V4L2 vivid 仮想ビデオドライバの活用
～V4L2アプリケーション開発のために～

OSAKA NDS Embedded Linux Cross Online Forum #11

(株)大阪エヌデーエス 秋山和慶

1人の満足から、社会の満足へ
株式会社大阪エヌデーエス

自己紹介

- 秋山 和慶
  - 2017年 株式会社大阪エヌデーエス～現在
  - Linux上での開発に従事

目次

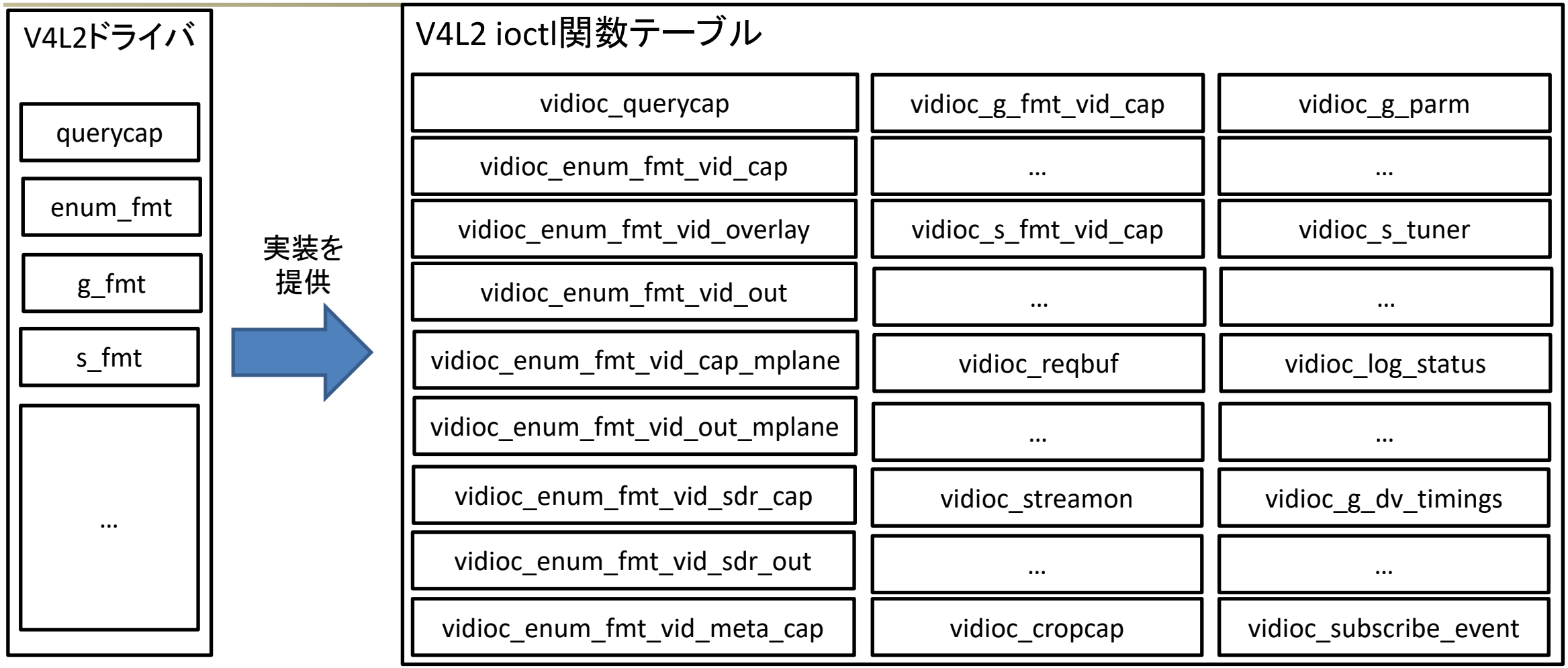
- V4L2について
- V4L2 Application開発上のよくある問題
- Vivid仮想ビデオドライバ

V4L2(Video For Linux 2)とは

- 映像/ラジオ等を扱うためのフレームワーク
  - カメラ、TVチューナ、ラジオ(AM/FM, Software Defined Radio(SDR))、コーデック etc… 様々なタイプのデバイスをサポート
- V4L2はデバイス種別(カメラ、ラジオ etc…)ごとに備えるべきインターフェース(read, write, ioctl etc…)を規定
  - 例えば、
    - カメラ入力ならVideo Capture Interface
      - デバイスファイル名 : /dev/videoX
    - 映像出力デバイスならVideo Output Interface
      - デバイスファイル名 : /dev/videoX
    - アナログラジオならRadio Interface
      - デバイスファイル名 : /dev/radioX

