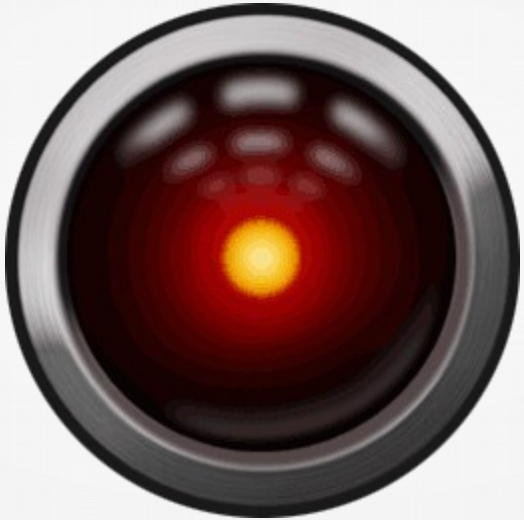
Image/Video capture with V4L2



Who am I

https://github.com/Tropicao

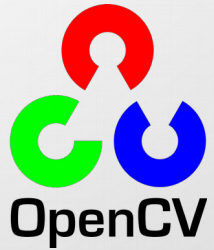
- Alexis Lothoré, embedded software engineer
  - 2016 to 2018 : Smile ECS, Toulouse
  - 2018 to now : Somfy Protect By Myfox, Toulouse
- Main work fields :
  - Complete linux firmwares for embedded systems
  - Connected alarm systems : video streaming, connected systems, embedded security

Summary

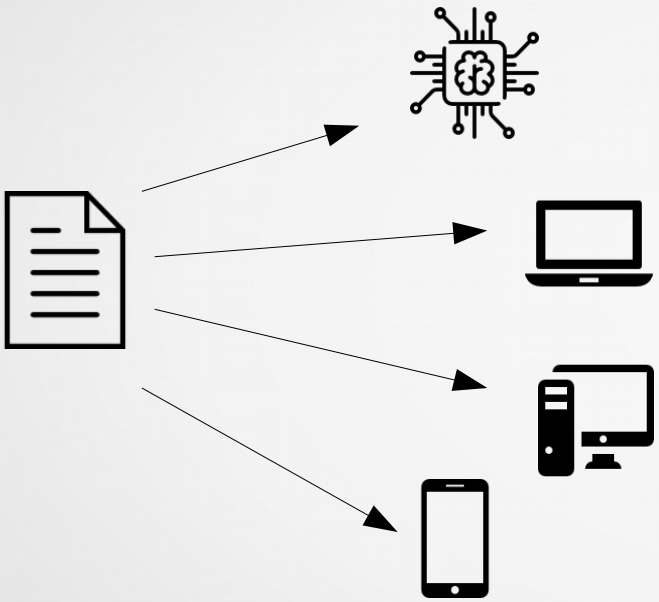
- What is V4L2
- Using V4L2 for video capture : initialization, stream loop
- Going further : encoding a raw image for image, for video

What is V4L2

- A standardized kernel ↔ userspace API, started from kernel 2.1.X for v1, then dropped and replaced by v2 in v2.5.x
- Originally supports various captures support (webcams, TV tuners, radio)
- Now supports a large set of features : input/output for audio/video, tuners, modulators, extended user controls, M2M, etc.
- This presentation focuses on video capture



Why should I use V4L2 ?



