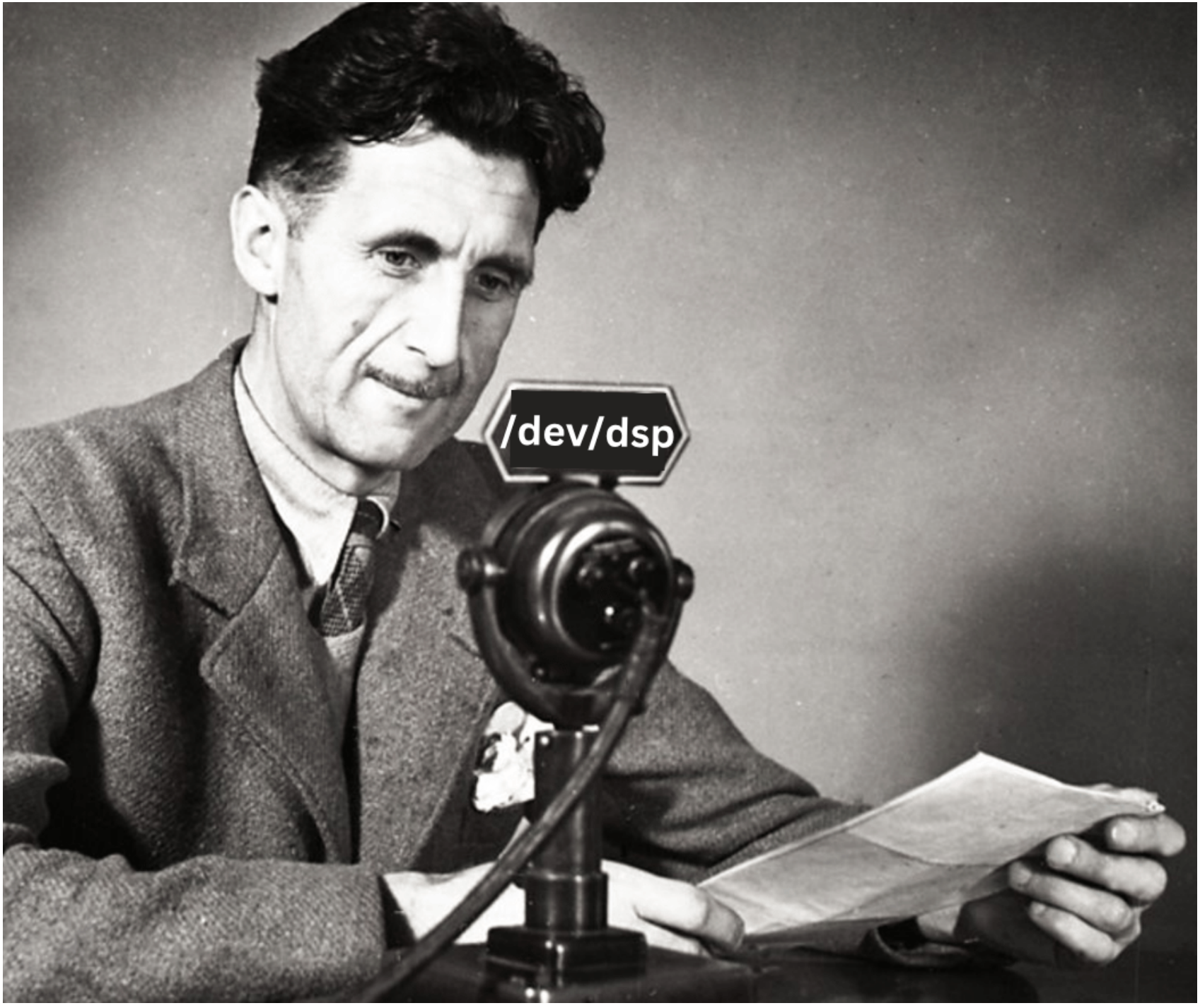
Vox FreeBSD: How sound(4) works

Christos Margiolis
christos@FreeBSD.org

June 14, 2025
BSDCan 2025 — Ottawa, Canada


/dev/dsp

Who?

- FreeBSD committer.
- The guy who keeps churning out sound bugs.

