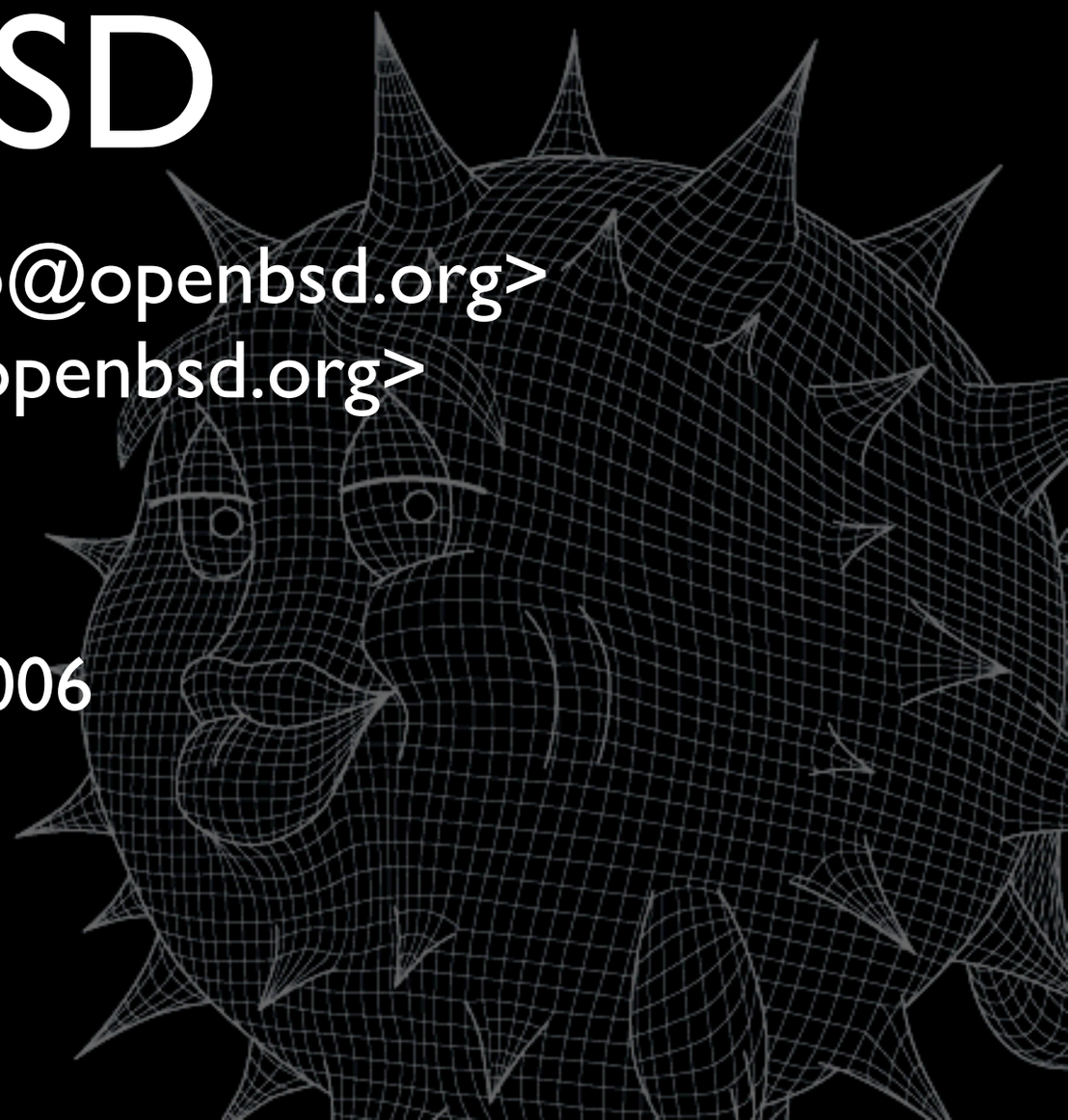
bio and sensors in OpenBSD