- One code base to rull them all !
- Can adapt to each platform at runtime (e.g. capabilities detection)
- But : you need proper V4L2 support (kernel and drivers)

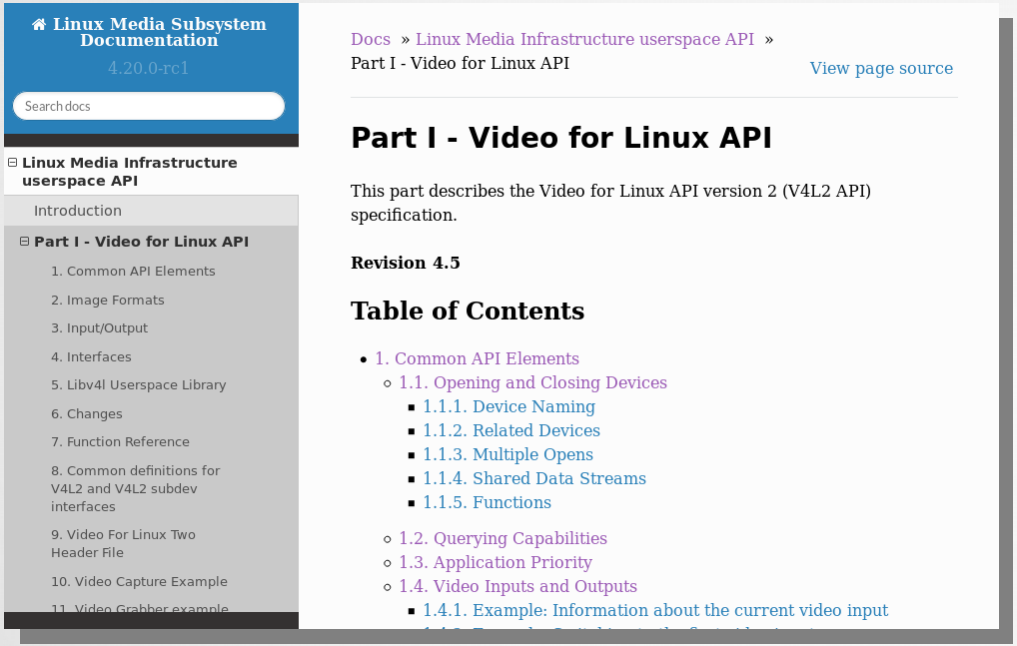
What do you need to begin

- Obviously, a video capture device
- A V4L2 capable linux-kernel
- Proper V4L2 driver for used device

```
[ 6877.094195] usb 1-6.2: new high-speed USB device number 11 using xhci_hcd
[ 6877.455368] usb 1-6.2: New USB device found, idVendor=046d, idProduct=0826, bcdDevice= 0.10
[ 6877.455372] usb 1-6.2: New USB device strings: Mfr=0, Product=2, SerialNumber=1
[ 6877.455375] usb 1-6.2: Product: HD Webcam C525
[ 6877.455377] usb 1-6.2: SerialNumber: B60A2F60
[ 6877.737575] usb 1-6.2: set resolution quirk: cval->res = 384
[ 6877.738303] uvcvideo: Found UVC 1.00 device HD Webcam C525 (046d:0826)
[ 6877.776277] uvcvideo 1-6.2:1.2: Entity type for entity Extension 5 was not initialized!
[ 6877.776280] uvcvideo 1-6.2:1.2: Entity type for entity Processing 2 was not initialized!
[ 6877.776281] uvcvideo 1-6.2:1.2: Entity type for entity Camera 1 was not initialized!
[ 6877.776283] uvcvideo 1-6.2:1.2: Entity type for entity Extension 6 was not initialized!
[ 6877.776285] uvcvideo 1-6.2:1.2: Entity type for entity Extension 7 was not initialized!
[ 6877.776286] uvcvideo 1-6.2:1.2: Entity type for entity Extension 8 was not initialized!
[ 6877.776353] input: HD Webcam C525 as /devices/pci0000:00/0000:00:14.0/usb1/1-6/1-6.2/1-6.2:1.2/input/input38
```

```
alexis@arch-alexis:~/|⇒  ls /dev/video*
/dev/video0  /dev/video1
```

Your new friend : the official V4L2 doc



https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html

- Can be used as an API documentation
- Can be used as a guide
- Userspace API, but also kernel API