What I installed

What I expected

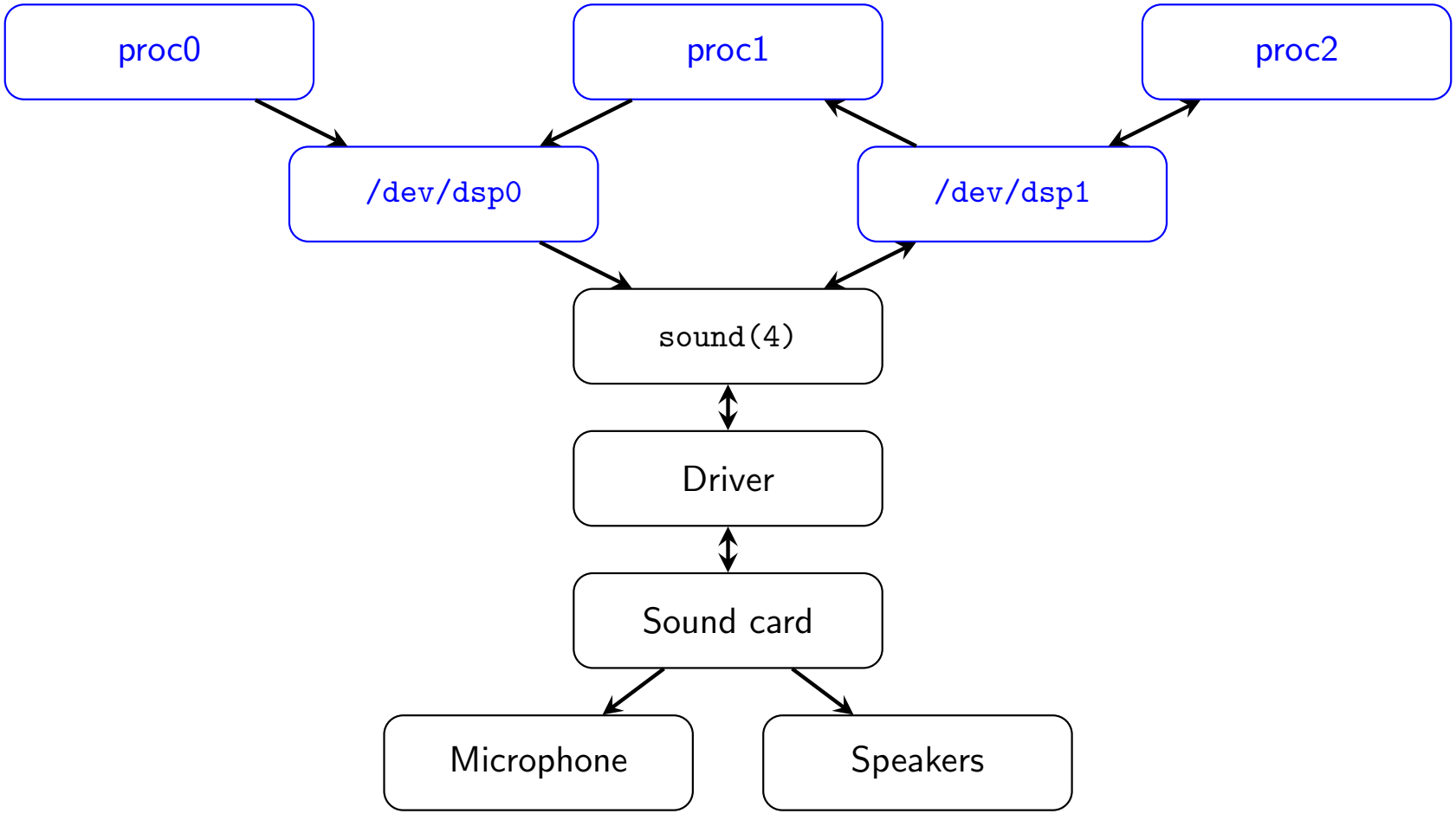
What I got

```
Fatal trap 12: page fault while in kernel mode
cpuid = 1; apic id = 01
fault virtual address   = 0x24
fault code              = supervisor read data, page not present
instruction pointer     = 0x20:0xffffffff829146d9
stack pointer           = 0x28:0xfffffe004645ade0
frame pointer           = 0x28:0xfffffe004645ae00
code segment            = base 0x0, limit 0xfffff, type 0x1b
                        = DPL 0, pres 1, long 1, def32 0, gran 1
processor eflags        = interrupt enabled, resume, IOPL = 0
current process         = 2 (clock (0))
rdi: fffff8000158c580 rsi: 0000000000000008 rdx: 0000000000000000
rcx: 0000000000010000  r8: 00000000000000d5  r9: ffffffff811680dd
rax: 0000000000000002 rbx: fffff8000192e000 rbp: fffffe004645ae00
r10: fffff80013d9b880 r11: ffffffff81a33a58 r12: 0000000000000000
r13: 0000000000000084 r14: fffff8000192e180 r15: 0000000000000000