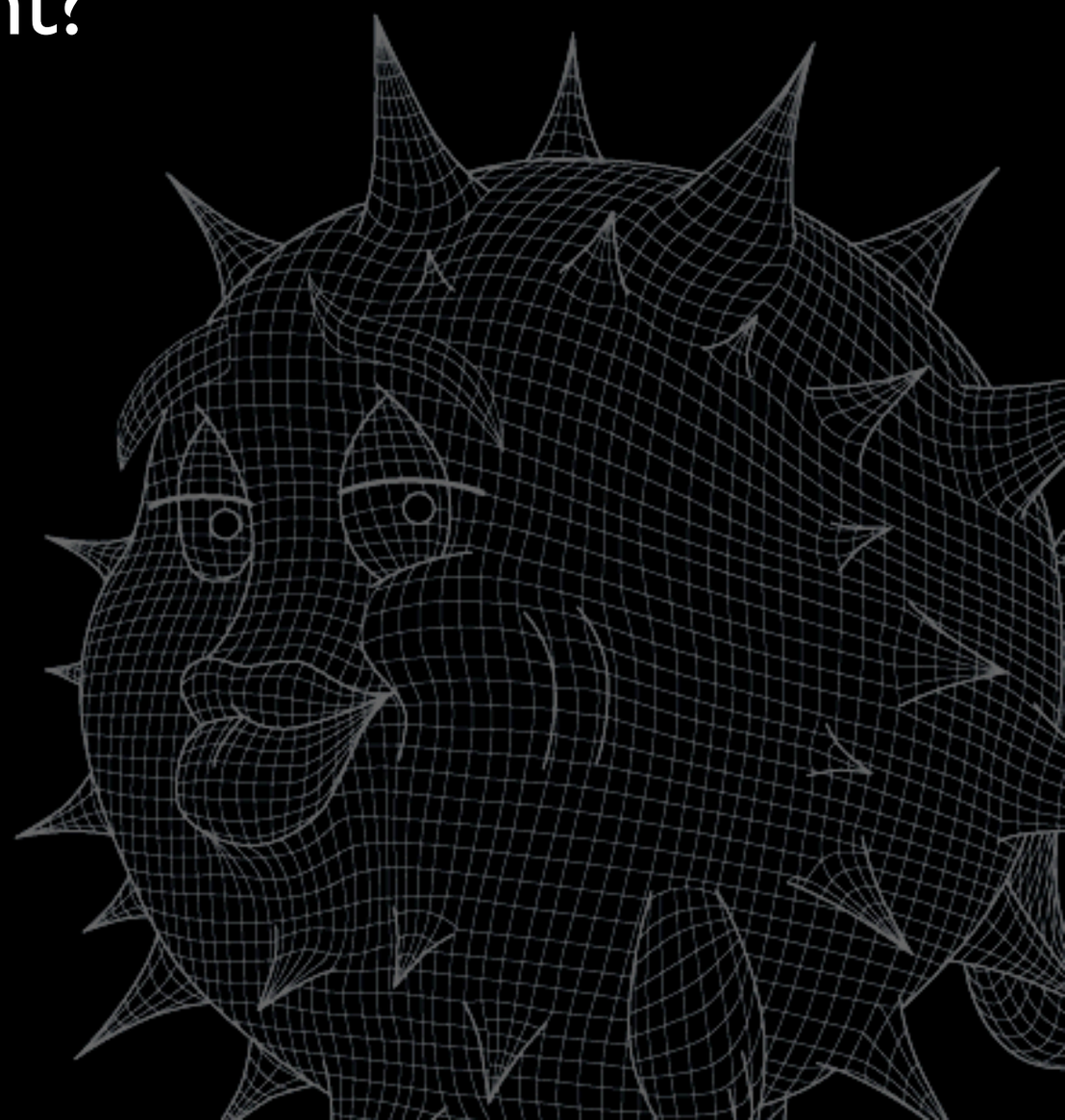
Marco Peereboom <marco@openbsd.org>
David Gwynne <dlg@openbsd.org>