The rest of the slides are heavily relying on this documentation

Discovering your device

```
struct v4l2_capability cap ;

dev_fd = open("/dev/video0", O_RDWR, 0) ;
if(ioctl(dev_fd, VIDIOC_QUERYCAP, &cap) == -1)
      fprintf(stderr, "Cannot fetch devices capabilities") ;

if (!(cap.capabilities & V4L2_CAP_VIDEO_CAPTURE))
      fprintf(stderr, "Selected device cannot capture video") ;
```

Configuring your device input

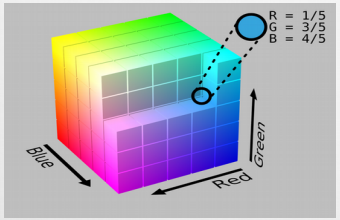
- setting the application priority : VIDIOC_S_PRIORITY
- querying / setting video input : VIDIOC_ENUMINPUT, VIDIOC_S_INPUT
- querying / setting video standards : VIDIOC_ENUMSTD
- cropping : VIDIOC_S_SELECTION
- setting streaming parameters VIDIOC_S_PARM
- and more...

User controls on image

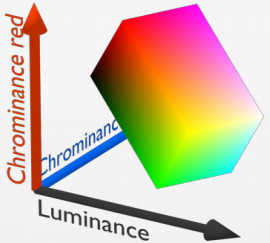
- A driver can expose many image controls (brightness, saturation, …)
- We can enumerate and expose those controls : VIDIOC_QUERYCTRL, VIDIOC_QUERY_EXT_CTRL
- Some controls can be menus : VIDIOC_QUERYMENU
- /!\ VIDIOC_QUERY_EXT_CTRL added later, check if supported by your kernel/driver !

The pixel format : quick summary

- Many different image formats are available, depends on drivers :
  - different colorspaces : RGB, HSL, YCbCr…
  - single plane or multi plane format
  - indexed formats
  - etc
- Each format is represented with a FourCC code



RGB format



YCbCr format

HSL format

Image format example : Bayer RGB

- FourCC code : ‘B’ ‘A’ ‘8’ ’1’
- Raw frame size : width * height
- Why more G than B and R ? Because human eye is more sensible to green !