trap number             = 12
panic: page fault
cpuid = 1
time = 1749865327
KDB: stack backtrace:
db_trace_self_wrapper() at db_trace_self_wrapper+0x2b/frame 0xfffffe004645ab10
vpanic() at vpanic+0x13f/frame 0xfffffe004645ac40
panic() at panic+0x43/frame 0xfffffe004645aca0
trap_pfault() at trap_pfault+0x48d/frame 0xfffffe004645ad10
calltrap() at calltrap+0x8/frame 0xfffffe004645ad10
--- trap 0xc, rip = 0xffffffff829146d9, rsp = 0xfffffe004645ade0, rbp = 0xfffffe004645ae00 ---
dummy_chan_io() at dummy_chan_io+0xe9/frame 0xfffffe004645ae00
softclock_call_cc() at softclock_call_cc+0x1a4/frame 0xfffffe004645aec0
softclock_thread() at softclock_thread+0xd6/frame 0xfffffe004645aef0
fork_exit() at fork_exit+0x82/frame 0xfffffe004645af30
fork_trampoline() at fork_trampoline+0xe/frame 0xfffffe004645af30
--- trap 0, rip = 0, rsp = 0, rbp = 0 ---
KDB: enter: panic
[ thread pid 2 tid 100031 ]
Stopped at      kdb_enter+0x33: movq    $0,0xe6cc32(%rip)
db>
```

