人间凑数
码龄5年 暂无认证
89 原创 | 21万+ 周排名 | 113万+ 总排名 | 33万+ 访问 | 等级
4192 积分 | 193 粉丝 | 217 获赞 | 80 评论 | 1596 收藏
私信 关注

搜博主文章

热门文章
Linux多线程编程---线程间同步（互斥锁、条件变量、信号量和读写锁） 26436
通过systemback制作系统镜像以及镜像安装，实现系统备份和复刻 18703
VMware虚拟机崩溃的解决方法（.vmx损坏，其他文件完好） 18057
V4L2图像采集+图片格式转换（YUYV、RGB、JPEG） 12986
OpenCV图像处理实际案例（一）---图像倾斜矫正（仿射变换）和去边（轮廓查找+ROI提取） 11088

最新评论
V4L2图像采集+图片格式转换（YUYV、...
tenacity0o: 博主大大 有源码吗?
通过systemback制作系统镜像以及镜像...
inspur_image: 什么破软件，ubuntu18.04不适配不让下载，一顿操作后下载成功了...
实现嵌入式linux自动同步网络时间---NTP
Quiet.W: 你好，我也遇到一样的问题，请问你解决了吗
V4L2下摄像头的详细参数调整
脚踏实地～: 这个要用while循环查，并且每次v4l2_fmtdesc.index加1，直到返回值不...
Opencv学习---图像通道分割（split）与...
wsclx_: planes[0] = 0; planes[1] = 0; planes[2] = 255; 博主，这里应该有点错误，应...

您愿意向朋友推荐“博客详情页”吗?
强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章
学习资源搜集
Windows下安装使用SVN
OpenCV实现视频背景消除与前景ROI提取
2021年 1篇 2020年 10篇
2019年 62篇 2018年 61篇

V4L2下摄像头的详细参数调整

原创 人间凑数 于 2019-02-26 10:24:55 发布 10129 收藏 65
分类专栏：嵌入式
嵌入式 专栏收录该内容 8 订阅 97 篇文章 订阅专栏

(Linux下V4L2相关头文件所在路径为/内核源码目录/include/linux/videodev2.h，V4L2相关API文档可查看链接 https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html)

摄像头（相机）常见参数：

白平衡（自动白平衡AWB）及色温、曝光（自动曝光AE、曝光补偿EV）、亮度、对比度、饱和度、色度（色调+饱和度）、锐度（也叫清晰度）、背光补偿（也叫逆光补偿）、增益、对焦等

（注：不同摄像头开放的参数不一致，需提前确认该款摄像头的可调参数，未开放的参数是无法调整的！！！）

上述参数涵义如有不懂，可自行维基百科https://zh.wikipedia.org/wiki/Wikipedia:%E9%A6%96%E9%A1%B5和百度词条https://baike.baidu.com/item/科普。

常见的 ioctl 命令：

```
VIDIOC_QUERYCAP    /* 获取设备支持的操作 */
VIDIOC_G_FMT       /* 获取设置支持的视频格式 */
VIDIOC_S_FMT       /* 设置捕获视频的格式 */
VIDIOC_REQBUFS     /* 向驱动提出申请内存的请求 */
VIDIOC_QUERYBUF    /* 向驱动查询申请到的内存 */
VIDIOC_QBUF        /* 将空闲的内存加入可捕获视频的队列 */
VIDIOC_DQBUF       /* 将已经捕获好视频的内存拉出已捕获视频的队列 */
VIDIOC_STREAMON    /* 打开视频流 */
VIDIOC_STREAMOFF   /* 关闭视频流 */
VIDIOC_QUERYCTRL   /* 查询驱动是否支持该命令 */
VIDIOC_G_CTRL      /* 获取当前命令值 */
VIDIOC_S_CTRL      /* 设置新的命令值 */
VIDIOC_G_TUNER     /* 获取调谐器信息 */
VIDIOC_S_TUNER     /* 设置调谐器信息 */
VIDIOC_G_FREQUENCY /* 获取调谐器频率 */
VIDIOC_S_FREQUENCY /* 设置调谐器频率 */
```

参数控制相关函数及结构体：

函数：ioctl(fd,VIDIOC_QUERYCAP,struct v4l2_streamparm *argp)；

作用：查询设备能力

```
struct v4l2_capability {
    __u8    driver[16];    /* i.e. "bttv" */
    __u8    card[32];      /* i.e. "Hauppauge WinTV" */
    __u8    bus_info[32];  /* "PCI:" + pci_name(pci_dev) */
    __u32   version;       /* should use KERNEL_VERSION() */
    __u32   capabilities;  /* Device capabilities */
    __u32   reserved[4];
};
```

