Extending V4L2

Hans Verkuil

Statistics

- Until kernel 2.6 the v4l subsystem was small: 1-2% of all drivers.
- 2.6 added dvb and lots of new v4l drivers and it has grown from 5% to almost 10%.
- Third-largest subsystem after scsi and net.
- Latest count: 9.15 MB for v4l and 2.59 MB for dvb = 11.74 MB for both.

Kernel	v4l-dvb tree (bytes)	% of all drivers
2.0	287953	1.00%
2.2.26	643187	1.30%
2.4.0	829547	1.50%
2.4.10	1076039	1.70%
2.4.20	1309846	1.70%
2.4.30	1477293	1.70%
2.4.36	1477303	1.70%
2.6.0	4634875	5.80%
2.6.10	4232200	4.60%
2.6.16	5569219	5.40%
2.6.17	7189484	7.00%
2.6.18	7910318	7.70%
2.6.19	8081530	7.60%
2.6.20	8419446	7.80%
2.6.21	8556067	7.80%
2.6.22	9121077	8.10%
2.6.23	9334386	8.20%
2.6.24	9667264	8.10%
2.6.25	10121497	8.10%
2.6.26	11244052	8.90%
2.6.27-rc4	12312140	9.60%

Statistics

- However, of the 9.15 MB of v4l code only 0.15 MB (1.65%) provides core v4l services. Half of that 0.15 MB is the videobuf services, the other half is video_device support.
- Compare with 0.66 MB of core services for scsi (4.9%).

Current state of V4L2

- V4L2 public API: pretty good. Proven to be reasonably future proof.
- V4L drivers: anything from plain broken to excellent. Too many are in poor condition, though.
- Drivers suffer from reinventing the wheel due to severe lack of v4l core services.
- Often drivers are clearly copied from earlier drivers and so feature the same bugs.
- Lack of driver compliance tests makes it hard to verify whether a driver works correctly.

Current state of V4L2

- Modern video hardware devices are ever more complicated. Some create up to 10 devices.
- Modern v4l drivers often also support framebuffer, alsa, i2c, lirc and/or dvb devices, so have to combine various subsystem APIs.
- Upcoming devices have to be able to reroute the internal videostreams.
- Increasingly difficult for both user and applications to pull everything together.

Conclusion

- Some unifying API is needed to keep track of and reroute (if supported) all the devices created by a driver.
- The V4L core services need to be improved.
- Combine the two into one project.