Contents

- How does sound travel from application to the real world (and vice versa)?
- Layers: userland, `sound(4)`, device drivers.
- New improvements.
- FreeBSD for music and audio production?

Userland


proc0
proc1
proc2
/dev/dsp0
/dev/dsp1
sound(4)
Driver
Sound card
Microphone
Speakers

Userland

Interacts with `sound(4)` through the Open Sound System (OSS) API, using a few basic syscalls on `/dev/dsp*` and `/dev/mixer*` character devices:

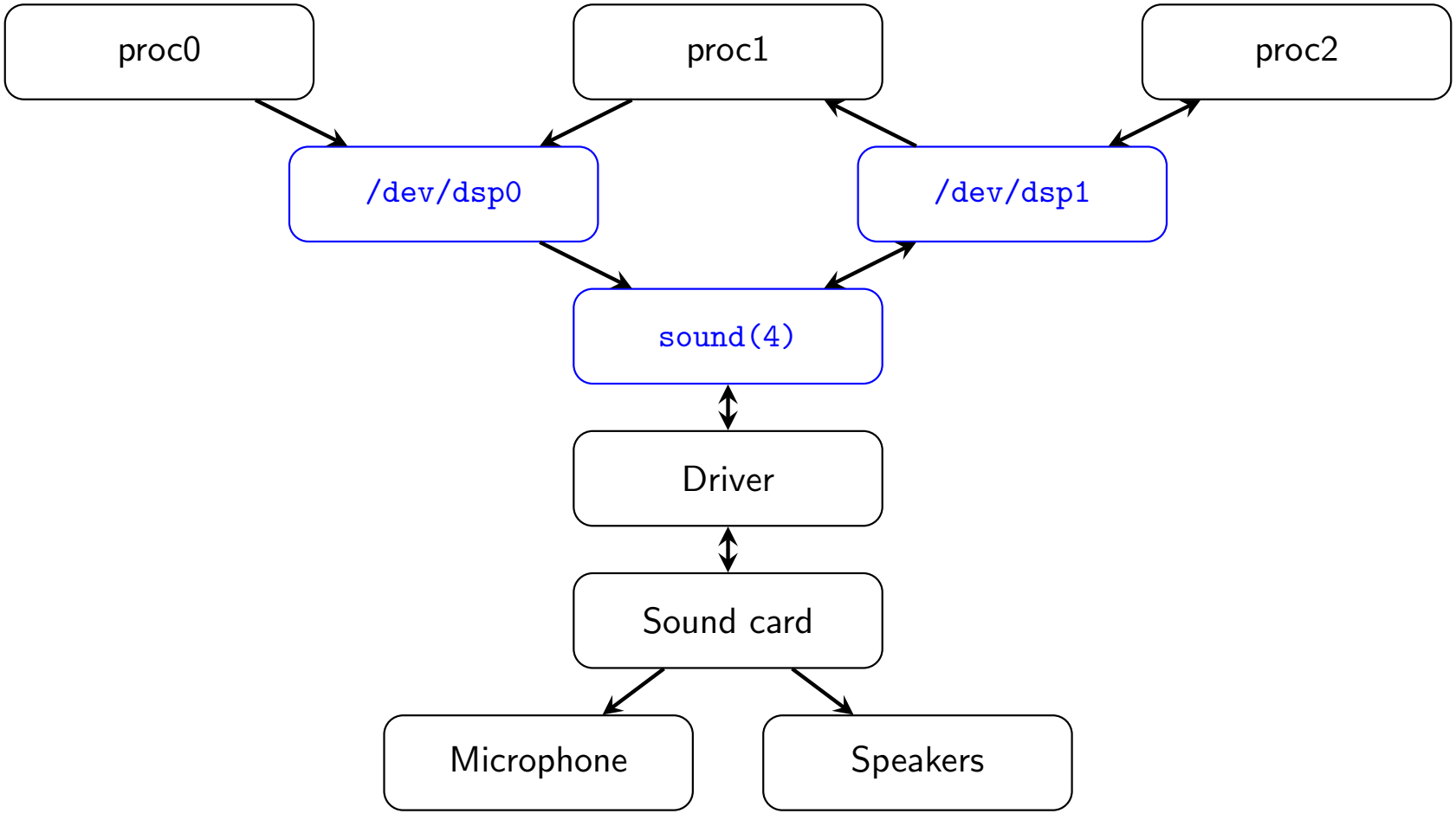
`open(2)`	Open device, obviously...
`close(2)`	Close device.
`read(2)`	Record audio.
`write(2)`	Play audio.
`ioctl(2)`	Query and manipulate settings (sample rate, format, volume, ...).
`select(2) & poll(2)`	Wait for events when in non-blocking mode.
`mmap(2)`	Direct IO with the sound card. Discouraged.

http://manuals.opensound.com/developer/

Basic audio loopback program

```
#include <sys/soundcard.h>

#include <fcntl.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
        uint32_t sample;
        int fd, fmt, chans, rate;

        /* No error checking. */
        fd = open("/dev/dsp", O_RDWR);

        chans = 1;
        ioctl(fd, SNDCTL_DSP_CHANNELS, &chans);

        fmt = AFMT_S16_LE;
        ioctl(fd, SNDCTL_DSP_SETFMT, &fmt);

        rate = 48000;
        ioctl(fd, SNDCTL_DSP_SPEED, &rate);

        for (;;) {
                read(fd, &sample, sizeof(sample));
                write(fd, &sample, sizeof(sample));
        }

        close(fd);

        return (0);
}
```