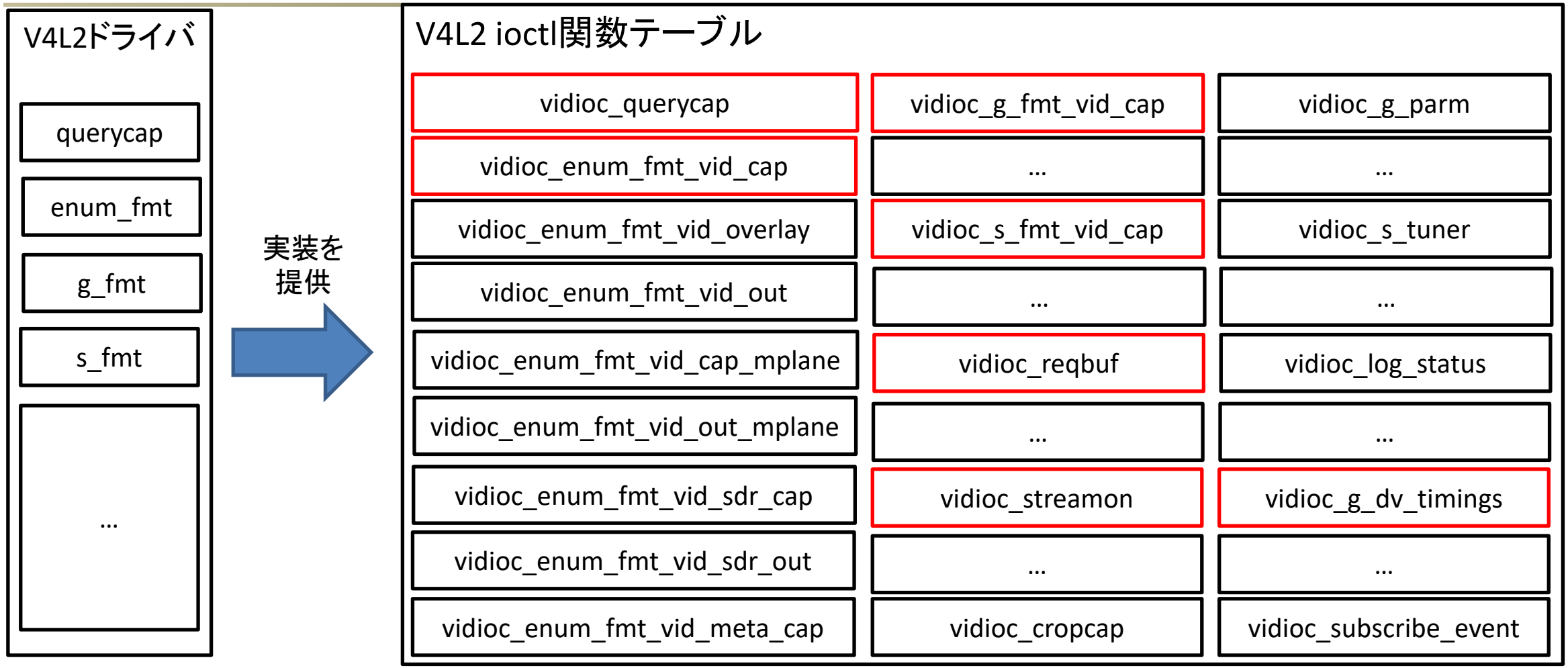
    etc…

V4L2(Video For Linux 2)とは



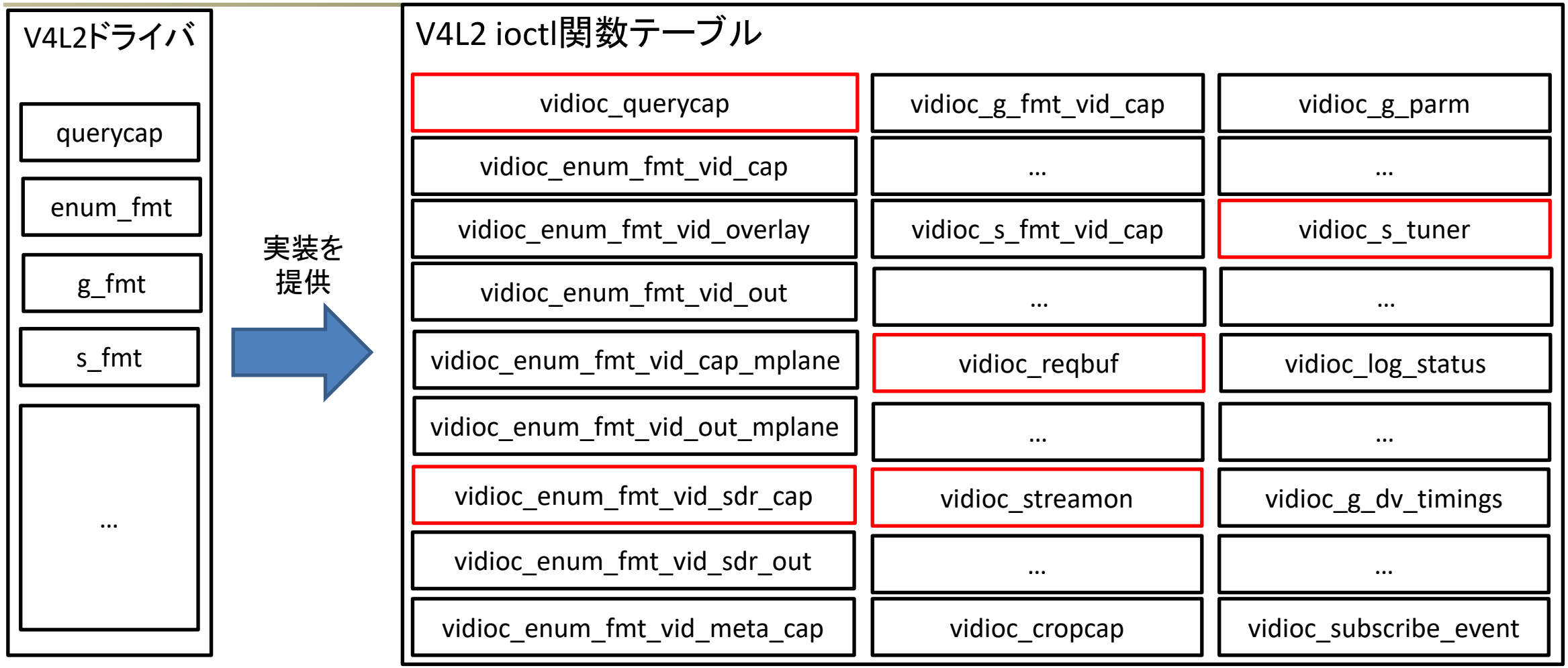
❖ インターフェース規定=ioctl関数テーブルの中で実装すべき関数についての規定

V4L2(Video For Linux 2)とは



❖ インターフェース規定=ioctl関数テーブルの中で実装すべき関数についての規定

V4L2(Video For Linux 2)とは

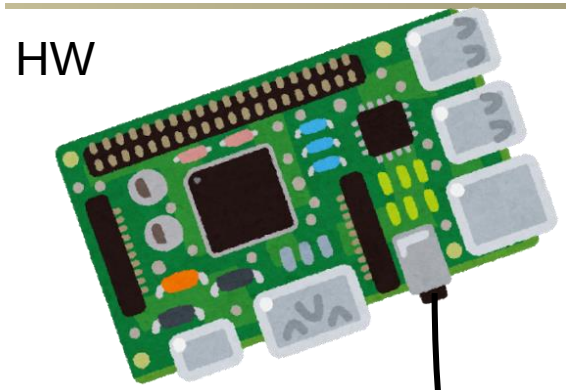


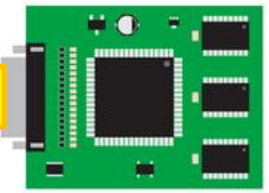
❖ インターフェース規定=ioctl関数テーブルの中で実装すべき関数についての規定

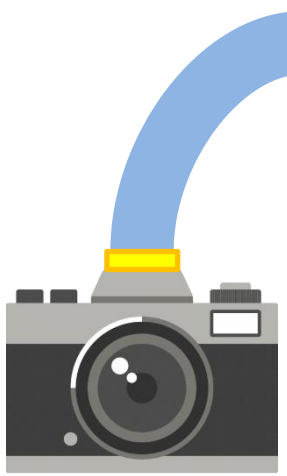
V4L2(Video For Linux 2)とは

- 映像経路上には複数のデバイスが存在することが多い
  - カメラ + エンコーダ etc...
- Applicationが映像の入出力として使用するのはV4L2 デバイス(ドライバ)のみ
  - Video Capture I/Fを実装したデバイスドライバ
  - Radio Receivcer I/Fを実装したデバイスドライバ etc...
- それ以外のデバイスはV4L2 Subdevice Interfaceを備えるV4L2サブデバイス(ドライバ)として実装
  - V4L2 Subdev Interface ≠ V4L2 Interface規定
  - V4L2デバイスはV4L2 Subdev Interface経由で情報取得/Subdev制御
  - (SubdeviceをApplicationから直接制御することも可能)
    - V4L2 Subdev Userspace API: https://www.kernel.org/doc/html/v4.14/media/kapi/v4l2-subdev.html#v4l2-sub-device-userspace-api

