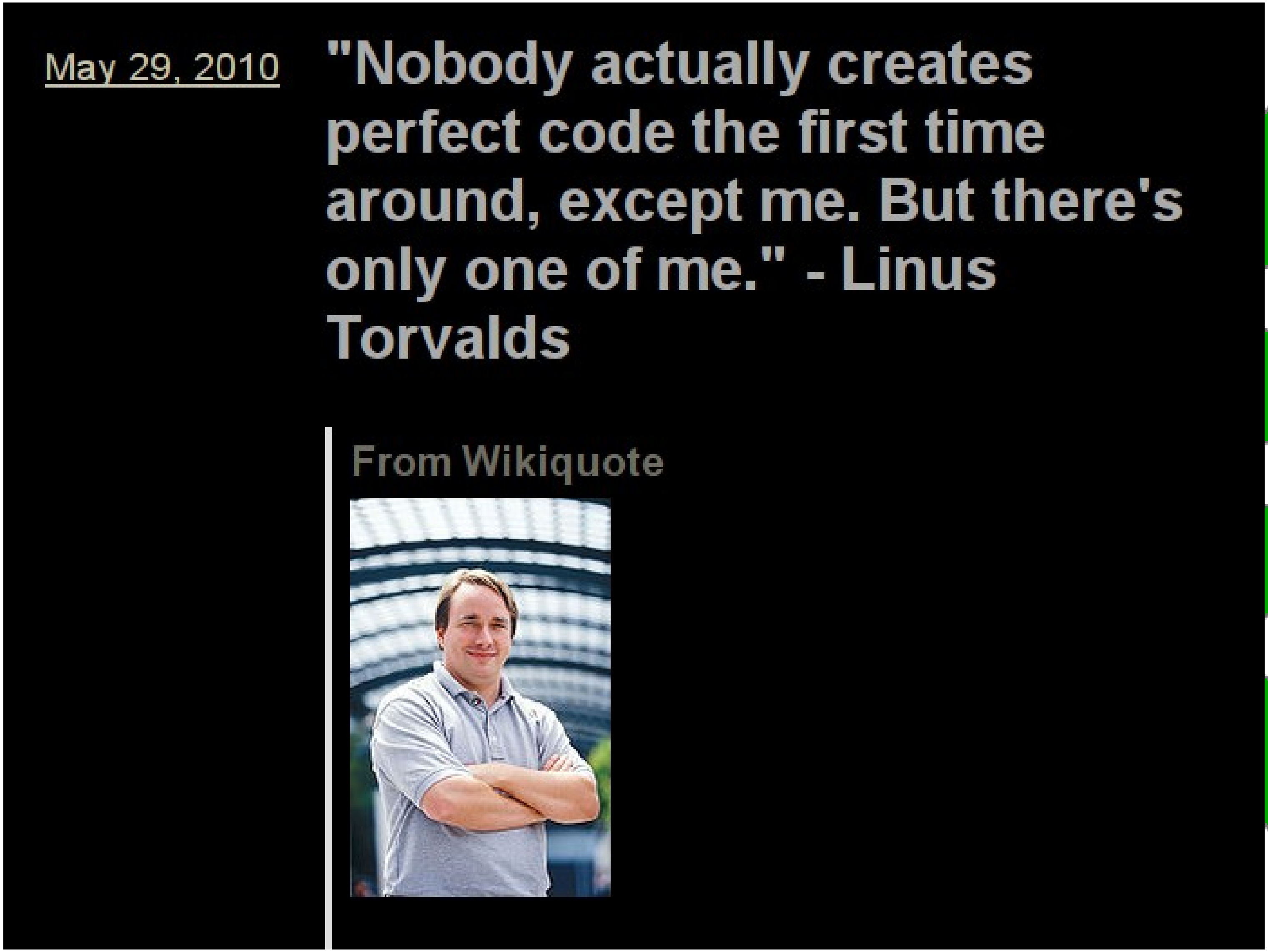
videobuf

The good, the bad and the ugly

V4L2 Helsinki Summit 2010-06-14

Laurent Pinchart
laurent.pinchart@ideasonboard.com



May 29, 2010

"Nobody actually creates perfect code the first time around, except me. But there's only one of me." - Linus Torvalds

From Wikiquote

- Manage a queue of video buffers
- Handle all video buffer related IOCTLs and file operations
- Minimize code in drivers
- Ensure V4L2 compliance
- read(), mmap() and user pointer support for free

The good - Goals

The good – Userspace API

- Video buffers IOCTLs handling
  - VIDIOC_REQBUFS → videobuf_reqbufs
  - VIDIOC_QUERYBUF → videobuf_querybuf
  - VIDIOC_QBUF → videobuf_qbuf
  - VIDIOC_DQBUF → videobuf_dqbuf
  - VIDIOC_STREAMON → videobuf_streamon
  - VIDIOC_STREAMOFF → videobuf_streamoff

```
static int bttv_reqbufs(struct file *file, void *priv,
                        struct v4l2_requestbuffers *p)
{
        struct bttv_fh *fh = priv;
        return videobuf_reqbufs(bttv_queue(fh), p);
}

static int bttv_querybuf(struct file *file, void *priv,
                         struct v4l2_buffer *b)
{
    struct bttv_fh *fh = priv;
    return videobuf_querybuf(bttv_queue(fh), b);
}

static const struct v4l2_ioctl_ops bttv_ioctl_ops = {
    .vidioc_reqbufs = bttv_reqbufs,
    .vidioc_querybuf = bttv_querybuf,
}
```