栗子：

```c
/*获取摄像头信息*/
struct v4l2_capability cap;

if (ioctl(cam_fd, VIDIOC_QUERYCAP, &cap) < 0)
{
    perror("get info failed!");
    return -1;
}

printf("Driver Name:%s\n Card Name:%s\n Bus info:%s\n version:%d\n capabilities:%x\n \n ", cap.driver, c

if ((cap.capabilities & V4L2_CAP_VIDEO_CAPTURE) == V4L2_CAP_VIDEO_CAPTURE)
{
    printf("Device %s: supports capture.\n",DEV_NAME);
}
if ((cap.capabilities & V4L2_CAP_STREAMING) == V4L2_CAP_STREAMING)
{
    printf("Device %s: supports streaming.\n",DEV_NAME);
}
```

函数： int ioctl(fd, VIDIOC_ENUM_FMT, struct v4l2_fmtdesc *argp);

作用：获取当前驱动支持的视频格式

```
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
};
```

栗子：

```c
printf(" [**********所有支持的格式: **********] \n");

struct v4l2_fmtdesc dis_fmtdesc;

dis_fmtdesc.index=0;

dis_fmtdesc.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;

printf("Support format:\n");

while(ioctl(cam_fd,VIDIOC_ENUM_FMT,&dis_fmtdesc)!=-1)

{

printf("\t%d.%s\n",dis_fmtdesc.index+1,dis_fmtdesc.description);

dis_fmtdesc.index++;

}

printf("\n");
```

函数：ioctl(fd,VIDIOC_QUERYCAP,struct v4l2_queryctrl *argp)；

作用：用作查询某些参数的一些属性要求（如最大最小范围，默认值等）

```
struct v4l2_queryctrl {
    __u32            id;
    enum v4l2_ctrl_type  type;
    __u8             name[32];  /* Whatever */
    __s32            minimum;   /* Note signedness */
    __s32            maximum;
    __s32            step;
    __s32            default_value;
    __u32            flags;
    __u32            reserved[2];
};
```

函数：ioctl(fd,VIDIOC_G_FMT,struct v4l2_format *argp)；//VIDIOC_S_FMT

作用：分辨率、图像采集格式相关设置

```
struct v4l2_format {
    enum v4l2_buf_type type;
    union {
        struct v4l2_pix_format      pix;     /* V4L2_BUF_TYPE_VIDEO_CAPTURE */
        struct v4l2_pix_format_mplane  pix_mp;  /* V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE */
        struct v4l2_window          win;     /* V4L2_BUF_TYPE_VIDEO_OVERLAY */
        struct v4l2_vbi_format      vbi;     /* V4L2_BUF_TYPE_VBI_CAPTURE */
        struct v4l2_sliced_vbi_format  sliced;  /* V4L2_BUF_TYPE_SLICED_VBI_CAPTURE */
        __u8    raw_data[200];           /* user-defined */
```


分类专栏
医学图像 1篇
网络编程
嵌入式 97篇
IOT
网络编程 10篇
QT 13篇
OPENCV 22篇
计算机视觉 17篇
数据库 1篇
系统 5篇

} fmt;
};
栗子：

```
    printf(" 【**********获取当前格式信息：**********】\n");
    Format.type= V4L2_BUF_TYPE_VIDEO_CAPTURE;
    if(ioctl(cam_fd,VIDIOC_G_FMT,&Format)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf(">>:  %d * %d\n",Format.fmt.pix.width,Format.fmt.pix.height);
    printf("pix.pixelformat:%c%c%c%c\n", \
            Format.fmt.pix.pixelformat & 0xFF,\
            (Format.fmt.pix.pixelformat >> 8) & 0xFF, \
            (Format.fmt.pix.pixelformat >> 16) & 0xFF,\
            (Format.fmt.pix.pixelformat >> 24) & 0xFF);
    printf("\n");
```

函数：ioctl(fd,VIDIOC_G_PARM,struct v4l2_streamparm *argp）；//VIDIOC_S_PARM

作用：流相关 (如帧率)

```
struct v4l2_streamparm {
    enum v4l2_buf_type type;
    union {
        struct v4l2_captureparm capture;
        struct v4l2_outputparm  output;
        __u8    raw_data[200];  /* user-defined */
    } parm;
};
```

栗子：

```
printf(" 【**********获取帧率信息**********】\n");

struct v4l2_streamparm Stream_Parm;

Stream_Parm.type=V4L2_BUF_TYPE_VIDEO_CAPTURE;

if(ioctl(cam_fd,VIDIOC_G_PARM,&Stream_Parm)==-1)

{

perror("ioctl");

exit(EXIT_FAILURE);

}

printf(">>: Frame rate: %u/%u\n",Stream_Parm.parm.capture.timeperframe.numerator,Stream_Parm.parm.capture.ti

printf("\n");
```

函数：ioctl(fd,VIDIOC_G_CTRL,struct v4l2_control*argp）；//VIDIOC_S_CTRL

作用：参数控制（白平衡、色温、对比度、饱和度、锐度、曝光等，均有该结构控制）

```
struct v4l2_control {
    __u32           id;     //id即要控制的参数id（例白平衡ID为V4L2_CID_AUTO_WHITE_BALANCE）
    __s32           value;
};
```

