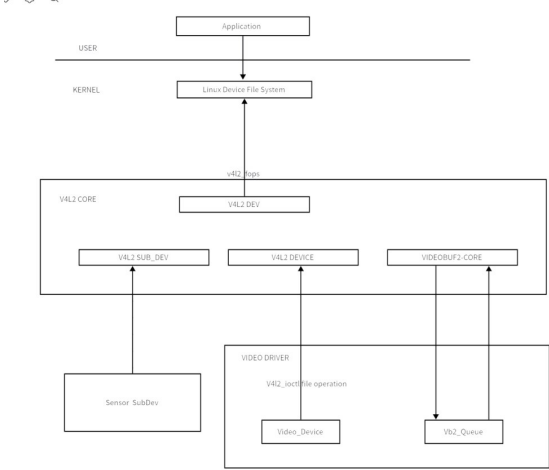


一.什么是 V4L2 框架?

V4L2英文全称是Video for Linux2, 它是专门为视频设备设计的内核驱动。在做视频的开发中, 一般我们操控V4L2的设备节点就可以直接对摄像头进行操作。通常V4L2在Linux的设备节点是"/dev/video0"。无论是MP摄像头还是UVC摄像头, 它们底层默认操作的都是/dev/video0的节点。

二. V4L2的代码框架图:



从这张图可以看出, 在使用V4L2进行摄像头操作的时候, 都需要访问内核驱动, 而整个设备驱动都是由Linux Device File System进行管理, Linux File System管理的是V4L2CORE的功能, 而V4L2_CORE里面包含了V4L2_DEV、V4L2_SUB_DEV、V4L2_DEVICE、VIDEOBUF2_CORE, 其中, V4L2_DEVICE主要是管理视频设备驱动、VIDEOBUF2_CORE主要是管理缓冲队列的数据, V4L2_SUB_DEV主要是管理视频设备的子系统, 而应用层若想对整个V4L2驱动层进行控制的话, 只需要对v4l2_dev进行ops(文件形式控制)操作即可, 因为v4l2_dev是驱动层对用户层提供的接口。

三. V4L2代码开发流程:

3.1.打开设备节点:

```
fd = open("/dev/video0", O_RDWR);
if (fd < 0)
{
    printf("open \"%s\" error\n", dev);
    return -1;
}
```

打开/dev/video0视频设备节点

3.2.查询设备的能力

```
struct v4l2_capability cap;
int ret = ioctl(fd, VIDIOC_QUERYCAP, &cap);
if (ret < 0)
{
    printf("VIDIOC_QUERYCAP error\n");
    return -1;
}
printf("driver : %s\n", cap.driver);
printf("device : %s\n", cap.card);
printf("bus : %s\n", cap.bus_info);
printf("version : %d\n", cap.version);
if (cap.capabilities & V4L2_BUF_TYPE_VIDEO_CAPTURE)
{
    /*判断是否为视频捕获设备*/
    if (cap.capabilities & V4L2_CAP_STREAMING)
    {
        /*判断是否支持视频流捕获*/
        printf("support capture\n");
    }
    else
    {
        printf("unsupport capture\n");
    }
}
else
{
    printf("error\n");
    return -1;
}
```

利用ioctl函数访问V4L2的底层命令VIDIOC_QUERYCAP主要是查询摄像头的性能属性。

3.3.获取摄像头支持的格式, 并进行像素格式设置

```
struct v4l2_fmtdesc fmdesc;
fmdesc.index = 0;
fmdesc.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
while (ioctl(fd, VIDIOC_ENUM_FMT, &fmdesc) != -1)
{
    printf("\t%d.%s\n", fmdesc.index + 1, fmdesc.description);
    fmdesc.index++;
}

struct v4l2_format fmt;
fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE; //摄像头缓冲
fmt.fmt.pix.width = 640;
fmt.fmt.pix.height = 480;
fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUV422P;
if (ioctl(fd, VIDIOC_S_FMT, &fmt) < 0)
{
    printf("set format:V4L2_PIX_FMT_MJPEG failed\n");
    return -1;
}
```

通过ioctl VIDIOC_ENUM_FMT获取对应的支持格式, 并且对摄像头进行相应格式的设置, 包括: width、height、pixelformat等等。设置完成之后, 再使用ioctl命令进行使能操作, 使能对应的关键字: VIDIOC_S_FMT。