V4L2(Video For Linux 2)とは







HW

SW

Application

映像・APIを提供

/dev/v4l-subdev0

/dev/video0

V4L2
Subdev I/F

Video
Capture I/F

実装

Encoder Drv

データ(映像のサイズ・色等)やSubdevice
制御のためのAPIを提供

実装

Camera Drv

V4L2 Application開発上のよくある問題

- 機材が確保できない
  - カメラ等の専用機材が必要だが数が少ない
  - カメラ自体が開発中でごく少数の試作機しかない
  - クリーンルームから出せない
- カーネル側の開発中はアプリケーション側の実装がなかなか進まない/進めにくい
  - Video Deviceだけではなく経路上のSubdevice全ての開発が必要
- 仮想環境を利用したりして、

  アプリケーションをボード・PC上両方で開発できるようにしたい

  → PCとボードで同じペリフェラルを使うのが難しい(PCにカメラを接続できない等)

  → PCとボードでドライバを分ける？ PCで映像を扱うのを諦める？

Vivid 仮想ビデオドライバ

- Vivid (drivers/media/platform/vivid)
  - V4L2のテストのために作成されたドライバ
    - 特にインターフェース規定のテストに使用される
  - 様々な種類のデバイスをエミュレート
    - Video Capture/Output Interface
    - VBI Capture/Output Interface
    - Radio RX/TX Interface
    - Software Defined Radi(SDR) RX Interface
  - ハードウェア的な実態のない仮想ドライバ。Subdeviceなしで単独で動作する
  - 標準的なRGB, YUVフォーマットをサポート
  - 4Kカメラのエミュレートも可能
  - 異常系の動作もエミュレート可能(Error Injection機能)
  - モジュールパラメータによってvividドライバの構成を幅広く設定可能

Vivid 仮想ビデオドライバ

- Vivid (drivers/media/platform/vivid)
  - V4L2のテストのために作成されたドライバ
    - v4l2-compliance: V4L2ドライバのテストツール。V4L2ドライバがインターフェース規定に従っているかテストする。Vividは特にこのツールのテストに使用される。
    - v4l2-ctl: V4L2デバイスの情報(実装されているInterface規定等)を表示したり、V4L2 Ctrl API(露光等のデバイスspecificな制御を行うAPI)を扱うツール
    - v4l2-dbg: レジスタ値のダンプ・チップ情報の表示を行うツール
      - ドライバ側で表示したい情報に合わせたioctl実装を行う必要がある
      - vividでは –log-statusオプションのみ有効
    - v4l2-sysfs-path: システム中に存在するV4L2 Deviceの情報を表示(ドライバ名・デバイス一覧表示等)
  - vividはV4L2インターフェース規定の標準的な実装なので、vivid上でアプリケーションを開発することで、実装をよりジェネリックにできる
  - 機材が確保できない/ドライバ開発待ちの場合の代替としてvividを使用する

Vivid 仮想ビデオドライバ

- Kernel Configuration
  - CONFIG_VIDEO_VIVID: vividドライバ有効/無効(LKM対応)
  - CONFIG_VIDEO_VIVID_MAX_DEVS: インスタンス最大数(最大64)
  - CONFIG_VIDEO_VIVID_CEC: HDMI CEC機能有効/無効
- 使い方
  - デフォルトでは、Error Injection機能が有効なので無効にする
    - 無効にしない場合はほとんどのioctlがエラーでreturnする

```
# modprobe vivid no_error_inj=1
# ls /dev/{video,radio,vbi,swradio}*
/dev/radio0 /dev/radio1 /dev/swradio0 /dev/vbi0 /dev/vbi1 /dev/video0 /dev/video1
# gst-launch-1.0 –v v4l2src device=/dev/video0 ¥
  ! video/x-raw, format=I420, width=1280, height=720 ¥
  ! filesink location=test.i420.1280.720
```

Vivid 仮想ビデオドライバ

- テスト映像をキャプチャ可能

Vivid 仮想ビデオドライバ

❖カメラ入力のエミュレートだけ行う場合

```
# modprobe vivid no_error_inj=1 n_devs=2 node_types=0x1,0x1 input_types=0x0
# ls /dev/video*
/dev/video0 /dev/video1
```