栗子：

```
    printf(" 【**********设置手动白平衡：**********】\n");
    ctrl.id = V4L2_CID_AUTO_WHITE_BALANCE;
    ctrl.value = V4L2_WHITE_BALANCE_MANUAL ;
    if(ioctl(cam_fd,VIDIOC_G_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");

    /**********设置白平衡色温**********/
    printf(" 【**********设置白平衡色温**********】\n");
    ctrl.id = V4L2_CID_WHITE_BALANCE_TEMPERATURE;
    ctrl.value = 5100;
     if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");

    printf(" 【**********设置亮度**********】\n");
    ctrl.id= V4L2_CID_BRIGHTNESS;
    ctrl.value = 40;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");

    printf(" 【**********设置对比度**********】\n");
    ctrl.id = V4L2_CID_CONTRAST;
    ctrl.value= 45;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    sleep(1);
    printf("\n");

    printf(" 【**********设置饱和度**********】\n");
    ctrl.id = V4L2_CID_SATURATION;
    ctrl.value= 60;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");

    printf(" 【**********设置色度**********】\n");
    ctrl.id = V4L2_CID_HUE;
    ctrl.value = 1;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");

    printf(" 【**********设置锐度**********】\n");
    ctrl.id = V4L2_CID_SHARPNESS;
    ctrl.value = 4;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");


    printf(" 【**********设置背光补偿**********】\n");
    ctrl.id = V4L2_CID_BACKLIGHT_COMPENSATION;
    ctrl.value = 3;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
    }
    printf("\n");

    printf(" 【**********设置伽玛值**********】\n");
    ctrl.id = V4L2_CID_GAMMA;
    ctrl.value = 120;
    if(ioctl(cam_fd,VIDIOC_S_CTRL,&ctrl)==-1)
    {
        perror("ioctl");
        exit(EXIT_FAILURE);
```

```
    }
    printf("\n");
```

设置曝光值：

1. 首先将曝光模式修改为手动曝光。
2. 设置曝光档次或者具体的曝光值。

例1：得到曝光模式，设置为手动曝光模式

```
int ret;
struct v4l2_control ctrl;
//得到曝光模式
ctrl.id = V4L2_CID_EXPOSURE_AUTO;
ret = ioctl(Handle, VIDIOC_G_CTRL, &ctrl);

if (ret < 0)
{
    printf("Get exposure auto Type failed\n");

    return V4L2_UTILS_GET_EXPSURE_AUTO_TYPE_ERR;
}

printf("\nGet Exposure Auto Type:[%d]\n", ctrl.value);
```

//设置曝光模式为手动模式

```
ctrl.id = V4L2_CID_EXPOSURE_AUTO;
ctrl.value = V4L2_EXPOSURE_MANUAL;  //手动曝光模式
ret = ioctl(Handle, VIDIOC_S_CTRL, &ctrl);
if (ret < 0)
{
   printf("Get exposure auto Type failed\n");

   return V4L2_UTILS_SET_EXPSURE_AUTO_TYPE_ERR;

}
```

例2：在设置为手动模式后，再得到和设置曝光等级

```
int ret;
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_EXPOSURE;  //得到曝光档次，A20接受从 -4到4 共9个档次
ret = ioctl(Handle, VIDIOC_G_CTRL, &ctrl);
if (ret < 0)
{
    printf("Get exposure failed (%d)\n", ret);
    return V4L2_UTILS_GET_EXPSURE_ERR;
}

printf("\nGet Exposure :[%d]\n", ctrl.value);

//设置曝光档次
ctrl.id = V4L2_CID_EXPOSURE;
ctrl.value = -4;
ret = ioctl(Handle, VIDIOC_S_CTRL, &ctrl);
if (ret < 0)
{
    printf("Set exposure failed (%d)\n", ret);
    return V4L2_UTILS_SET_EXPSURE_ERR;
}
```

例3：在设置为手动模式后，再得到和设置曝光绝对值：

```
int ret;
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_EXPOSURE_ABSOLUTE;
ret = ioctl(handle, VIDIOC_G_CTRL, &ctrl);
if (ret < 0)
{
    printf("Set exposure failed (%d)\n", ret);

    //return V4L2_UTILS_SET_EXPSURE_ERR;
}

printf("\nGet ABS EXP Success:[%d]\n", ctrl.value);

sleep(1);

//设置曝光绝对值
ctrl.id = V4L2_CID_EXPOSURE_ABSOLUTE;
ctrl.value = 300;  //单位100us
ret = ioctl(handle, VIDIOC_S_CTRL, &ctrl);
if (ret < 0)
{
    printf("Set exposure failed (%d)\n", ret);

    //return V4L2_UTILS_SET_EXPSURE_ERR;
}
```

（以上参数经过测试均能有效设置，但**曝光、自动对焦、增益**无法设置通过,会出现**ioctl: Invalid argument**问题，原因未知，如有朋友知晓，诚请赐教，在此先行谢过）。