是供内核访问的代码和数据，用户不能直接访问。v4l2捕获的数据，最初是存放在内核空间的，这意味着用户不能直接访问该段内存，必须通过某些手段来转换地址。

一共有三种视频采集方式：使用read、write方式；内存映射方式和用户指针模式。

read、write方式:在用户空间和内核空间不断拷贝数据，占用了大量用户内存空间，效率不高。

内存映射方式：把设备里的内存映射到应用程序中的内存控件，直接处理设备内存，这是一种有效的方式。上面的mmap函数就是使用这种方式。

用户指针模式：内存片段由应用程序自己分配。这点需要在v4l2_requestbuffers里将memory字段设置成V4L2_MEMORY_USERPTR。

处理采集数据

V4L2有一个数据缓存，存放req.count数量的缓存数据。数据缓存采用FIFO的方式，当应用程序调用缓存数据时，缓存队列将最先采集到的 视频数据缓存送出，并重新采集一张视频数据。这个过程需要用到两个ioctl命令,VIDIOC_DQBUF和VIDIOC_QBUF：

structv4l2_buffer buf;

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

buf.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;

buf.memory=V4L2_MEMORY_MMAP;

buf.index=0;

//读取缓存

if(ioctl(cameraFd, VIDIOC_DQBUF, &buf) == -1)

{

return-1;

}

//……视频处理算法

//重新放入缓存队列

if(ioctl(cameraFd, VIDIOC_QBUF, &buf) == -1) {

return-1;

}

关闭视频设备

使用close函数关闭一个视频设备

close(cameraFd)

