v4l2ctl

Release 0.1a5

Feb 28, 2021

Contents:

CHAPTER 1

What v4l2ctl is

v4l2ctl is a python package to control V4L2 (Video 4 Linux 2) drivers.

CHAPTER 2

Project Status

The project is still in its early development stages. It can, however, already be used to read the name, driver, version and capabilities of a v4l2 device driver.

CHAPTER 3

The documentation

3.1 v4l2ctl package

3.1.1 Module contents

class v4l2ctl.**V4l2Device**(*device='/dev/video0'*)

Initialize the V4l2Device object and read its basic information.

Keyword Arguments **device** (*str, path-like, int*) – the video device (default r"/dev/video0") if an int is given, it is assumed to be number after "video" in "/dev".

Raises *IoctlError* – if a non-video device file is given.

buffer_type

The buffer type (see *V4l2BufferType*) required for several operations. This attribute does not change anything in the device itself. It is used by other operations.

bus

The bus through which this device is connected (read-only).

capabilities

The device specific capabilities (read-only). These are the capabilities associated with this dev-file only. The physical device can have more than one dev-file, and hence more capabilities. See physical_capabilities.

close()

Flush and close the IO object.

This method has no effect if the file is already closed.

cropping_capabilities

The cropping capabilities (read-only). These are the cropping capabilities of this video device.

Only valid for these buffer types:

- V4l2BufferType.VIDEO_CAPTURE

- V4l2BufferType.VIDEO_CAPTURE_MPLANE
- V4l2BufferType.VIDEO_OUTPUT
- V4l2BufferType.VIDEO_OUTPUT_MPLANE
- V4l2BufferType.VIDEO_OVERLAY

cropping_rectangle
The cropping rectangle (see `V4l2Rectangle`).

Note: The cropping rectange is specfic to the set buffer type. (See *buffer_type*)

---

device
The device file (read-only).

driver
The linux driver (read-only).

fileno()
Returns underlying file descriptor if one exists.

OSError is raised if the IO object does not use a file descriptor.

flush()
Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

formats
A generator for the suported formats by this video device.

Note: The formats are specfic to the set buffer type. (See *buffer_type*)

---

isatty()
Return whether this is an 'interactive' stream.

Return False if it can't be determined.

iter_buffer_formats(*buffer_type*)
Iterate over the formats supported by a certain buffer.

Keyword Arguments **buffer_type** – see *V4l2BufferType*.

Returns a generator

static iter_devices(*skip_links=True*)
Return an iterator over the available v4l2 devices.

Keyword Arguments **skip_links** (*bool*) – skip links and return every device only once (default True)

Returns an iterator

name
The card name (read-only).

physical_capabilities
The general physical capabilities (read-only). These are the capabilities associated with the physical device as a while, and not limited to this dev-file only.

readable()
Return whether object was opened for reading.

If False, read() will raise OSError.

readline(*size=-1*)
Read and return a line from the stream.

If size is specified, at most size bytes will be read.

The line terminator is always b'n' for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

readlines(*hint=-1*)
Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

seekable()
Return whether object supports random access.

If False, seek(), tell() and truncate() will raise OSError. This method may need to do a test seek().

supported_buffer_types
The supported buffer types by this video device (read-only).

version
The kernel version as a string (read-only).

version_tuple
The kernel version as a tuple (read-only).

writable()
Return whether object was opened for writing.

If False, write() will raise OSError.

writelines(*lines*)
Write a list of lines to stream.

Line separators are not added, so it is usual for each of the lines provided to have a line separator at the end.

class v4l2ctl.**V4l2Capabilities**
The v4l2 capability flags.

These are the flags defining the supported capabilities of a V4l2 devince.

Example

Check if device /dev/video0 supports video capturing:

```
vid_dev = VideoDevice(r"/dev/video0")
if CapabilityFlags.VIDEO_CAPTURE in vid_dev.capabilities:
    start_recording()
```

ASYNCIO = 33554432
Async I/O.

AUDIO = 131072
Has audio support.

DEVICE_CAPS = 2147483648
Sets device capabilities field.

EXT_PIX_FORMAT = 2097152
Supports the extended pixel format.

HW_FREQ_SEEK = 1024
Can do hardware frequency seek.

META_CAPTURE = 8388608
Is a metadata capture device.

META_OUTPUT = 134217728
Is a metadata output device.

MODULATOR = 524288
Has a modulator.

RADIO = 262144
Is a radio device.

RDS_CAPTURE = 256
RDS data capture.