OpenCON 2006



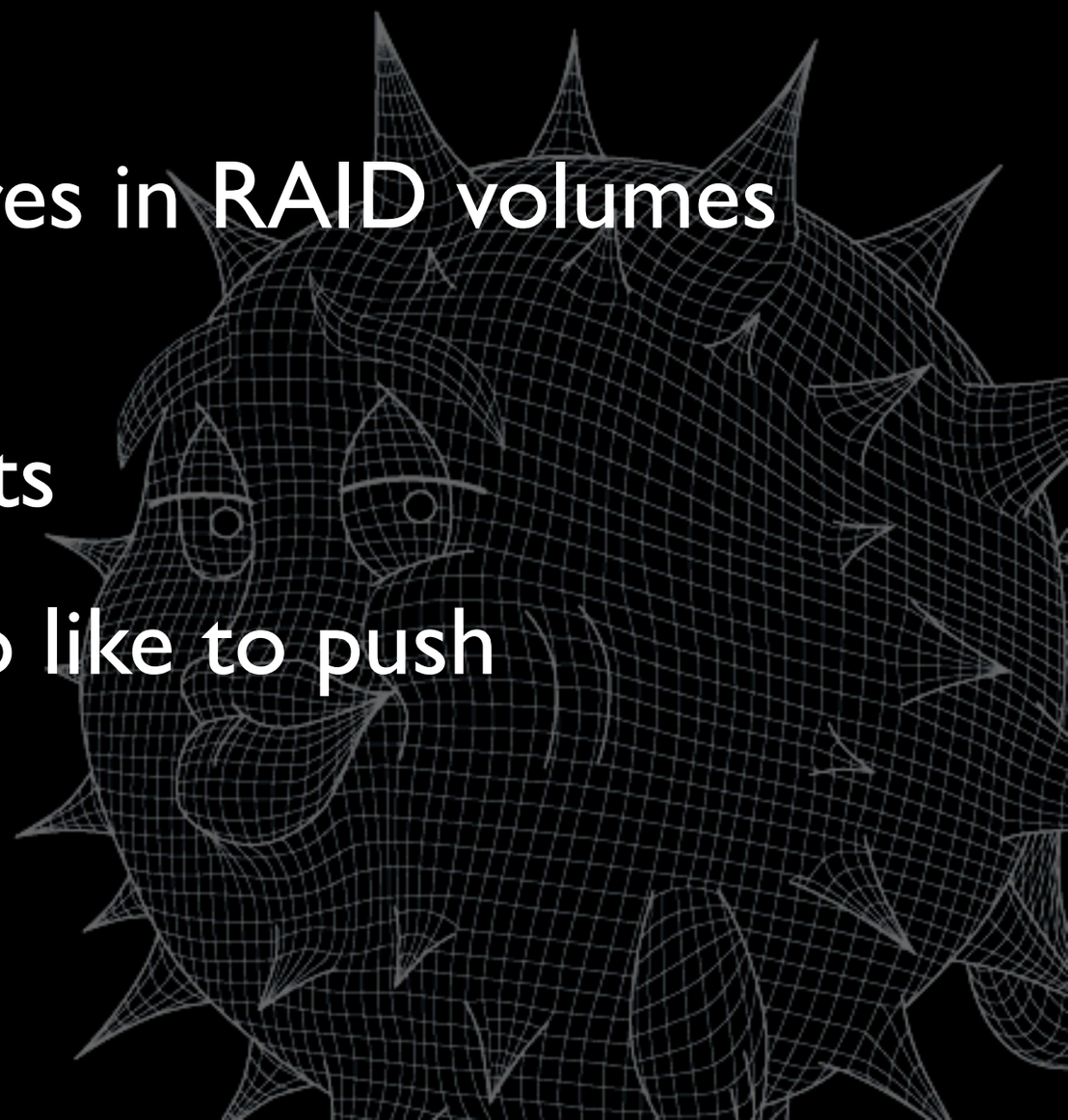
introduction

- what is RAID management?
- what are sensors?
- why do we care?
- what's the problem?
- what's the solution?