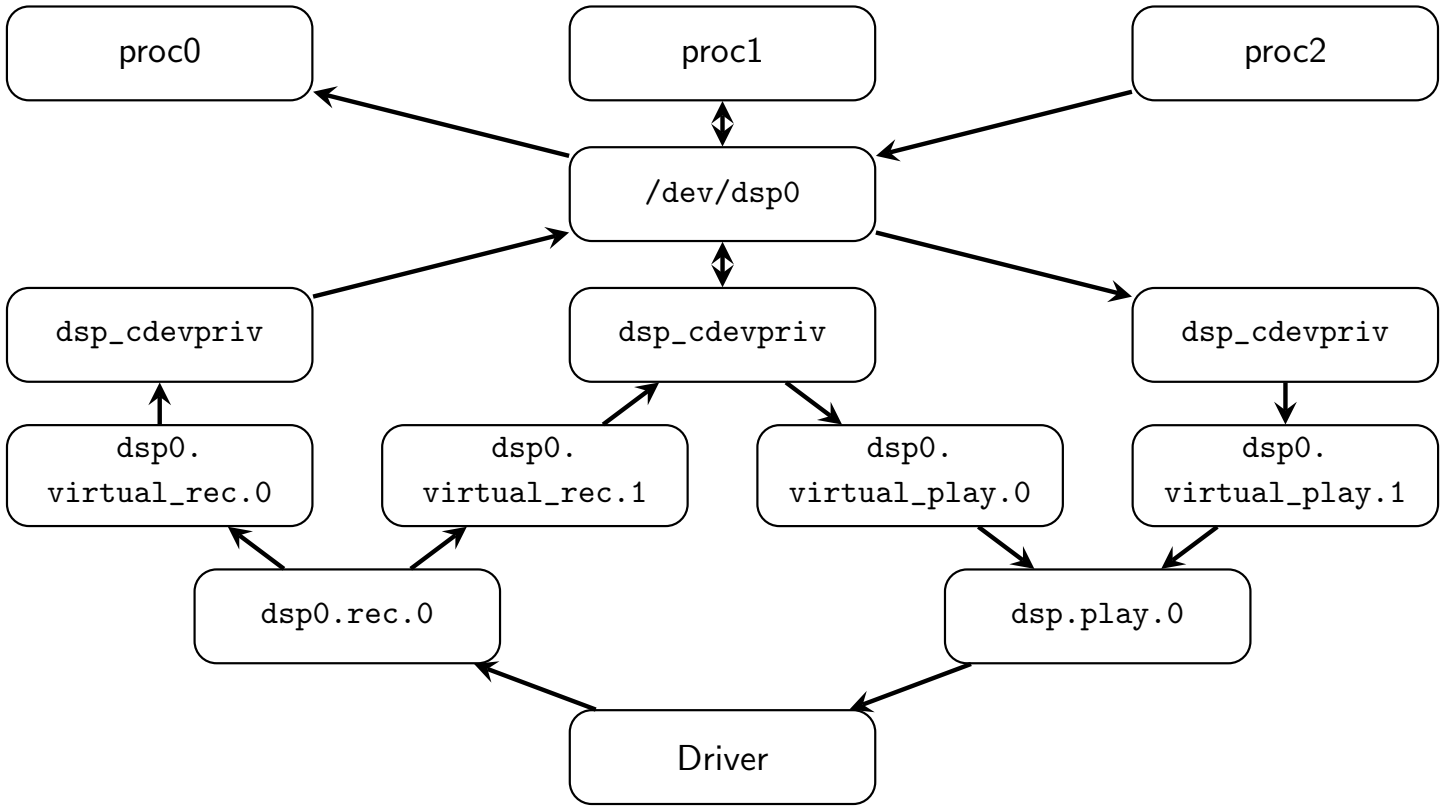
sound(4)


proc0
proc1
proc2
/dev/dsp0
/dev/dsp1
sound(4)
Driver
Sound card
Microphone
Speakers

- You might have also seen it mentioned as `pcm`.
- Generic layer.
- Implements the OSS API.
- Exposes devices and their mixers as character devices: `/dev/dsp*`, `/dev/mixer*`
- Handles channels and buffers.
- Processing chain.
- sysctls: `hw.snd.*`, `dev.pcm.*`
- `/dev/sndstat`

sound(4): /dev/dsp*

- Access for playback and/or recording.
- Uses DEVFS_CDEVPRIV(9).
- There is also /dev/dsp which routes to the default device (hw.snd.default_unit).