3.4.内存映射到用户空间并进行队列操作

```
int amap_v4l2_buffer()
{
    usr_buf = (struct USER_BUFFER *)calloc(buf_num, sizeof(struct USER_BUFFER));
    if (!usr_buf)
    {
        printf("calloc \"frame buffer\" error : Out of memory\n");
        return -1;
    }

    struct v4l2_requestbuffers req;
    req.count = buf_num; //帧缓冲数量
    req.type = V4L2_BUF_TYPE_VIDEO_CAPTURE; //视频捕获缓冲区类型
    req.memory = V4L2_MEMORY_MMAP; //使用映射缓冲区方式
    if (ioctl(fd, VIDIOC_REQBUFS, &req) < 0)
    {
        printf("VIDIOC_REQBUFS fail\n");
        return -1;
    }

    /*映射内核缓冲区到用户空间缓冲区*/
}
```

目录

一.什么是V4L2框架?

二. V4L2的代码框架图:

三. V4L2代码开发流程:

四. 整个工程的运行流程:

最后:

分类专栏

- 音视频项目实战! [付费] 3篇
- 网络编程 4篇
- Linux内核学习笔记! 4篇
- 音视频开发 4篇
- 面试和学习路线 5篇
- C++学习 24篇
- 数据结构和算法 4篇

```

for (unsigned int i = 0; i < buf_num; ++i)
{
    /*查询内核缓冲区信息*/
    struct v4l2_buffer v4l2_buf;
    memset(&v4l2_buf, 0, sizeof(v4l2_buf));
    v4l2_buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    v4l2_buf.memory = V4L2_MEMORY_MMAP;
    v4l2_buf.index = i;
    if (ioctl(fd, VIDIOC_QUERYBUF, &v4l2_buf) < 0)
    {
        printf("VIDIOC_QUERYBUF failed\n");
        return -1;
    }

    usr_buf[i].length = v4l2_buf.length;
    usr_buf[i].start = (char *)mmap(0, v4l2_buf.length, PROT_READ | PROT_WRITE, MAP_SHARED, fd, v4l2_buf.m.offset);
    if (MAP_FAILED == usr_buf[i].start)
    {
        printf("mmap failed: %d\n", i);
        return -1;
    }
}
else
{
    if (ioctl(fd, VIDIOC_QBUF, &v4l2_buf) < 0)
    {
        printf("VIDIOC_QBUF failed\n");
        return -1;
    }
}
}
return 0;
}

```

利用ioctl控制VIDIOC_REQBUFS，进行视频缓冲区的申请。申请完成之后，把驱动的视频缓冲区映射到用户空间，映射的api使用的是mmap，若映射用户空间成功的话，则把视频数据入到缓冲区队列，入队对应的ioctl命令是VIDIOC_QBUF。

3.5. 开启摄像头获取视频流

```

int stream_on()
{
    enum v4l2_buf_type type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    if (ioctl(fd, VIDIOC_STREAMON, &type) < 0)
    {
        printf("VIDIOC_STREAMON failed\n");
        return -1;
    }
    return 0;
}

```

使用stream_on指令使能v4l2摄像头，让它正常启动采集流程。

3.6. 从缓冲区中把视频数据取出

```

int write_video_frame()
{
    struct v4l2_buffer v4l2_buf;
    v4l2_buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    v4l2_buf.memory = V4L2_MEMORY_MMAP;
    if (ioctl(fd, VIDIOC_DQBUF, &v4l2_buf) < 0) // 内核缓冲区出队列
    {
        printf("VIDIOC_DQBUF failed, dropped frame\n");
        return -1;
    }
    char buffer[256];
    sprintf(buffer, "%d.mjpg", v4l2_buf.index);
    int file_fd = open(buffer, O_RDWR | O_CREAT); // 若打开失败则不存储该帧图像
    if (file_fd > 0)
    {
        printf("saving %d images\n", v4l2_buf.index);
        write(file_fd, usr_buf[v4l2_buf.index].start, v4l2_buf.bytesused);
        close(file_fd);
    }

    if (ioctl(fd, VIDIOC_QBUF, &v4l2_buf) < 0) // 缓冲区重新入队
    {
        printf("VIDIOC_QBUF failed, dropped frame\n");
        return -1;
    }
    return v4l2_buf.index;
}

```

利用ioctl把缓冲区的视频数据取出，它对应的命令是VIDIOC_DQBUF。使用了VIDIOC_DQBUF之后，v4l2_buffer结构体就有对应的视频数据了。并把对应的数据写到mjpg图片。做完上述所有操作之后，再把摄像头的数据进行入队操作VIDIOC_QBUF。