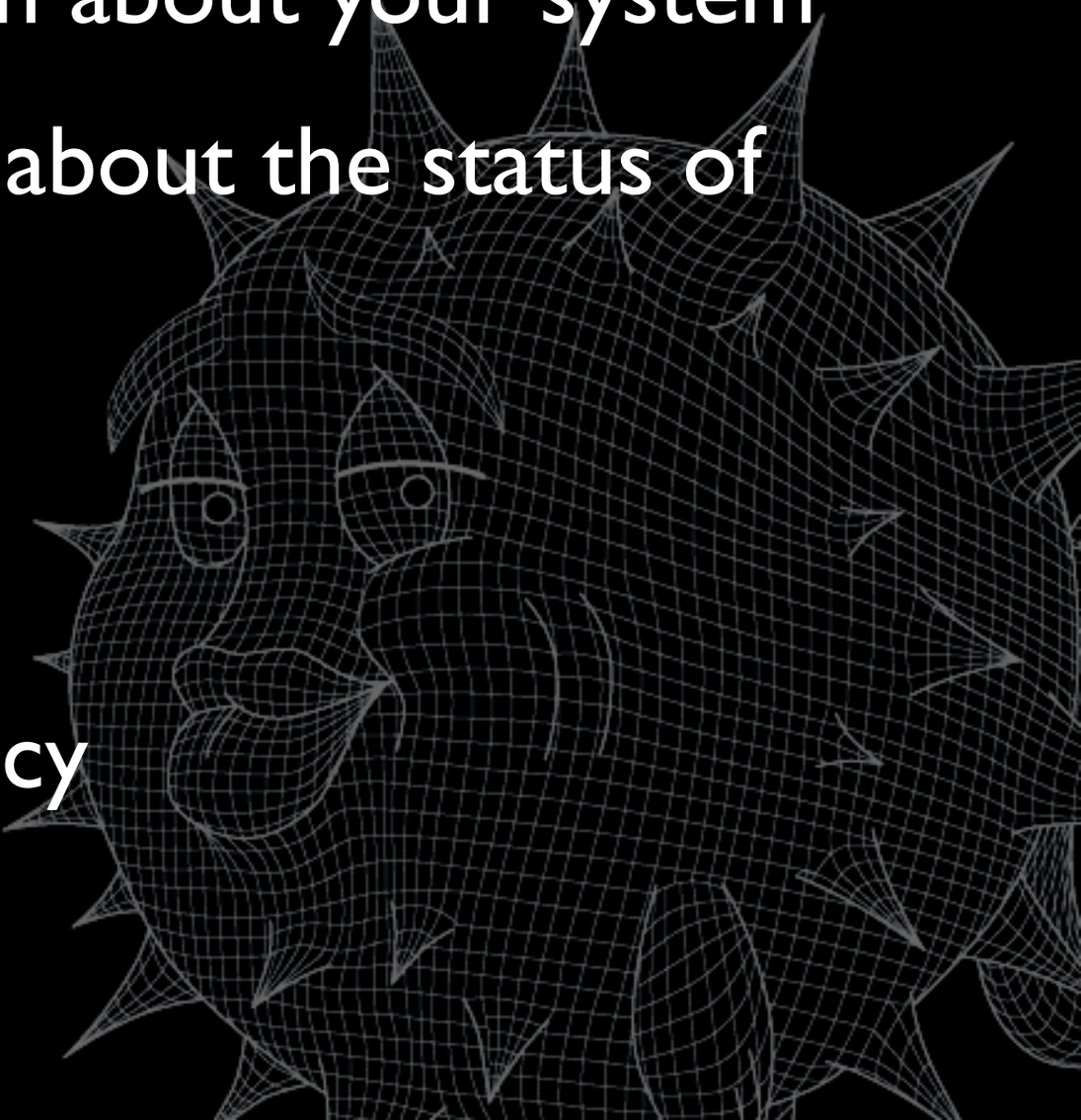
what is RAID management?