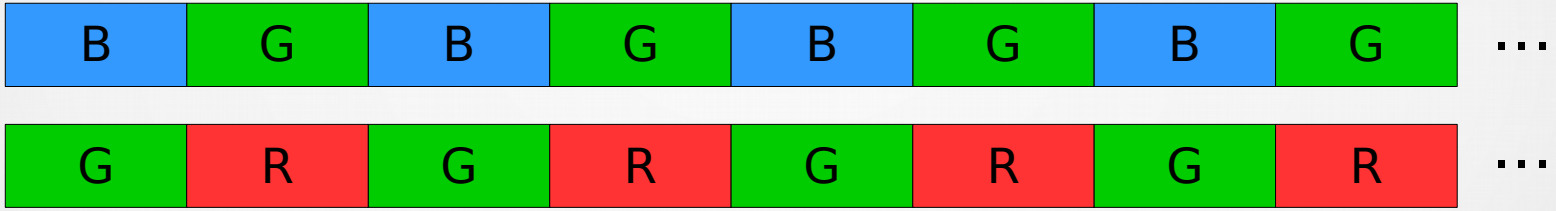
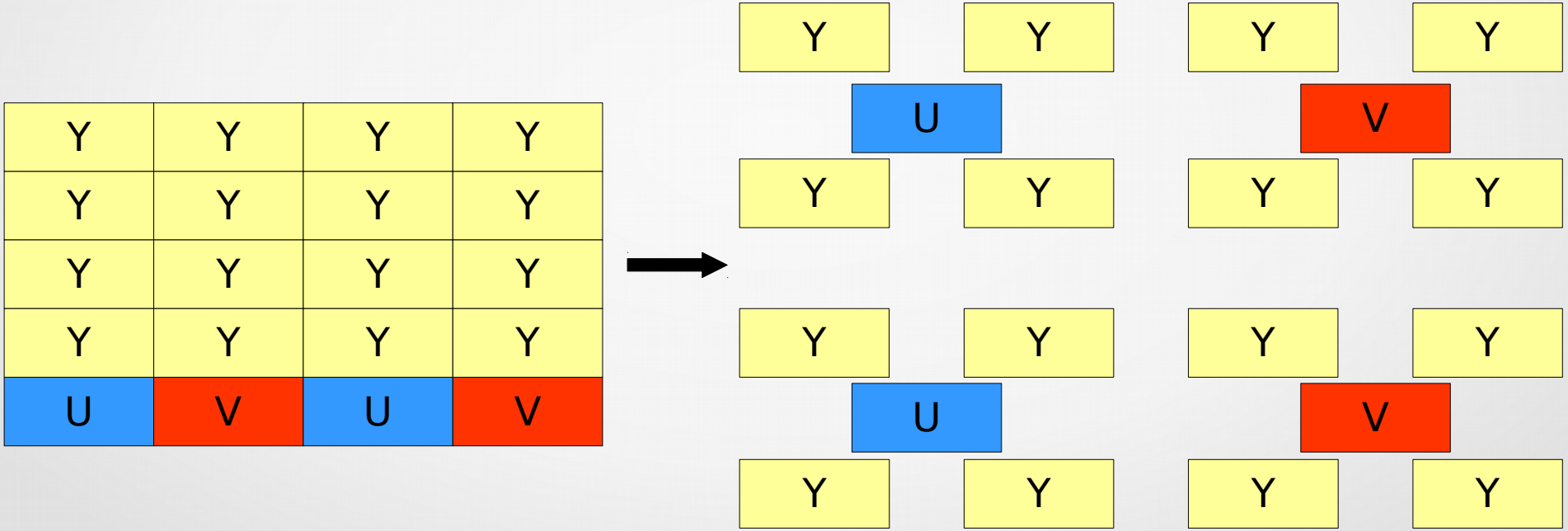


Image format example : YUV422, or YUY2

- FourCC code : ‘Y’ ‘U’ ‘Y’ ’V’
- Raw frame size : width * height * 2
- Why more Y than U and V ? Because most of the image information is in Luma !

Image format example : NV12, or YUV420SP

- FourCC code : ‘N’ ‘V’ ‘1’ ’2’
- Raw frame size : width * height * 3/2



Negotiating image format

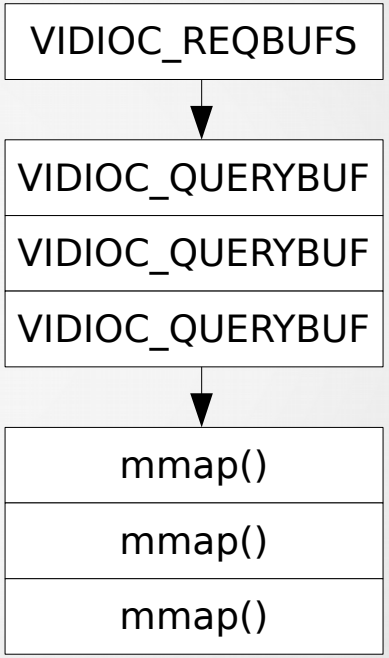
- Asking driver format capabilities : VIDIOC_ENUM_FMT
- Each format as a unique code, represented with 4 letters : FourCC code
- Selecting a format : VIDIOC_S_FMT
  - fill the v4l2_format structure according to your needs
  - format is not expressed the same way between APIs (single vs multiplanar)
  - remember to check what the driver has really taken !