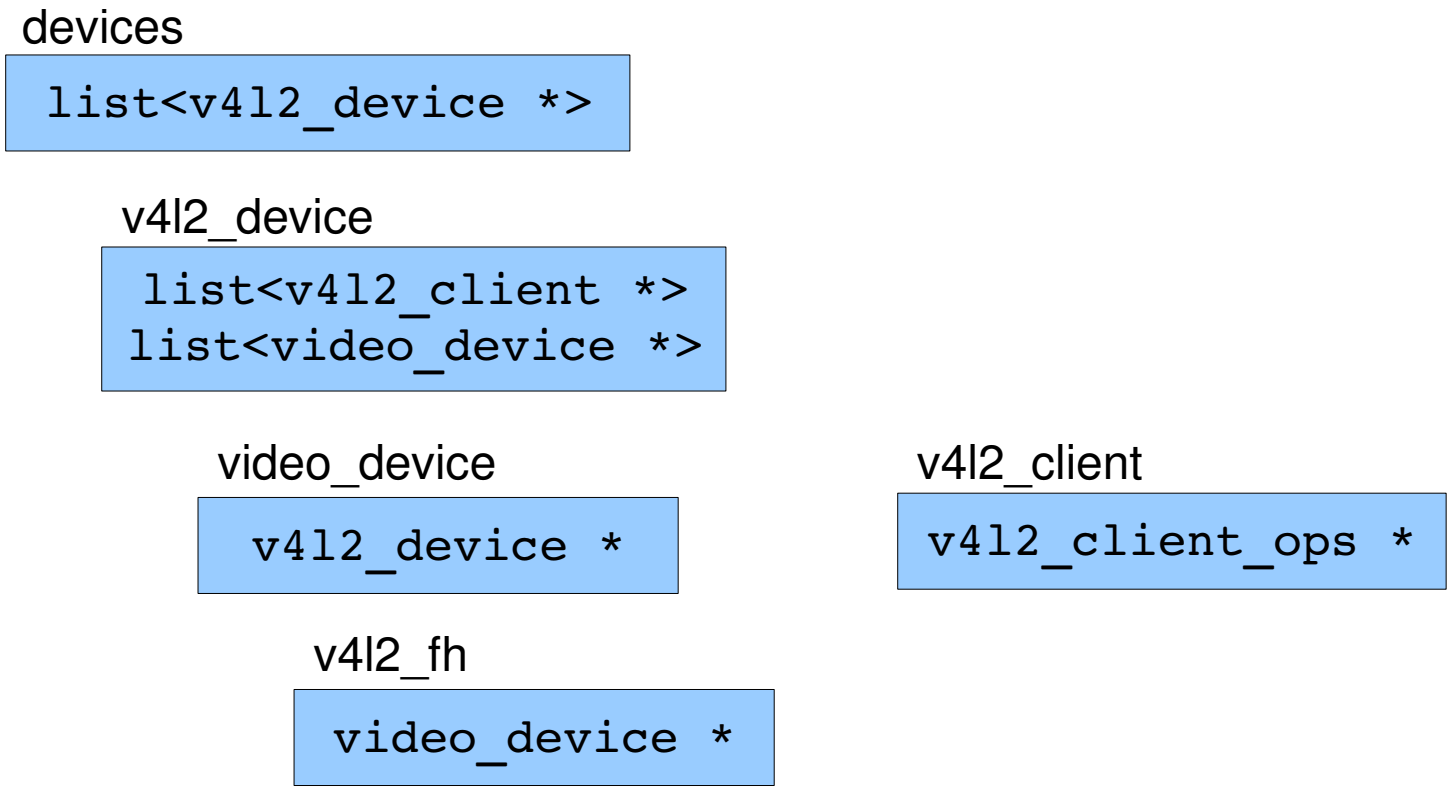
Proposal

- Create a struct v4l2_device for basic device-global data. Register each struct in a global list for easy lookup (e.g. an alsa driver can look for the main driver device data).
- Create a struct v4l2_client to communicate with client (usually i2c) devices. Register them with v4l2_device. When v4l2_device is removed, release the v4l2_clients automatically.
- Create a struct v4l2_fh to store per-filehandle data. Includes priority handling, keeping track of active captures, etc.
- Add standard functions for open/release/read/write/poll similar to the video_ioctl2 call to take care of some of the standard boilerplate code.

Proposal

- Improve control handling support: drivers should not need to care about `QUERYCTRL`, `QUERYMENU` and control value checking.
- Always using v4l2_client for client drivers allows the introduction of utilities that safely load and lock client modules. Almost all drivers do this wrong. Lots of other v4l2_client services can be created to make life easier for the driver programmer.
- Introduce a v4l2_driver object as well? Mostly useful for standardized module option handling.