IOCTL API example

The good - Userspace API

- mmap() and poll()
  - mmap → videobuf_mmap_mapper
  - poll → videobuf_poll_stream
- read()
  - frame-based video → videobuf_read_one
  - VBI and MPEG → videobuf_read_stream

```
static int bttv_mmap(struct file *file, struct vm_area_struct *vma)
{
    struct bttv_fh *fh = file->private_data;
    return videobuf_mmap_mapper(bttv_queue(fh),vma);
}

static unsigned int bttv_poll(struct file *file, poll_table *wait)
{
    struct bttv_fh *fh = file->private_data;

    if (V4L2_BUF_TYPE_VBI_CAPTURE == fh->type)
        return videobuf_poll_stream(file, &fh->vbi, wait);

    ...
}

static const struct v4l2_file_operations bttv_fops = {
    .mmap = bttv_mmap,
    .poll = bttv_poll,
};
```

mmap and poll API example

```
static ssize_t bttv_read(struct file *file, char __user *data,
                         size_t count, loff_t *ppos)
{
    struct bttv_fh *fh = file->private_data;

    switch (fh->type) {
    case V4L2_BUF_TYPE_VIDEO_CAPTURE:
        return videobuf_read_one(&fh->cap, data, count, ppos,
                                 file->f_flags & O_NONBLOCK);
    case V4L2_BUF_TYPE_VBI_CAPTURE:
        return videobuf_read_stream(&fh->vbi, data, count, ppos, 1,
                                    file->f_flags & O_NONBLOCK);
    }
}

static const struct v4l2_file_operations bttv_fops = {
    .read = bttv_read,
};
```

read API example

The good – Drivers callbacks

- Drivers need to implement four operations

```
struct videobuf_queue_ops {
    int (*buf_setup)(struct videobuf_queue *q,
                     unsigned int *count, unsigned int *size);
    int (*buf_prepare)(struct videobuf_queue *q,
                       struct videobuf_buffer *vb,
                       enum v4l2_field field);
    void (*buf_queue)(struct videobuf_queue *q,
                      struct videobuf_buffer *vb);
    void (*buf_release)(struct videobuf_queue *q,
                        struct videobuf_buffer *vb);
};
```

The good – Modular allocation

- Fixed core: implements the V4L2 API
- Modular memory type: implements memory management
  - vmalloc: Virtual memory (USB)
  - dma_contig: DMA to physically contiguous memory (embedded without IOMMU)
  - dma_sg: DMA to physically scattered memory (PCI, embedded with IOMMU)

The good – Memory management

- Allocate and free buffer memory, validate user provided memory.
- Translation between various memory representations (physical address, virtual address, pages list, scatter list, …)
- Lock pages in memory
- Synchronize DMA and cache
- Implement mmap()

The bad – Drivers API

- The API has a history
  - Inconsistent or vague naming
  - Internal functions exported for a single caller
  - Functions reimplemented by drivers for no apparent reason
  - Inconsistent usage
  - Per-buffer wait queues
  - Unused fields, magic numbers
  - Cumbersome V4L1 compatibility
  - Too little documentation

The bad – API use and abuse

- videobuf-dma-sg has two levels of APIs
  - videobuf memory type API to handle memory management for driver that have DMA scatter-gather support
  - Internal API to handle scatter list creation, page locking, … for random memory blocks
- The internal API is exported and used by drivers for audio buffers management
- The memory type API is used by the internal API with fake arguments

```
struct videobuf_qtype_ops {
    struct videobuf_buffer *(*alloc)(size_t size);
    void *(*vaddr)             (struct videobuf_buffer *buf);
    int (*iolock)              (struct videobuf_queue *q,
                                struct videobuf_buffer *vb,
                                struct v4l2_framebuffer *fbuf);
    int (*sync)                (struct videobuf_queue *q,
                                struct videobuf_buffer *buf);
    int (*mmap_mapper)         (struct videobuf_queue *q,
                                struct videobuf_buffer *buf,
                                struct vm_area_struct *vma);
};
```

- One do-it-all function (iolock)
- Bad DMA and cache management (sync)

The bad – Memory operations

The bad – It could be worse

- That's fixable (hopefully)
  - Many problems are identified, some have already been fixed
  - Hans and Pavel submitted patches and got them integrated
  - 7 patches pending on linux-media, more still required
  - Kudos to Jonathan Corbet for the documentation
  - Help from drivers maintainers is needed

- V4L2 non-compliance
- Allocation on mmap or qbuf (or worse on page fault), implicitly by videobuf
- Free on munmap or streamoff, explicitly by drivers
- Do-it-all iolock operation

The ugly – Memory allocation

The ugly – videobuf_dma_sg

- VIDIOC_REQBUFS: buffer objects are allocated, memory isn't
- mmap(): mapping is created, still no underlying memory (access will fault)
- VIDIOC_QBUF: get_user_pages() called, will fault every page
- Page fault handler: allocate pages one at a time

- No clearly defined semantics
- buf_release called on streamoff
  - Free driver-specific resources
  - Free buffers explictly
- buf_release used by drivers synchronization purpose

The ugly – API semantic