- the ability to see the configuration of RAID sets
- the ability to detect failures in RAID volumes and components
- the ability to fix RAID sets
- extra bits for people who like to push buttons



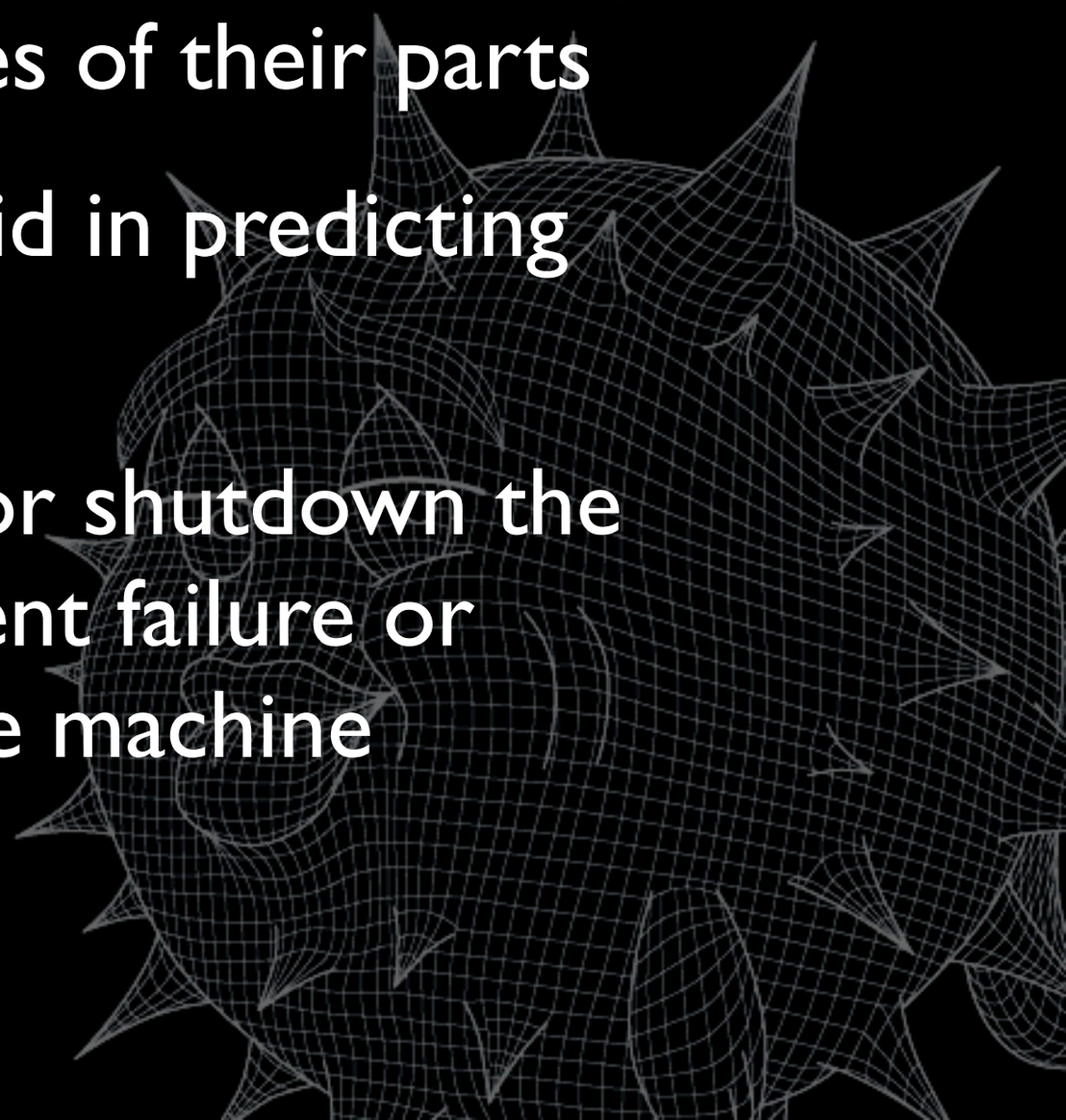
what are sensors?

- sensors are anything that provides environmental information about your system
- anything that can tell you about the status of your components, eg:
  - cpu temp and voltage
  - ambient temp
  - power supply redundancy



why do we care?