3.7. 关闭摄像头获取流

```

int camera_stream_off()
{
    /*关闭视频流*/
    enum v4l2_buf_type type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    if (ioctl(fd, VIDIOC_STREAMOFF, &type) == -1)
    {
        printf("Fail to ioctl VIDIOC_STREAMOFF");
        return -1;
    }
    return 0;
}

```

通过ioctl把指令VIDIOC_STREAMOFF，关闭摄像头采集工作。

3.8. 解除映射缓冲区

```

int unmap_v4l2_buffer()
{
    /*解除内核缓冲区到用户缓冲区的映射*/
    for (unsigned int i = 0; i < buf_num; i++)
    {
        int ret = munmap(usr_buf[i].start, usr_buf[i].length);
        if (ret < 0)
        {
            printf("munmap failed\n");
            return -1;
        }
    }
    free(usr_buf); // 释放用户缓冲区内存
    return 0;
}

```

使用munmap api解决内核缓冲区到用户缓冲区的映射操作。

3.9. 释放所有的资源

```

void release_camera()
{
    close(fd);
}

```

四. 整个工程的运行流程:

```

int main(int argc, char *argv[])
{
    int ret = init_camera("/dev/video0");
    if (ret < 0)
    {
        printf("init_camera error\n");
        return -1;
    }

    ret = mmap_v4l2_buffer();
    if (ret < 0)
    {
        printf("mmap_buffer error\n");
        return -1;
    }

    ret = camera_stream_on();
    if (ret < 0)
    {
        printf("stream_on error\n");
        return -1;
    }

    for (int i = 0; i < 20; i++)
    {
        write_frame();
    }

    ret = camera_stream_off();
    if (ret < 0)
    {
        printf("stream_off error\n");
        return -1;
    }

    ret = unmap_v4l2_buffer();
    if (ret < 0)
    {
        printf("unmap_buffer error\n");
        return -1;
    }
}

```

```
{
    printf("unmap_buffer error\n");
    return -1;
}

release_camera();

return 0;
}
```

最后：

对嵌入式音频开发感兴趣的朋友，可以加入星球来学习：



最新视频教程更新到：

音视频...

物体识别算法模型代码讲解

音视频编解码基础知识和rv1126推流环境搭建

h264编解码讲解和rtmp推流开发环境搭建和工程演示!

h264码流结构解析(1)和srt服务器搭建

h264码流结构解析(2)

H264码流结构解析-3

H265讲解

VENC编码讲解

音频基础讲解(1)

音频基础讲解(2)-AAC讲解

音频基础讲解(3)-RV1126采集AENC

封装格式之TS数据流讲解

FLV数据类型讲解

最后感谢大家的支持哈，加入了星球的朋友，记得加我微信：lu18879499804,我拉大家进群答疑！

您 文章知识点与官方知识档案匹配，可进一步学习相关知识

云原生入门技能树 > 首页 > 概述 8534 人正在系统学习中

Linux ARM平台开发系列讲解（摄像头V4L2子系统） 2.12.7 摄像头V4L2驱动开发流程总结 DSMGUOGUO的博文 1963

1.概述前几章对摄像头V4L2驱动开发各个环节做了展开分析，这一章书就做一个汇总，用来回顾前几章得知识点，概括一下摄像头开发得流程，本次使用...

V4L2摄像头驱动程序在ARM9平台上程序实现 09-13

摄像头V4L2编程应用开发 weixin_46571142的博文 673

1.V4L2简介 Video for Linux two(Video4Linux2)简称V4L2，是V4L的改进版。V4L2是linux操作系统下一套用于采集图片、视频、和音频数据的通用...

V4L2驱动开发详解 x-2010的笔记 4338

环境：OS： Ubuntu 16.04 (Win10 hyperv) Kernel Version： 3.13.0-24-generic 这里终极目标是注册一个DevVideo0的设备，再通过一个应用程序去读取...

Linux ARM平台开发系列讲解（摄像头V4L2子系统） 2.12.4 V4L2子设备操作函数结构体分析 DSMGUOGUO的博文 802

对于子设备，其核心就是去实现子设备操作接口，下面就介绍操作函数接口用到的一些关键成员是什么意思。子设备，负责实现具体的功能。可将其抽象...

linux 帧缓存 v4l2_基于V4L2的视频驱动开发（2） weixin_35912365的博文 733

基于V4L2 的视频驱动开发(2) 作者：刘清涛，华清远见嵌入式学院 讲师。 三、 V4L2 API 及数据结构 V4L2 是 V4L 的升级版本，为 linux 下视频设备程序...

linux之V4L2摄像头应用流程 danielming的博文 311

对于v4l2，上次是在调试收音机驱动的时候用过，其他也就只是用2c配置一些寄存器就可以了，那时只是粗略的了解了，把收音机当作v4l2的设备后会包...