还需要使用munmap方法。

下面是我自己参照网上一步步写的,成功采集.

```
/*============================================================================
#     FileName: v4l2.c
#         Desc: this program aim to get image from USB camera,
#               used the V4L2 interface.
#       Author: LiXiaoming
#        Email: lixiaoming5700@gmail.com
#     HomePage: http://www.cnblogs.com/lixiaoming90
#      Version: 0.0.1
#   LastChange: 2012-08-22 15:52:37
#      History:
============================================================================*/
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <stdlib.h>
#include <linux/types.h>
#include <linux/videodev2.h>
#include <malloc.h>
#include <math.h>
#include <string.h>
#include <sys/mman.h>
```

```
#include <errno.h>
#include <assert.h>

#define FILE_VIDEO  "/dev/video2"
#define JPG "/lxm/picture/image%d.jpg"

typedef struct{
    void *start;
    int length;
}BUFTYPE;
BUFTYPE *usr_buf;
static unsigned int n_buffer = 0;

//set video capture ways(mmap)
int init_mmap(int fd)
{
    //to request frame cache, contain requested counts
    struct v4l2_requestbuffers reqbufs;
    //request V4L2 driver allocation video cache
    //this cache is locate in kernel and need mmap mapping
    memset(&reqbufs, 0, sizeof(reqbufs));
    reqbufs.count = 4;
    reqbufs.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    reqbufs.memory = V4L2_MEMORY_MMAP;

    if(-1 == ioctl(fd,VIDIOC_REQBUFS,&reqbufs)){
        perror("Fail to ioctl 'VIDIOC_REQBUFS'");
        exit(EXIT_FAILURE);
    }

    n_buffer = reqbufs.count;
    printf("n_buffer = %d\n", n_buffer);
    usr_buf = calloc(reqbufs.count, sizeof(usr_buf));
    if(usr_buf == NULL){
        printf("Out of memory\n");
        exit(-1);
    }

    //map kernel cache to user process
    for(n_buffer = 0; n_buffer < reqbufs.count; ++n_buffer){
        //stand for a frame
        struct v4l2_buffer buf;
        memset(&buf, 0, sizeof(buf));
        buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
        buf.memory = V4L2_MEMORY_MMAP;
        buf.index = n_buffer;

        //check the information of the kernel cache requested
        if(-1 == ioctl(fd,VIDIOC_QUERYBUF,&buf))
        {
            perror("Fail to ioctl : VIDIOC_QUERYBUF");
            exit(EXIT_FAILURE);
        }

        usr_buf[n_buffer].length = buf.length;
        usr_buf[n_buffer].start =
            (char *)mmap(
                    NULL,
                    buf.length,
                    PROT_READ | PROT_WRITE,
                    MAP_PRIVATE,
                    fd,
                    buf.m.offset
                );
        if(MAP_FAILED == usr_buf[n_buffer].start)
        {
            perror("Fail to mmap");
```

```
            exit(EXIT_FAILURE);
        }
    }
    return 0;
}

//initial camera device
int init_camera_device(int fd)
{
    //decive fuction, such as video input
    struct v4l2_capability cap;
    //video standard,such as PAL,NTSC
    struct v4l2_standard std;
    //frame format
    struct v4l2_format tv_fmt;
    //check control
    struct v4l2_queryctrl query;
    //detail control value
    struct v4l2_fmtdesc fmt;
    int ret;
    //get the format of video supply
    memset(&fmt, 0, sizeof(fmt));
    fmt.index = 0;
    //supply to image capture
    fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    // show all format of supply
    printf("Support format:\n");
    while(ioctl(fd, VIDIOC_ENUM_FMT, &fmt) == 0){
        fmt.index++;
        printf("pixelformat = ''%c%c%c%c''\ndescription = ''%s''\n",fmt.pixelformat &
0xFF, (fmt.pixelformat >> 8) & 0xFF,(fmt.pixelformat >> 16) & 0xFF, (fmt.pixelformat >
> 24) & 0xFF,fmt.description);
    }
    //check video decive driver capability
    ret = ioctl(fd, VIDIOC_QUERYCAP, &cap);
    if(ret < 0){
        perror("Fail to ioctl VIDEO_QUERYCAP");
        exit(EXIT_FAILURE);
    }

    //judge wherher or not to be a video-get device
    if(!(cap.capabilities & V4L2_BUF_TYPE_VIDEO_CAPTURE))
    {
        printf("The Current device is not a video capture device\n");
        exit(-1);
    }

    //judge whether or not to supply the form of video stream
    if(!(cap.capabilities & V4L2_CAP_STREAMING))
    {
        printf("The Current device does not support streaming i/o\n");
        exit(EXIT_FAILURE);
    }

    //set the form of camera capture data
    tv_fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    tv_fmt.fmt.pix.width = 680;
    tv_fmt.fmt.pix.height = 480;
    tv_fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_MJPEG;
    tv_fmt.fmt.pix.field = V4L2_FIELD_INTERLACED;
    if (ioctl(fd, VIDIOC_S_FMT, &tv_fmt)< 0) {
        printf("VIDIOC_S_FMT\n");
        exit(-1);
        close(fd);
    }
    //initial video capture way(mmap)
    init_mmap(fd);
```

```
159     return 0;
160 }
161
162 int open_camera_device()
163 {
164     int fd;
165     //open video device with block
166     fd = open(FILE_VIDEO, O_RDONLY);
167     if(fd < 0){
168         perror(FILE_VIDEO);
169         exit(EXIT_FAILURE);
170     }
171     return fd;
172 }
173
174 int start_capture(int fd)
175 {
176     unsigned int i;
177     enum v4l2_buf_type type;
178     //place the kernel cache to a queue
179     for(i = 0; i < n_buffer; i++){
180         struct v4l2_buffer buf;
181         memset(&buf, 0, sizeof(buf));
182         buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
183         buf.memory = V4L2_MEMORY_MMAP;
184         buf.index = i;
185
186         if(-1 == ioctl(fd, VIDIOC_QBUF, &buf)){
187             perror("Fail to ioctl 'VIDIOC_QBUF'");
188             exit(EXIT_FAILURE);
189         }
190     }
191
192     //start capture data
193     type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
194     if(-1 == ioctl(fd, VIDIOC_STREAMON, &type)){
195         printf("i=%d.\n", i);
196         perror("VIDIOC_STREAMON");
197         close(fd);
198         exit(EXIT_FAILURE);
199     }
200     return 0;
201 }
202
203 int process_image(void *addr, int length)
204 {
205     FILE *fp;
206     static int num = 0;
207     char image_name[20];
208
209     sprintf(image_name, JPG, num++);
210     if((fp = fopen(image_name, "w")) == NULL){
211         perror("Fail to fopen");
212         exit(EXIT_FAILURE);
213     }
214     fwrite(addr, length, 1, fp);
215     usleep(500);
216     fclose(fp);
217     return 0;
218 }
219
220 int read_frame(int fd)
221 {
222     struct v4l2_buffer buf;
223     unsigned int i;
224     memset(&buf, 0, sizeof(buf));
225     buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
```

```
226     buf.memory = V4L2_MEMORY_MMAP;
227     //put cache from queue
228     if(-1 == ioctl(fd, VIDIOC_DQBUF,&buf)){
229         perror("Fail to ioctl 'VIDIOC_DQBUF'");
230         exit(EXIT_FAILURE);
231     }
232 
233     assert(buf.index < n_buffer);
234     //read process space's data to a file
235     process_image(usr_buf[buf.index].start, usr_buf[buf.index].length);
236     if(-1 == ioctl(fd, VIDIOC_QBUF,&buf)){
237         perror("Fail to ioctl 'VIDIOC_QBUF'");
238         exit(EXIT_FAILURE);
239     }
240     return 1;
241 }
242 
243 int mainloop(int fd)
244 {
245     int count = 10;
246 
247     while(count-- > 0)
248     {
249         for(;;)
250         {
251             fd_set fds;
252             struct timeval tv;
253             int r;
254 
255             FD_ZERO(&fds);
256             FD_SET(fd,&fds);
257 
258             /*Timeout*/
259             tv.tv_sec = 2;
260             tv.tv_usec = 0;
261             r = select(fd + 1,&fds,NULL,NULL,&tv);
262 
263             if(-1 == r)
264             {
265                 if(EINTR == errno)
266                     continue;
267                 perror("Fail to select");
268                 exit(EXIT_FAILURE);
269             }
270 
271             if(0 == r)
272             {
273                 fprintf(stderr,"select Timeout\n");
274                 exit(-1);
275             }
276 
277             if(read_frame(fd))
278                 break;
279         }
280     }
281     return 0;
282 }
283 
284 void stop_capture(int fd)
285 {
286     enum v4l2_buf_type type;
287     type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
288     if(-1 == ioctl(fd,VIDIOC_STREAMOFF,&type))
289     {
290         perror("Fail to ioctl 'VIDIOC_STREAMOFF'");
291         exit(EXIT_FAILURE);
292     }
```

```
293     return;
294 }
295
296 void close_camera_device(int fd)
297 {
298     unsigned int i;
299     for(i = 0;i < n_buffer; i++)
300     {
301         if(-1 == munmap(usr_buf[i].start,usr_buf[i].length)){
302             exit(-1);
303         }
304     }
305     free(usr_buf);
306
307     if(-1 == close(fd))
308     {
309         perror("Fail to close fd");
310         exit(EXIT_FAILURE);
311     }
312     return;
313 }
314
315 int main()
316 {
317     int fd;
318     fd = open_camera_device();
319     init_camera_device(fd);
320     start_capture(fd);
321     mainloop(fd);
322     stop_capture(fd);
323     close_camera_device(fd);
    return 0;
}
```