- computers are now built with redundancy so they can withstand failures of their parts
- environmental readings aid in predicting potential future failures
- we can replace the part or shutdown the machine before component failure or permanent damage to the machine

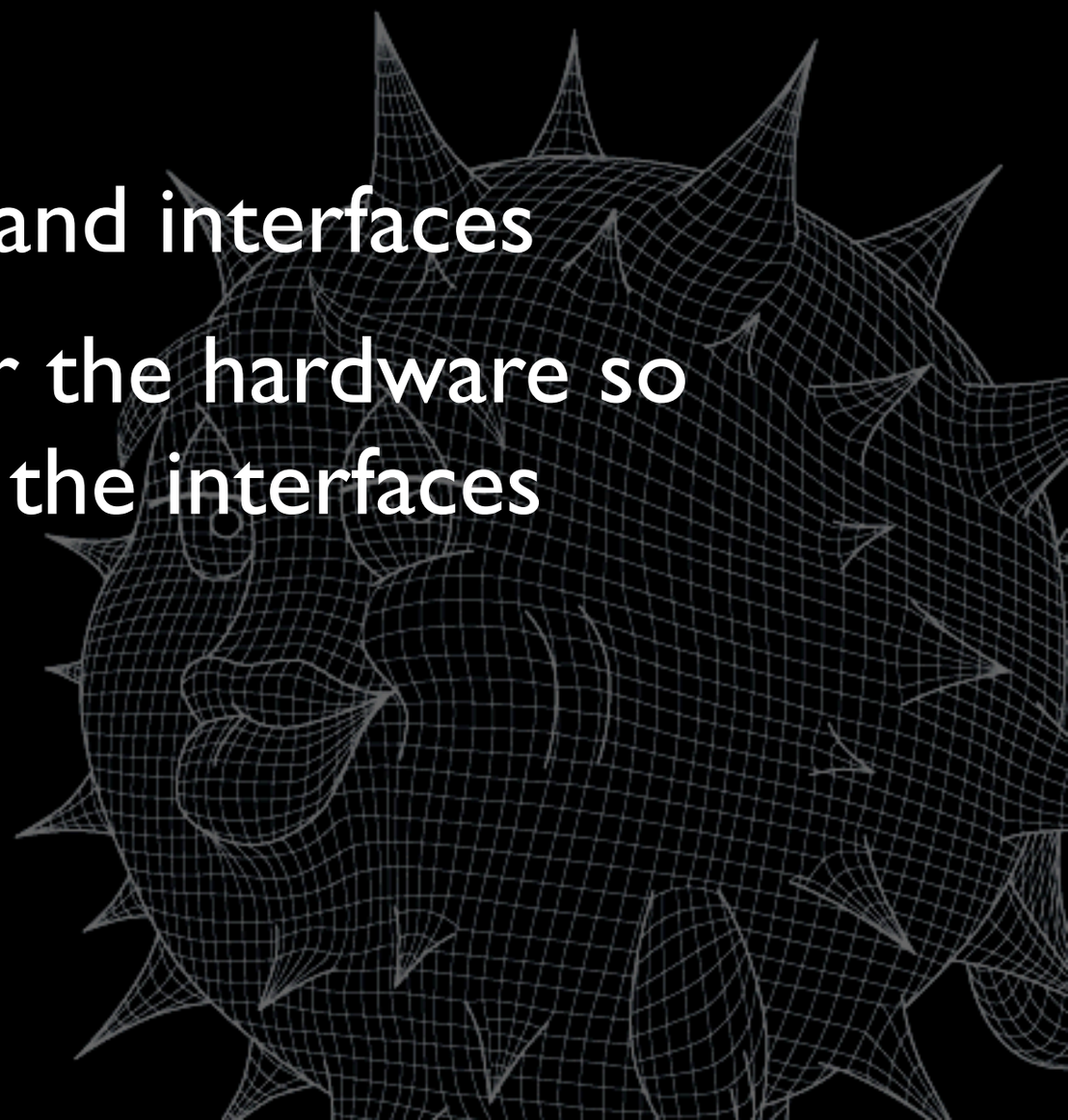


what's the problem?