`sound(4)`: `/dev/mixer*`

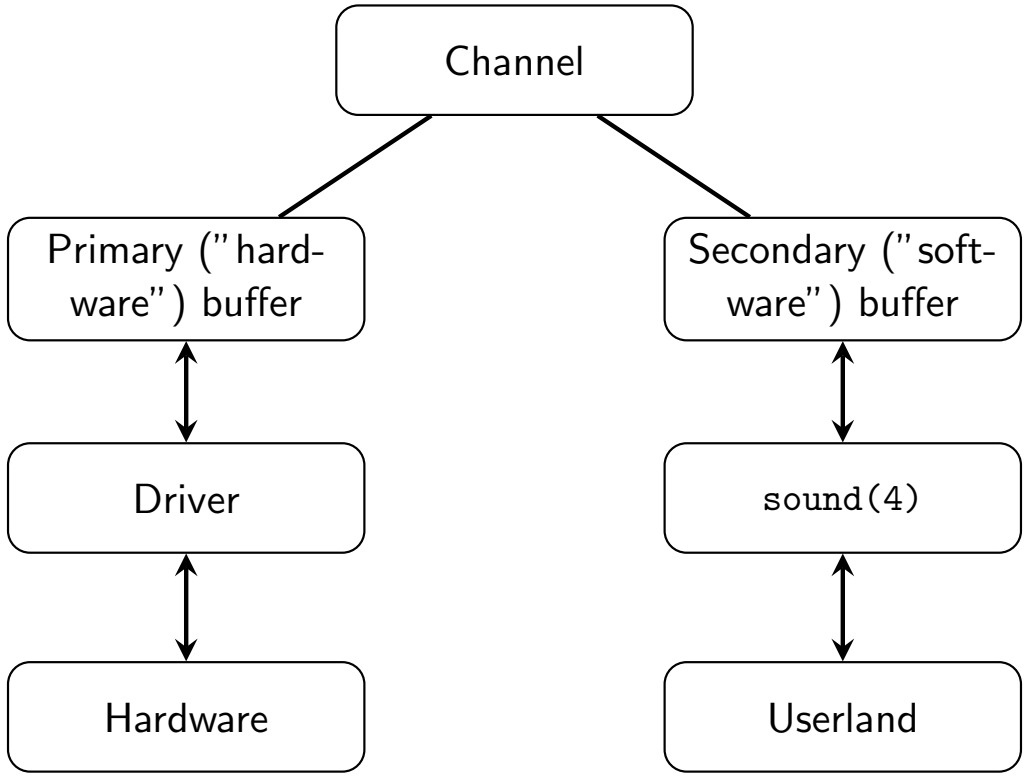
- Mainly used for volume, (un-)muting, and recording source setting.
- Theoretically not really needed anymore since OSSv4, but we still use it.
- Used by `mixer(8)` through `mixer(3)`.

sound(4): /dev/sndstat

- Information about attached sound devices.
- Also provides an nv(9) interface. Used by sndctl(8) (more on that later), virtual_oss(8), ...
- hw.snd.verbose

`sound(4)`: Channels

- Primary ("hardware") channels.
- Virtual channels (VCHANs). Can be disabled.