终于可以读取USB摄像头的图像了，感谢前人的文章。后面需要编写的就是图像处理,例如H.264编码和通过UDP数据传输到客户端等等。

参考文章：

fengjingge815的空间：http://hi.baidu.com/fengjingge815/item/64597e0c68e870e3a01034fd

铠甲&秦的梅阿查： http://blog.chinaunix.net/uid/11765716.html（这篇文章总结的很好，很详细）

草根老师： http://blog.chinaunix.net/uid-26833883-id-3249346.html

分类: 嵌入式

We'll be better off
关注 - 2
粉丝 - 11
+加关注

1 0

(请您对文章做出评价)

« 上一篇：嵌入式H.264移植
» 下一篇：12年暑假

posted @ 2012-08-25 22:58 We'll be better off 阅读(2500) 评论(2) 编辑 收藏

评论列表

#1楼 2013-11-27 15:54 莫扎特也是程序猿

博主，我运行文中的例子程序，mm a p出现错误，提示：Invalid argument，请问是什么原因呢，真是纠结啊

支持(0) 反对(0)

#2楼[楼主] 2013-11-27 22:45 Simiar

@莫扎特也是程序猿
我不太清楚你的情况是怎样的，从报错来看问题处在内存映射mmap,你可以查一下你的mmap操作有什么问题没有。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

博客园首页 博问 新闻 闪存 程序员招聘 知识库

最新IT新闻:
· IT界最难以替代的9个角色
· 碾压谷歌：牛人自制“火星街景”
· 阿里巴巴最怕投资者追问的三件事
· 业界良心：Square开源Viewfinder，25万行代码全公布！
· 当数学家遇上大数据
» 更多新闻...

最新知识库文章:
· 当随机不够随机：一个在线扑克游戏的教训
· 闭包漫谈（从抽象代数及函数式编程角度）
· Facebook和Google如何激发工程师的创造力
· 好的程序员到底好在哪里？
· 关于坐标系和投影的相关知识探讨
» 更多知识库文章...