- every vendor implements tools to manage raid devices and sensors differently
- these tools have evolved over the years into extremely complex and brittle stacks
- open source operating systems seem merely content to boot on the hardware and let the vendor provide the monitoring
- every implementation looks different

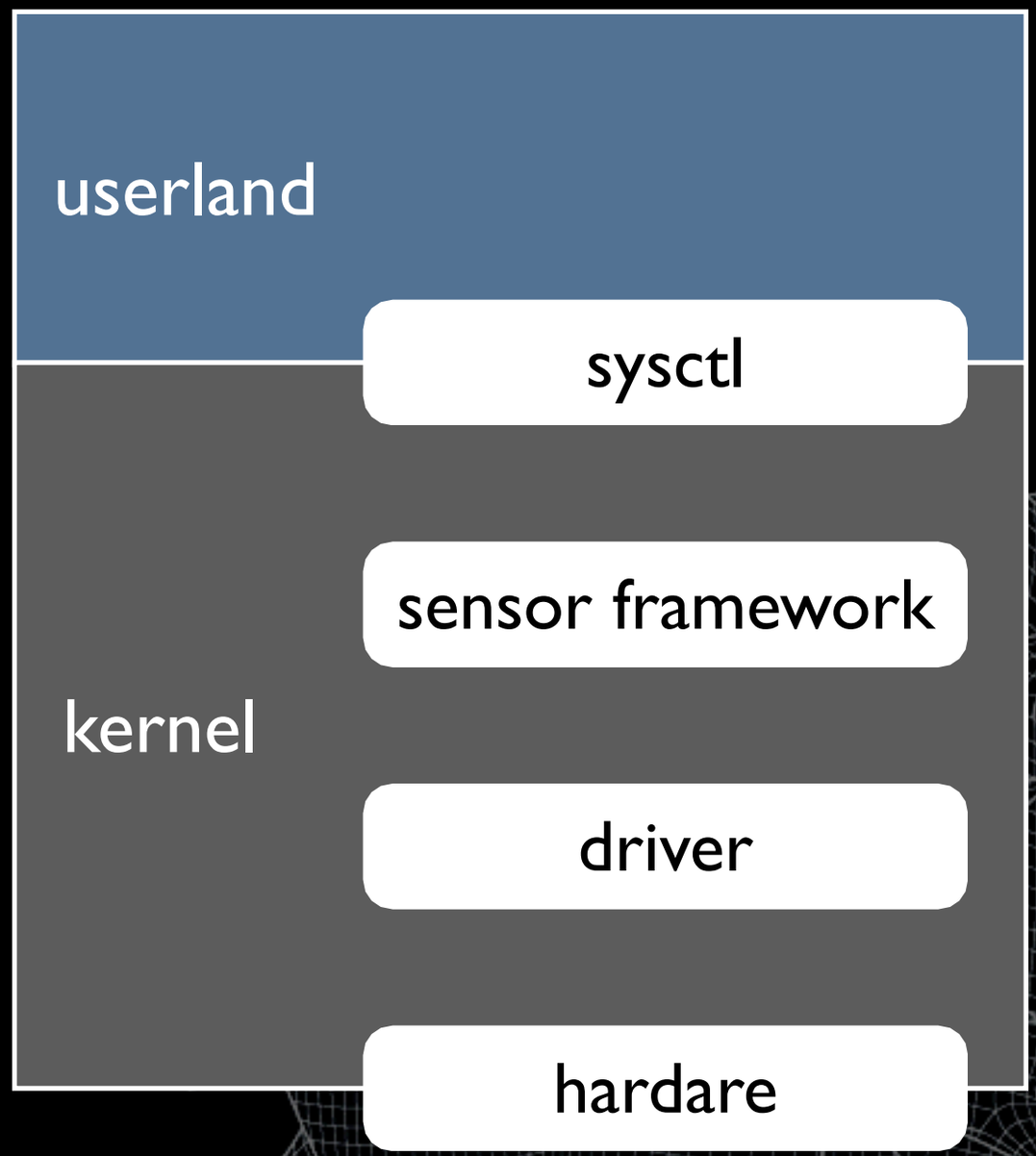
what's the solution?