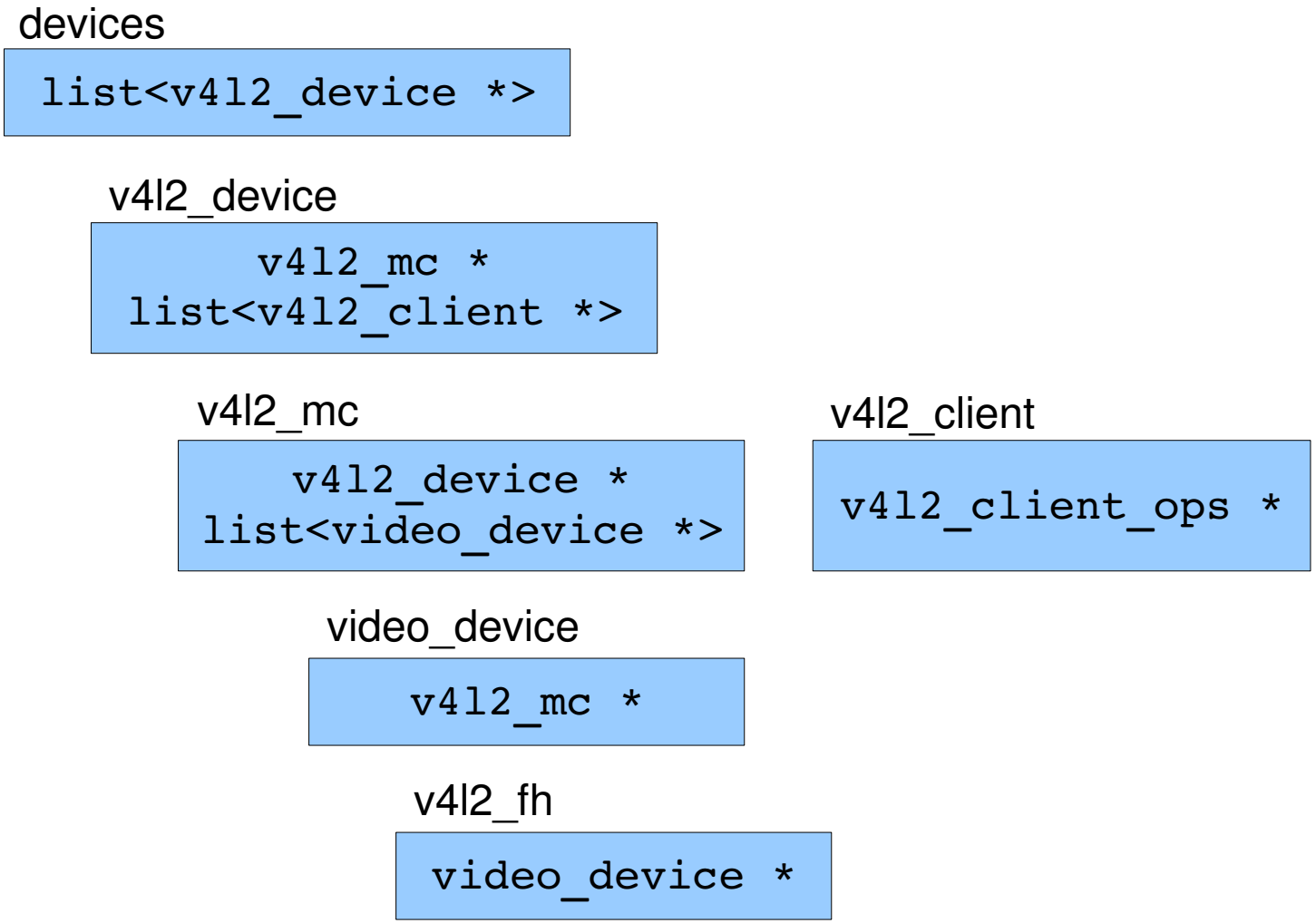
V4L Core Objects



Proposal

- Create a struct v4l2_mc and a /dev/controller0 media controller device.
- Media controller is a new device that can be used to enumerate all v4l/dvb/alsa/fb devices that a given driver created ('Discover the topology').
- Applications only need to find all controller devices and open them to learn what all the capabities are and what other devices to open.
- A media controller can also be used to reroute devices if this is supported.

V4L Core Objects



Proposal

- Make it easier to add new types of device nodes, rather than just video, radio, vbi and vtx (rarely seen). With a media controller it is much easier to give some more meaningful names, e.g. encoder0 or decoder0 for MPEG encoder/decoder streams.
- Think about media processor devices: can be used to tell hardware how to process media. Use for compositors, video effects, etc. Must be very flexible.

Proposal

- New objects can be introduced step-by-step.
- Converting drivers to use these objects should improve consistency and quality.
- All drivers create their own variations of these structs. It makes a lot of sense to move that code to a v4l core framework so that drivers can concentrate on getting the hardware to work.
- Applications should see much more uniform behavior by drivers.

v4l2_device

```
struct v4l2_device {
        struct list_head list;
        struct list_head clients;
        struct mutex lock;

        /* id + num must be unique */
        enum v4l2_driver_id id;
        u16 num;
        /* name must be unique */
        char name[V4L2_DEVICE_NAME_SIZE];
        struct v4l2_prio_state prio;
};
```

v4l2_mc

```
struct v4l2_dev_node {
        struct list_head list;
        char name[V4L2_NODENAME_SZ];
        enum v4l2_dev_node_type type;
        union {
           struct video_device *vdev;
           struct fb_info *fb;
           struct alsa_??? *alsa;
           struct dvb_adapter *dvb;
        };
};

struct v4l2_mc {
        struct list_head dev_nodes;
        struct mutex lock;
        struct video_device *mc_dev;
        struct v4l2_dev *dev;
};
```

v4l2_client

```
struct v4l2_client_ops {
        const struct v4l2_client_core_ops  *core;
        const struct v4l2_client_tuner_ops *tuner;
        const struct v4l2_client_audio_ops *audio;
        const struct v4l2_client_video_ops *video;
};

struct v4l2_client {
        struct list_head list;
        struct module *owner;
        struct v4l2_device *dev;
        const struct v4l2_client_ops *ops;
        void *priv;
};
```

v4l2_fh

```
enum v4l2_fh_mode {
        V4L2_FH_PASSIVE = 0,
        V4L2_FH_ACTIVE = 1,
};

struct v4l2_fh {
        struct video_device *vdev;      /* v4l2_vdev */
        enum v4l2_fh_mode mode;
        enum v4l2_priority prio;        /* priority */
};
```