RDS_OUTPUT = 2048
Is an RDS encoder.

READWRITE = 16777216
Read/write systemcalls.

SDR_CAPTURE = 1048576
Is a SDR capture device.

SDR_OUTPUT = 4194304
Is a SDR output device.

SLICED_VBI_CAPTURE = 64
Is a sliced VBI capture device.

SLICED_VBI_OUTPUT = 128
Is a sliced VBI output device.

STREAMING = 67108864
Streaming I/O ioctls.

TOUCH = 268435456
Is a touch device.

TUNER = 65536
Has a tuner.

VBI_CAPTURE = 16
Is a raw VBI capture device.

VBI_OUTPUT = 32
Is a raw VBI output device.

VIDEO_CAPTURE = 1
Is a video capture device.

VIDEO_CAPTURE_MPLANE = 4096
Is a video capture device that supports multiplanar formats.

VIDEO_M2M = 32768
Is a video mem-to-mem device.

VIDEO_M2M_MPLANE = 16384
Is a video mem-to-mem device that supports multiplanar formats.

VIDEO_OUTPUT = 2
Is a video output device.

VIDEO_OUTPUT_MPLANE = 8192
Is a video output device that supports multiplanar formats.

VIDEO_OUTPUT_OVERLAY = 512
Can do video output overlay.

VIDEO_OVERLAY = 4
Can do video overlay.

class v4l2ctl.**V4l2BufferType**
The v4l2 buffer types.

Used with **:attribute:'enum_fmt'**.

META_CAPTURE = 13
Buffer for metadata capture, see Metadata Interface.

META_OUTPUT = 14
Buffer for metadata output, see Metadata Interface.

SDR_CAPTURE = 11
Buffer for Software Defined Radio (SDR) capture stream, see Software Defined Radio Interface (SDR).

SDR_OUTPUT = 12
Buffer for Software Defined Radio (SDR) output stream, see Software Defined Radio Interface (SDR).

SLICED_VBI_CAPTURE = 6
Buffer of a sliced VBI capture stream, see Sliced VBI Data Interface.

SLICED_VBI_OUTPUT = 7
Buffer of a sliced VBI output stream, see Sliced VBI Data Interface.

VBI_CAPTURE = 4
Buffer of a raw VBI capture stream, see Raw VBI Data Interface.

VBI_OUTPUT = 5
Buffer of a raw VBI output stream, see Raw VBI Data Interface.

VIDEO_CAPTURE = 1
Buffer of a single-planar video capture stream, see Video Capture Interface.

VIDEO_CAPTURE_MPLANE = 9
Buffer of a multi-planar video capture stream, see Video Capture Interface.

VIDEO_OUTPUT = 2
Buffer of a single-planar video output stream, see Video Output Interface.

VIDEO_OUTPUT_MPLANE = 10
Buffer of a multi-planar video output stream, see Video Output Interface.

VIDEO_OUTPUT_OVERLAY = 8
Buffer for video output overlay (OSD), see Video Output Overlay Interface.

VIDEO_OVERLAY = 3
Buffer for video overlay, see Video Overlay Interface.

class v4l2ctl.**V4l2Formats**
An Enum-Container for all V4l2Formats. This class delegates its operations to the contained enums. For more information, see:

py:class:*V4l2PixFormats* py:class:*V4l2SdrFormats* py:class:*V4l2TouchFormats* py:class:*V4l2MetaFormats*

class v4l2ctl.**V4l2FormatDescFlags**
The v4l2 format flags.

COMPRESSED = 1
This is a compressed format.

EMULATED = 2
This format is not native to the device but emulated through software (usually libv4l2), where possible try to use a native format instead for better performance.

NONE = 0
No flags are set.

exception v4l2ctl.**IoctlError**(*device*, *name*, *request*, *return_code*, *extra_msg=None*)
Raised when ioctl() returns a non-zero value.

exception v4l2ctl.**FeatureNotSupported**

CHAPTER 4

Internal documentation

4.1 Internal documentation

This is the documentation of the internals of v4l2ctl

4.1.1 Module ioctlmacros

4.1.2 Module v4l2interface

4.1.3 Module v4l2device

exception v4l2ctl.v4l2device.**FeatureNotSupported**

Bases: Exception

__module__ = 'v4l2ctl.v4l2device'

__weakref__
list of weak references to the object (if defined)

class v4l2ctl.v4l2device.**V4l2Device**(*device='/dev/video0'*)

Bases: io.IOBase

Initialize the V4l2Device object and read its basic information.