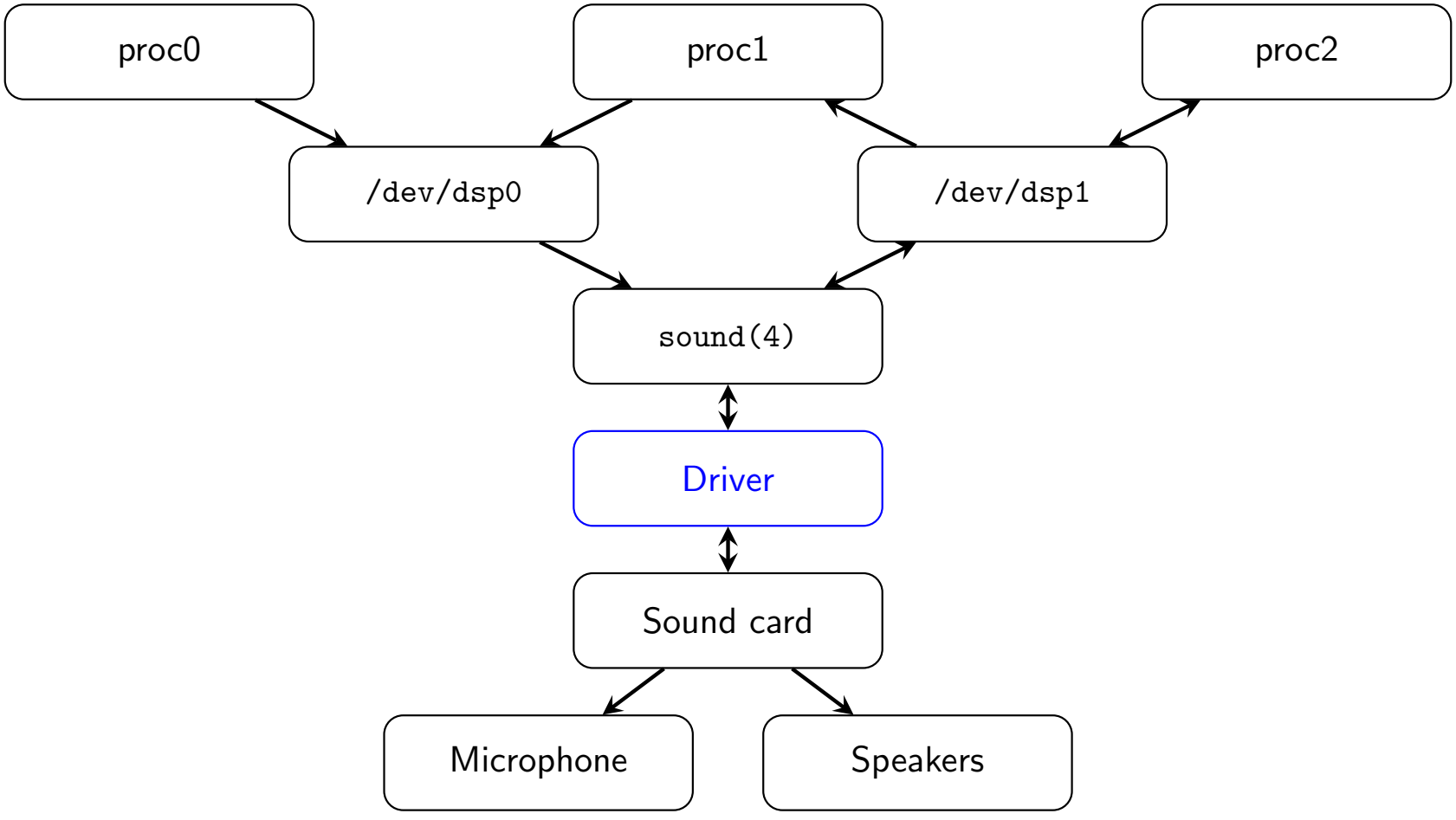
`sound(4)`: Processing chain

- Sample rate & format conversions, equalizer, multi-channel mixing, channel matrixing, volume control.
- Each channel gets its own chain during creation.
- Triggered by the driver: `chn_intr()`.
- `sndctl feederchain`

`sound(4)`: Reducing latency

- Disable VCHANs: `sndctl play.vchans=0 rec.vchans=0`
- Skip processing (bitperfect): `sndctl bitperfect=1`
- Shorthand: `sndctl realtime=1 autoconv=0`
- `hw.snd.latency`
- More sysctls, including the driver-specific ones...
- `mac_priority(4)` and `rtprio(1)`.
- Florian Walpen's notes on low latency with JACK: https://www.submerge.ch/FreeBSD/freebsd_jack_notes/index.html