Choosing a capture method

- Three way of retrieving raw frames :
  - basic I/O : read/write directly from/to device (remember to check for device capabilities : V4L2_CAP_READWRITE !)
  - streaming (needs V4L2_CAP_STREAMING !) :
    - memory mapping : request, mmap, enqueue and dequeue buffers
    - user pointers
    - DMA buffers (to export data between devices)

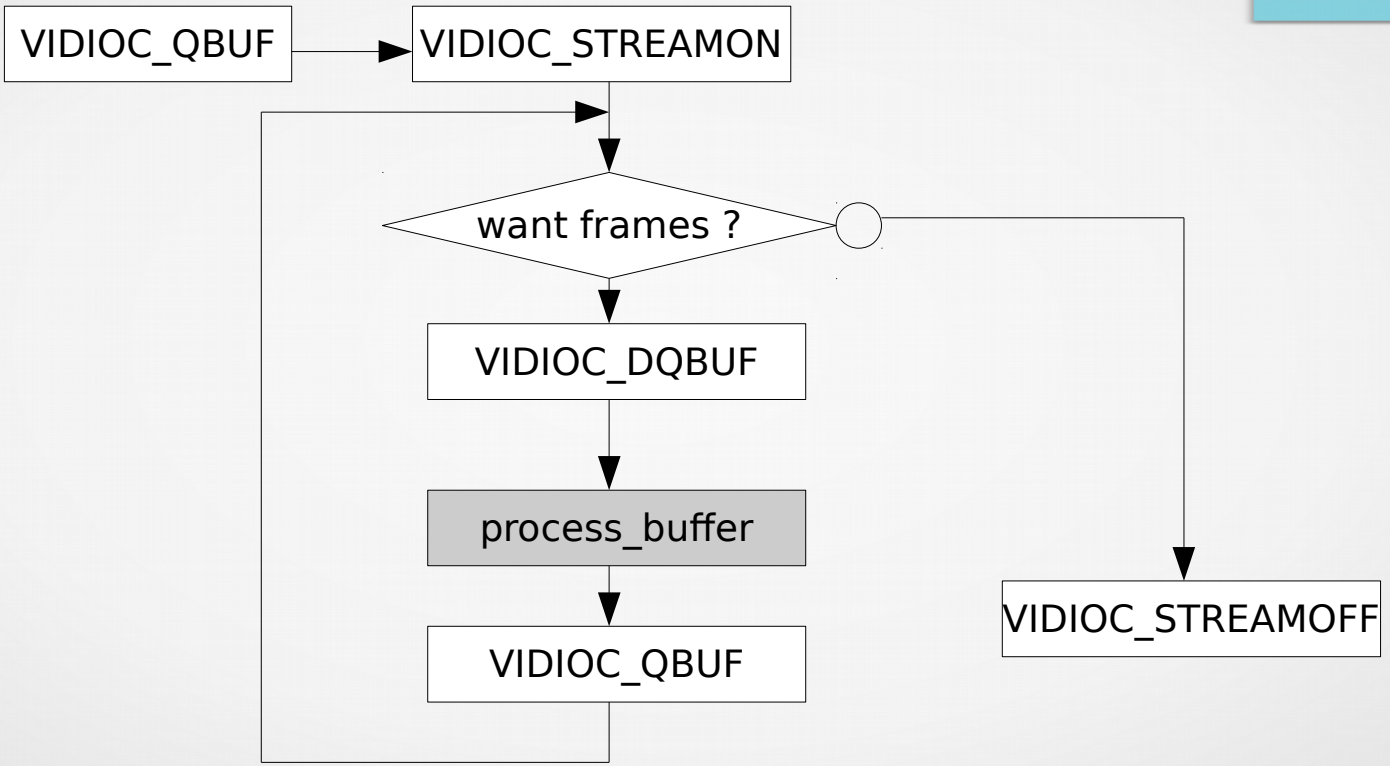
Focusing on MMAP buffers

- Allocate buffers on driver side(set number of buffers, and type V4L2_MEMORY_MMAP
- For each buffer, query length and location
- mmap the buffer



=> How many buffers do I need ???

The capture loop



Using raw images in real world ?