Keyword Arguments **device** (*str, path-like, int*) – the video device (default r"/dev/video0") if an int is given, it is assumed to be number after "video" in "/dev".

Raises IoctlError – if a non-video device file is given.

__abstractmethods__ = frozenset()

__enter__()

__exit__(*exc_type*, *exc*, *tb*)

__init__(*device='/dev/video0'*)
Initialize self. See help(type(self)) for accurate signature.

__iter__()
Implement iter(self).

__module__ = 'v4l2ctl.v4l2device'

__next__()
Implement next(self).

__repr__()
Return repr(self).

_abc_impl = <_abc_data object>

_open()

buffer_type
The buffer type (see `V4l2BufferType`) required for several operations. This attribute does not change anything in the device itself. It is used by other operations.

bus
The bus through which this device is connected (read-only).

capabilities
The device specific capabilities (read-only). These are the capabilities associated with this dev-file only. The physical device can have more than one dev-file, and hence more capabilities. See physical_capabilities.

close()
Flush and close the IO object.

This method has no effect if the file is already closed.

closed

cropping_buffer_types = [<V4l2BufferType.VIDEO_CAPTURE: 1>, <V4l2BufferType.VIDEO_CAPT

cropping_capabilities
The cropping capabilities (read-only). These are the cropping capabilities of this video device.

Only valid for these buffer types:

- V4l2BufferType.VIDEO_CAPTURE
- V4l2BufferType.VIDEO_CAPTURE_MPLANE
- V4l2BufferType.VIDEO_OUTPUT
- V4l2BufferType.VIDEO_OUTPUT_MPLANE
- V4l2BufferType.VIDEO_OVERLAY

cropping_rectangle
The cropping rectangle (see `V4l2Rectangle`).

Note: The cropping rectange is specfic to the set buffer type. (See *buffer_type*)

---

device
The device file (read-only).

driver
The linux driver (read-only).

fileno()
Returns underlying file descriptor if one exists.

OSError is raised if the IO object does not use a file descriptor.

flush()
Flush write buffers, if applicable.

This is not implemented for read-only and non-blocking streams.

formats
A generator for the suported formats by this video device.

Note: The formats are specfic to the set buffer type. (See *buffer_type*)

---

isatty()
Return whether this is an 'interactive' stream.

Return False if it can't be determined.

iter_buffer_formats(*buffer_type*)
Iterate over the formats supported by a certain buffer.

Keyword Arguments **buffer_type** – see `V4l2BufferType`.

Returns a generator

static iter_devices(*skip_links=True*)
Return an iterator over the available v4l2 devices.

Keyword Arguments **skip_links** (*bool*) – skip links and return every device only once (default True)

Returns an iterator

name
The card name (read-only).

physical_capabilities
The general physical capabilities (read-only). These are the capabilities associated with the physical device as a while, and not limited to this dev-file only.

readable()
Return whether object was opened for reading.

If False, read() will raise OSError.

readline(*size=-1*)
Read and return a line from the stream.

If size is specified, at most size bytes will be read.

The line terminator is always b'n' for binary files; for text files, the newlines argument to open can be used to select the line terminator(s) recognized.

readlines(*hint=-1*)
Return a list of lines from the stream.

hint can be specified to control the number of lines read: no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

seekable()
Return whether object supports random access.

If False, seek(), tell() and truncate() will raise OSError. This method may need to do a test seek().

supported_buffer_types
The supported buffer types by this video device (read-only).

version
The kernel version as a string (read-only).

version_tuple
The kernel version as a tuple (read-only).

writable()
Return whether object was opened for writing.

If False, write() will raise OSError.

writelines(*lines*)
Write a list of lines to stream.

Line separators are not added, so it is usual for each of the lines provided to have a line separator at the end.

class v4l2ctl.v4l2device.**V4l2DeviceIterator**(*skip_links*)
Bases: object

__dict__ = mappingproxy({'__module__': 'v4l2ctl.v4l2device', '_v4l2_device_prefixes':

__init__(*skip_links*)
Initialize self. See help(type(self)) for accurate signature.

__iter__()

__module__ = 'v4l2ctl.v4l2device'

__weakref__
list of weak references to the object (if defined)

_v4l2_device_prefixes = ['video', 'radio', 'vbi', 'swradio', 'v4l-subdev']

Or check the website's genindex.

CHAPTER 5

Copyright and licence

5.1 See also

If your purpose is to implement high end streaming applications or some sort of video processing, you might want to take a look at gstreamer and opencv.

Python Module Index

v

Index

Symbols

A

B

C

D

W