Device drivers



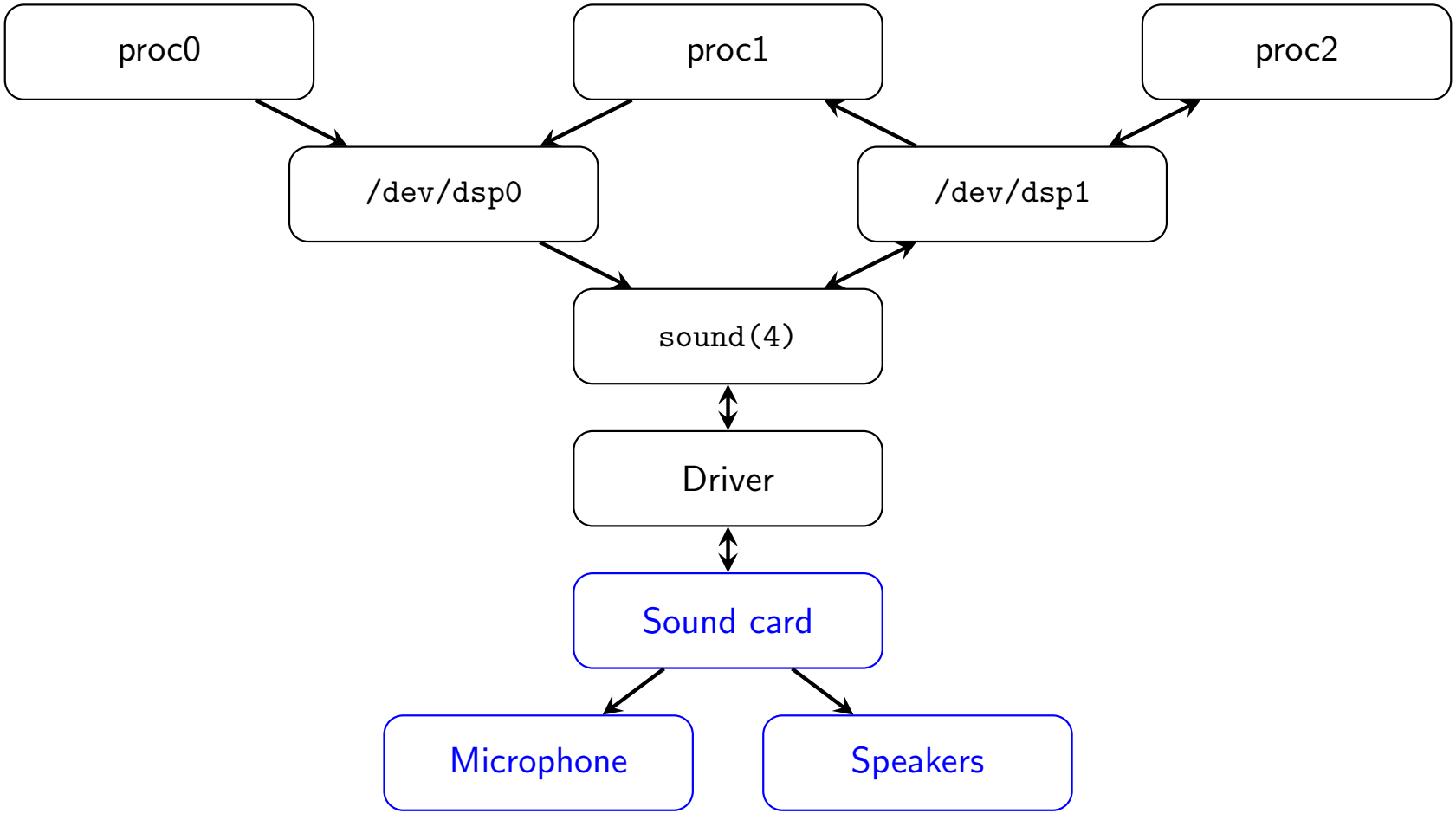
Device drivers

- Communication layer between `sound(4)` and the sound card.
- Implement the `sound(4)` kernel object interfaces.
- `snd_uaudio(4)`, `snd_hda(4)`, `snd_hdsp(4)`, ...
- Also a testing driver: `snd_dummy(4)`.
- Some implement their own sysctls as well (e.g., `hw.usb.uaudio`, `dev.hdaa`, ...).

Device drivers: Setting up

- Initialize driver-internal resources (locks, DMA, USB, PCI, callouts, ...).
- Implement the `channel_if.m` and `mixer_if.m` methods.
- Create primary channels: `pcm_addchan()`.
- Register to `sound(4)`: `pcm_init()`, `pcm_register()`.
- Create the mixer: `mixer_init()`.
- See `sys/dev/sound/dummy.c`.

Hardware



This is not a hardware talk...

New improvements

- Better laptop support.
  - https://reviews.freebsd.org/D50070
- New tools: `sndctl(8)`, `mididump(1)`.
- Hot-unplug.
- Bug fixes.
- Clean ups and refactors.
- Tests.
- `AFMT_FLOAT` support.
- Took over development of `virtual_oss(8)`.
- More...

FreeBSD for music production?

- Yes. There are people who do this thing (me).
- Solid and fast sound system.
- Good and growing collection of DAWs and LV2 ports.

Acknowledgements

Ed Maste <emaste@FreeBSD.org>
Florian Walpen <dev@submerge.ch>
Goran Mekić <meka@tilda.center>

Thank you.