- n_devs：vividが作成するデバイスインスタンス数を指定
- node_types：作成するデバイスタイプを指定
  - Bit0: Video Capture node
  - Bit2-3: VBI Capture node
  - Bit4: Radio Receiver node
  - Bit5: Software Defined Radio Receiver node
  - Bit8: Video Output node
  - Bit10-11: VBI Output node
  - Bit12: Radio Transmitter node
  - Bit16: Framebuffer for testing overlays

Vivid 仮想ビデオドライバ

❖カメラ入力のエミュレートだけ行う場合

```
# modprobe vivid no_error_inj=1 n_devs=2 node_types=0x1,0x1 input_types=0x0
# ls /dev/video*
/dev/video0 /dev/video1
```

- Input_type : Video Capture Deviceとしてタイプを指定
  - 00: Webカメラ入力
  - 01: TVチューナー入力
  - 10: S-Video 入力
  - 11: HDMI入力
  - 一番目の入力デバイスはbit0-1, 二番目はbit2-3, …というように指定する

Vivid 仮想ビデオドライバ

- Vividインスタンスに特定のデバイス番号を割り振りたい場合
  - vid_cap_nrに番号を指定(Video Captureデバイスの場合)

```
# modprobe vivid no_error_inj=1 n_devs=2 node_types=0x1,0x1 input_types=0x0
vid_cap_nr=2,3
# ls /dev/video*
/dev/video2 /dev/video3
```

- Crop, Compose, Scalingを有効化する場合
  - ccs_cap_mode: ビデオ入力のCrop, Compose, Scaling有効/無効
    - Bit0: Crop
    - Bit1: Compose
    - Bit2: Scaling

```
# modprobe vivid no_error_inj=1 n_devs=2 node_types=0x1,0x1 input_types=0x0
vid_cap_nr=2,3 ccs_cap_mode=0x7
```

Vivid 仮想ビデオドライバ

- vividのデバッグメッセージ
  - vivid_debugで有効化するメッセージレベル(0, 1, 2)を指定
  - vivid_debugはsysfs経由で設定可能

```
: 無効
# echo 0 > /sys/modules/vivid/parameters/vivid_debug
: レベル1有効
# echo 1 > /sys/modules/vivid/parameters/vivid_debug
: レベル1, 2有効
# echo 2 > /sys/modules/vivid/parameters/vivid_debug
```

- vividのメモリ割り当て
  - allocators: vmalloc(=0)かdma-contig(=1)を設定可能

```
# modprobe vivid no_error_inj=1 n_devs=2 node_types=0x1,0x1 input_types=0x0
vid_cap_nr=2,3 ccs_cap_mode=0x7 allocators=1
```

Vivid 仮想ビデオドライバ

❖サポートされていない機能

❑Subdeviceとの連携

- Ioctl(VIDIOC_SUBDEV_S_FMT)等でsubdeviceを直接操作する
  必要のあるシステムのエミュレートはできない
- Subdeviceを操作する部分をシェルスクリプトにする…等メインのロジックから分離することである程度対処可能

❑Media Controllerとの連携

- Media Controller: デバイス構成を動的に変更する仕組み
  - 例 : /dev/v4l-subdev0 - /dev/video0の組み合わせを
    → /dev/v4l-subdev1 - /dev/video0に変更
- Media Controllerを操作する部分をシェルスクリプトにする…等メインのロジックから分離することである程度対処可能

etc…

まとめ

- V4L2は複数のデバイスドライバが連携して動作する
  - Video Capture Device – V4L2 Subdev – V4L2 Subdev …
- 経路上の全ドライバの開発完了するまで動作できない
- カメラ等専用機材が必要だが、確保できない場合も
- Vividドライバを使用することでドライバ開発完了前・機材確保が難しい場合でもアプリケーションの実装を進められる
- VividドライバはV4L2 Subdev, Media Controller等
  他のドライバとの連携が未サポートなので、
  全てのシステムに適用できるわけではないが、アプリケーションの実装を
  工夫することである程度対処可能

参考資料

- https://www.kernel.org/doc/html/v4.14/media/index.html
- https://kernel.org/doc/html/v4.14/media/uapi/v4l/devices.html
- https://www.kernel.org/doc/html/v4.14/media/uapi/v4l/common.html
- https://www.kernel.org/doc/html/v4.14/media/uapi/v4l/user-func.html
- https://www.kernel.org/doc/html/v4.14/media/v4l-drivers/vivid.html