V4L2视频采集的基本流程 anghrcu2000的博文 1406

V4L2视频采集的基本流程 嵌入式Linux使用视频驱动V4L2 (Video For Linux Two) 来进行视频采集、输出。本文就V4L2的使用方式做简要说明，视频...

八、V4L2 ioctl 控制接口 调用流程 zzsky的博文 769

V4L2 ioctl 控制接口 调用流程 当 使用v4l2打开配置好Video节点后，还可能需要对设备进行参数 sensor 或者ISP的参数配置，例如自动对焦、自动曝光、...

和菜鸟一起学习linux之V4L2摄像头应用流程 weixin_42632472的博文 1237

一、概述 Video for Linuxtwo(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...

V4L2开发应用流程的各类组实用VIDIOC命令及其结构体集锦 Mark_minGE的博文 4270

本人作为初入音视频领域的新人，观摩了各位大佬关于V4L2详细开发流程的满满干货，特意为这两周的学习做个总结，希望后来者能顺利完成关于V4L2...

V4l2视频输出实现流程 h15278208355的专栏 3078

实现功能 设备读取摄像头传感器的数据，通过UVC协议传给上位机。同时，上位机发送控制命令给设备侧。 参考源码：https://github.com/whle/lvvc-ga...

V4L2-框架 h_8410435的博文 2850

1.概述 V4L2 是专门为linux 设备设计的一套视频框架，其主体框架在linux内核，可以理解为是整个linux系统上面的视频源捕捉驱动框架。相机驱动层位于...

V4L2编程之USB摄像头采集jpeg图像 爱学习的凌晨铁锤的博文 1462

V4L2编程实践 开发环境 虚拟机Ubuntu 16.04 编辑器VsCode 交叉编译工具 arm-linux-gnueabihf 已制作文件系统，已使能UVC相关驱动 正点原子ZYNG7...

V4L2采集图像流程 Eraser的博文 250

简介 V4L2是Video for linux2的简称,为linux中关于视频设备的内核驱动。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...

linux v4l2 示例程序_Linux关于Camera使用V4L2 weixin_35817967的博文 325

简介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，工作表程打开设...

音视频开发—linux下V4L2拍照 辨别熊瞎子的博文 273

目录 1. 流程说明 2. 代码说明 2.1摄像头初始化 2.2 启动摄像头 2.3拍照 2.4停止摄像头 3. 参考链接 V4L2(Video4linux2)是linux中关于视频设备的内核驱...

第二十讲：神州路由器静态路由的配置 最新发布 weixin_41687096的博文 263

csgo服务器搭建（linux）-极客云 织雪的博文 667

查看csgo服务器中workshop里每个地图文件的大小，命令：du -h --max-depth=0 /home/steam/csgo_server/csgo/maps/workshop/搭建CSGO服务器比...

“相关推荐”对你有帮助么？

非常没帮助

没帮助

一般

有帮助

非常有帮助

©2022 CSDN 皮肤主题：大白 设计师：CSDN官方博客 返回首页

关于我们 招聘骑士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

txp玩Linux 关注

10 0 0 0 0 0 0

txp玩Linux 码龄5年 暂无认证

113 1万+ 1万+ 7万+ 等级

1519 119 66 81 285

积分 粉丝 获赞 评论 收藏

私信 关注

搜博主文章

热门文章

作为面试官的你，是不是也有这种疑惑和烦恼！ 10890

ffmpeg第一弹：ffmpeg介绍和开发环境搭建！ 8054

困扰很多人ubuntu连网问题，终于搞定了！ 7451

得动，使用ers进行webnc推流体验！ 3361

简单的状态机入门！ 2545

最新评论

得动，使用ers进行webnc推流体验！ 苏林牧之：怎么配置https+域名的方式？

得动，使用ers进行webnc推流体验！ niktang：可以使用his生成m3u8,后面自己合并mp4

得动，使用ers进行webnc推流体验！ weixin_37849325：请问nc推流怎么录制？

全网最全的音视频书籍推荐！ 春起来春起来：ffmpeg+SDL制作播放器的项目，麻烦学长再分享一下雷神u3站视频配置...

简单的状态机入门！ 陆道helloworld1：这应该是简单的状态机的实现，是状态机

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

select多路复用学习(1)

音视频编解码经典问题汇总(1)

FFmpeg读取本地文件推流到RTMP服务器！

2022年 34篇 2021年 24篇 2020年 55篇 2019年 1篇

