#include <stdio.h>

#include <string.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/ioctl.h>

#include <linux/videodev2.h>

#include <stdlib.h>

#include <sys/mman.h>

#include <errno.h>

struct cam\_buf{

void \*start;

size\_t length;

};

struct cam\_buf bufs[4];

struct v4l2\_requestbuffers reqbufs;

### 小车摄像头的初始化

int camera\_init(char \*devpath, unsigned int \*width, unsigned int \*height, unsigned int \*size, unsigned int \*ismjpg)

{

int ret;

unsigned int i;

struct v4l2\_capability capability;

struct v4l2\_format format;

struct v4l2\_buffer vbuf;

**打开设备**

int camerafd = open(devpath, O\_RDWR | O\_NONBLOCK, 0);

if(0 > camerafd)

{

perror("open failed!\n");

return -1;

}

**查看支持的驱动**

ret = ioctl(camerafd, VIDIOC\_QUERYCAP, &capability);

if(0 > ret)

{

perror("camera init failed!\n");

return -1;

}

**判断设备是否支持视频采集**

if(!(capability.capabilities & V4L2\_CAP\_VIDEO\_CAPTURE))

{

perror("device can not support V4L2\_CAP\_VIDEO\_CAPTURE!\n");

close(camerafd);

return -1;

}

**判断设备是否支持视频流采集**

if(!(capability.capabilities & V4L2\_CAP\_STREAMING))

{

perror("device can not support V4L2\_CAP\_STREAMING!\n");

close(camerafd);

return -1;

}

**设置捕获的视频格式 MYJPEG**

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

format.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

format.fmt.pix.pixelformat = V4L2\_PIX\_FMT\_MJPEG;

format.fmt.pix.width = \*width;

format.fmt.pix.height = \*height;

format.fmt.pix.field = V4L2\_FIELD\_ANY;

ret = ioctl(camerafd, VIDIOC\_S\_FMT, &format);

if(0 > ret)

{

perror("camera init failed !\n");

return -1;

}

else

{

\*ismjpg = 0;

fprintf(stdout, "picture format is mjpeg\n");

goto get\_fmt;

}

get\_fmt:

ret = ioctl(camerafd, VIDIOC\_G\_FMT, &format);

if(0 > ret)

{

perror("camera init\n");

return -1;

}

**向驱动申请缓存**

memset(&reqbufs, 0, sizeof(struct v4l2\_requestbuffers));

reqbufs.count = 4;

reqbufs.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

reqbufs.memory = V4L2\_MEMORY\_MMAP;

ret = ioctl(camerafd, VIDIOC\_REQBUFS, &reqbufs);

if(0 > ret)

{

perror("camera init\n");

close(camerafd);

return -1;

}

**循环映射并入队**

for (i = 0; i < reqbufs.count; i++) {

**真正获取缓存的地址大小**

memset(&vbuf, 0, sizeof(struct v4l2\_buffer));

vbuf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

vbuf.memory = V4L2\_MEMORY\_MMAP;

vbuf.index = i;

ret = ioctl(camerafd, VIDIOC\_QUERYBUF, &vbuf);

if(0 > ret)

{

perror("camera init\n");

close(camerafd);

return -1;

}

**映射缓存到用户空间，通过mmap将内核的缓存地址映射到用户空间，并和文件描述符相关联**

bufs[i].length = vbuf.length;

bufs[i].start = mmap(NULL, vbuf.length, PROT\_READ | PROT\_WRITE, MAP\_SHARED, camerafd, vbuf.m.offset);

if(MAP\_FAILED == bufs[i].start)

{

perror("camera init\n");

close(camerafd);

return -1;

}

//每次映射都会入队，放入缓冲队列

vbuf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

vbuf.memory = V4L2\_MEMORY\_MMAP;

ret = ioctl(camerafd, VIDIOC\_QBUF, &vbuf);

if(0 > ret)

{

perror("camera init\n");

close(camerafd);

return -1;

}

}

**返回真正设置成功的宽、高、大小**

\*width = format.fmt.pix.width;

\*height = format.fmt.pix.height;

\*size = bufs[0].length;

return camerafd;

}