How to write a Device Driver in FreeBSD

John-Mark Gurney

Frameworks

- kld
- newbus
- rman/bus_space(9)
- cdevsw
- bus_dma(9)
- sysctl
- SYSINIT

newbus

- Device Methods
  - Tree with root
- Device is a bus if it has children
- Inheritance
  - ofw_pci from pci
- Unit number allocation (via devclass(9))
  - Same devclass for all device's bus
- Softc allocation (via driver(9))

sys/dev/re/if_re.c

```
static device_method_t re_methods[] = {
        /* Device interface */
        DEVMETHOD(device_probe    re_probe),
        DEVMETHOD(device_attach,  re_attach),
        DEVMETHOD(device_detach,  re_detach),
        DEVMETHOD(device_suspend,         re_suspend),
        DEVMETHOD(device_resume, re_resume),
        DEVMETHOD(device_shutdown,        re_shutdown),

        /* MII interface */
        DEVMETHOD(miibus_readreg,re_miibus_readreg),
        DEVMETHOD(miibus_writereg,        re_miibus_writereg),
        DEVMETHOD(miibus_statchg, re_miibus_statchg),

        { 0, 0 }
};

static driver_t re_driver = {
        "re",
        re_methods,
        sizeof(struct rl_softc)
};

static devclass_t re_devclass;

DRIVER_MODULE(re, pci, re_driver, re_devclass, 0, 0);
DRIVER_MODULE(re, cardbus, re_driver, re_devclass, 0, 0);
DRIVER_MODULE(miibus, re, miibus_driver, miibus_devclass, 0, 0);

MODULE_DEPEND(re, pci, 1, 1, 1);
MODULE_DEPEND(re, ether, 1, 1, 1);
MODULE_DEPEND(re, miibus, 1, 1, 1);
```

Device States

- Probing
  - Each possible device_probe method called till best or BUS_PROBE_SPECIFIC found
- Attaching
- Busy
  - Calling device_busy(dev) prevents detach

Device Attach

- Setup softc
  - Automaticly allocated and zero'd via driver
- Allocate Resources
  - bus_alloc_resources
- Setup Interrupt
- Setup Character Devices

Resources

- Managed by the parent of device
- One of:
  - SYS_RES_MEMORY
  - SYS_RES_IOPORT
  - SYS_RES_IRQ
  - SYS_RES_DRQ
- Memory and IO accessed via bus_space(9)

bus_setup_intr

```
int bus_setup_intr(device_t , struct resource * , int flags ,
   driver_intr_t , void * , void **cookiep);

          error = bus_setup_intr(dev, bktrau->bktrau_irq,
               INTR_TYPE_TTY|INTR_MPSAFE, bktrau_intr, bktrau,
               &bktrau->bktrau_irqh);
          if (error) {
                    device_printf(dev, "could not setup irq\n");
                    goto fail;
          }
```

Character Devices

- Interface for common Unix operations
  - open
  - read
  - write
  - ioctl
  - mmap
  - poll
  - select
  - close

d_functions

- typedef int d_{read,write}_t(struct cdev *dev, struct uio *uio, int ioflag);
- typedef int d_kqfilter_t(struct cdev *dev, struct knote *kn);
- typedef int d_mmap_t(struct cdev *dev, vm_offset_t offset, vm_paddr_t *paddr, int nprot);

d_ioctl

- #define BKTRAU_SETAUDIO _IOW('A', 0, struct bktrau_audio) /* set options */
- #define BKTRAU_GETAUDIO _IOR('A', 1, struct bktrau_audio) /* get options */
- Kernel copies necessary data for you

d_poll

```
typedef int d_poll_t(struct cdev *dev, int events, struct thread
*td);

    revents = 0;
    if (bktrau->bktrau_status == BKTRAU_S_RUNNING) {
        if (events & (POLLIN|POLLRDNORM)) {
            if (bktrau->bktrau_head != bktrau->bktrau_avail)
                revents = events & (POLLIN|POLLRDNORM);
            else
                selrecord(td, &bktrau->bktrau_sel);
        }
    }

    return revents;
```

Attached, Now What?

- Wait for d_open to be called
- For bktrau, most work done via d_ioctl

bus_dma

- tag describes limitations to dma
- map used for each memory block to dma to/from
- must sync before and after dma operations to ensure proper bounce buffer handling

bus_dma_tag_create

- Provides restrictions
  - Alignment
  - Boundary
  - Address (can be filtered)
  - Maximum total size
  - Number of segments
  - Maximum size of each segment
- Lock to hold over callbacks
- Parent tag coming

bus_dma_tag_create

```
	if ((error = bus_dma_tag_create(NULL, 4, 0,
BUS_SPACE_MAXADDR_32BIT, BUS_SPACE_MAXADDR, NULL,
NULL, BKTRAU_MAXBUFSIZE, BKTRAU_MAXNSEGS,
BKTRAU_MAXBUFSIZE, 0, busdma_lock_mutex,
&bktrau->bktrau_lock, &bktrau->bktrau_tag))) {
		device_printf(dev, "tag create\n");
		goto fail;
	}

	if ((error = bus_dmamap_create(bktrau->bktrau_tag, 0,
&ptr[i])))
		break;

	error = bus_dmamap_load_uio(bktrau->bktrau_tag,
bktrau->bktrau_maps[i], &u, bktrau_uio_cb, (void *)bktrau,
BUS_DMA_WAITOK);
```

bus_dma maps

- After creating, need to load it
  - bus_dmamap_load
  - bus_dmamap_load_mbuf[_sg]
    - _sg doesn't do a callback
  - bus_dmamap_load_uio
- Create and alloc memory at once
  - bus_dmamem_alloc
- Unload before bus_dmamap_destroy

bus_dmamap_callback2_t

```
static void
bktrau_uio_cb(void *arg, bus_dma_segment_t *segs, int nsegs,
   bus_size_t mapsize, int error)
{
        struct bktrau_softc *bktrau;
        int i;

        if (error) {
              bktrau->bktrau_mapdoing = -error;
              return;
        }

        bktrau = (struct bktrau_softc *)arg;
        bktrau->bktrau_addrs[bktrau->bktrau_mapdoing].bs_cnt = nsegs;
        for (i = 0; i < nsegs; i++)
              bktrau->bktrau_addrs[bktrau->bktrau_mapdoing].bs_segs[i] =
   segs[i];
}
```

bus_dma_sync

- Handles bouncing data if necessary
- Flushes cpu caches
- WRITE is pushing data to the device
  - Packet to be transmitted
  - Ring buffer for packets
- READ is getting data from the device
  - Receiving packet
  - Captured Video data

Configuration Knobs

- SYSCTL
  - read/write
  - Basic types

```
int dv_cfg;
SYSCTL_INT(_debug, OID_AUTO, dv_cfg,
CTLFLAG_RW, &dv_cfg, 0, “Config driver”);
```

SYSINITs

- Used to control entier system startup
- Dynamicly aggregated
  - No symbol polution via linker sets
- Provides order (in an otherwise unordered set)
- Calls function with a void * argument

Callouts and Taskqueues

- Callouts
  - Timeouts
  - MPSAFE
- Taskqueue
  - Move heavy work to another thread

I2C aka IIC

- IC to IC communications protocol
  - Two wires (SDA and SCL), many devices
- iicbb can be attached to GPIO pins
- i2c->zi_iicbb = device_add_child(dev, "iicbb", -1);
- sys/dev/iicbus/iicbb_if.m
  - iicbb_callback – used to lock bus
  - iicbb_{get,set}{sda,scl}
  - iicbb_reset