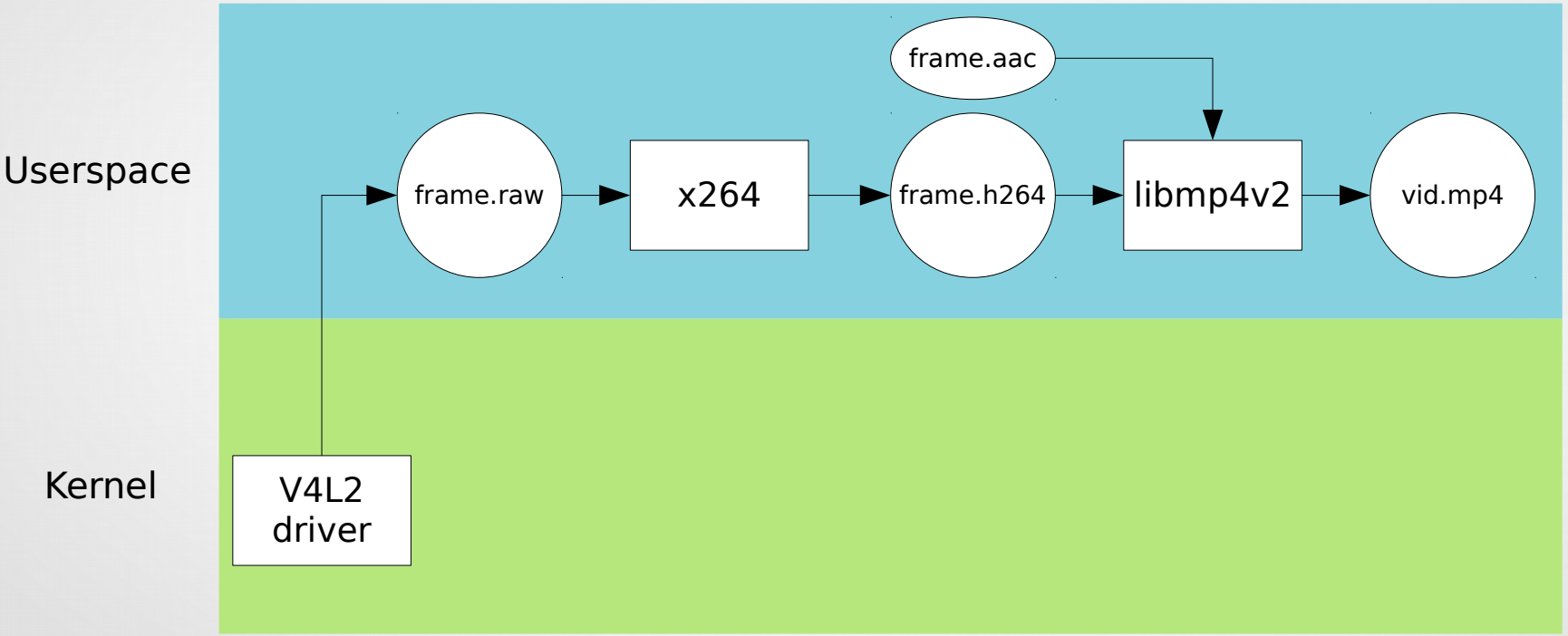
Displaying raw images

http://www.rawpixels.net/

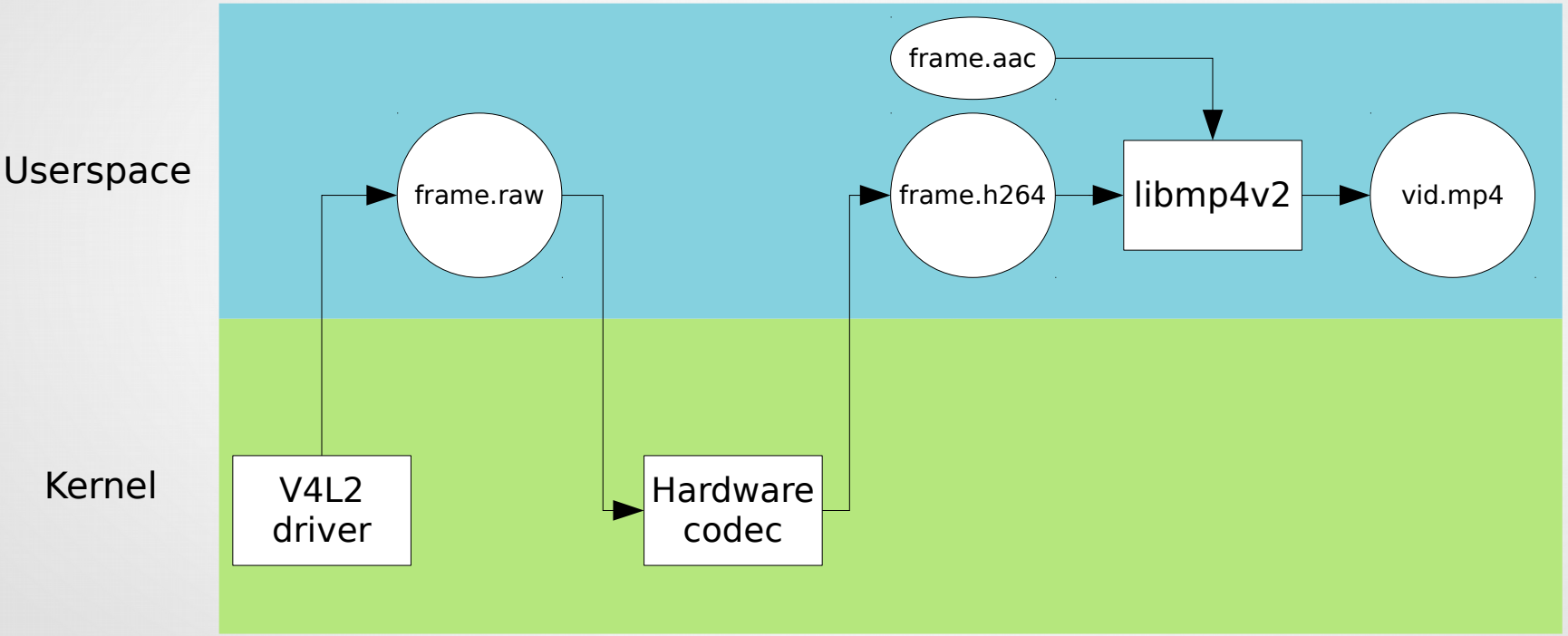
Encoding images :

we can pass raw RGB or YUV images to a JPEG encoder (software encoder, like jpeglib, or even a hardware JPEG encoder)

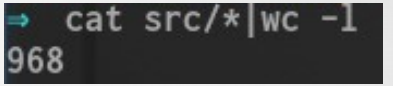
Exploiting raw frames for video



Exploiting raw frames for video



Summary

- Let’s face it : it was quite fastidious, just to get images… (

)
- And we have just reinvented part of the wheel : see **v4l2-ctl, v4l2ucp**
- see also : ***libv4lconvert***, ***libv4l***, ***libv4l2***
- ***libcamera***, made to replace V4L2 ?

Summary

- Having proper kernel support is sometimes a big assumption
  - Unimplemented ioctls ?
  - Unexpected behavior ?
- But we have a generic code which can run on multiple platforms !
- We have only discussed a very little part of the framework !

Sources

- https://fr.wikipedia.org/wiki/Video4Linux
- http://www.latelierducable.com/tv-televiseur/yuv-420-ycbcr-422-rgb-444-cest-quoi-le-chroma-subsampling/
- https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html
- http://linuxgizmos.com/libcamera-successor-to-v4l2-hopes-to-ease-embedded-linux-camera-headaches/