- take some responsibility and make our own
- more specifically:
  - define your own stack and interfaces
  - get the specification for the hardware so you can fit drivers into the interfaces
  - write the code
  - give talks about it



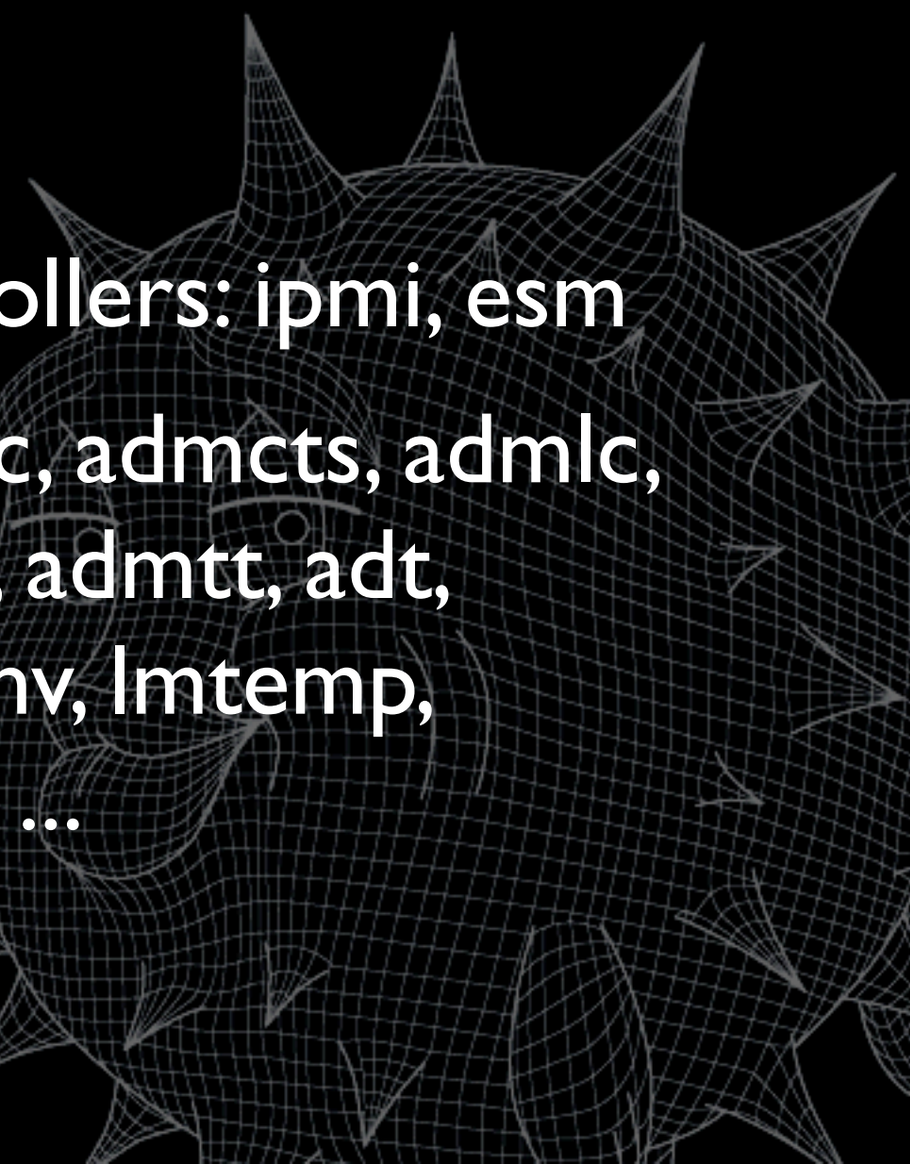
sensors in depth

- sensors are a stack made up of:
  - the hardware
  - the driver
  - the sensor framework
  - sysctl
- all the smarts are in the sensor framework



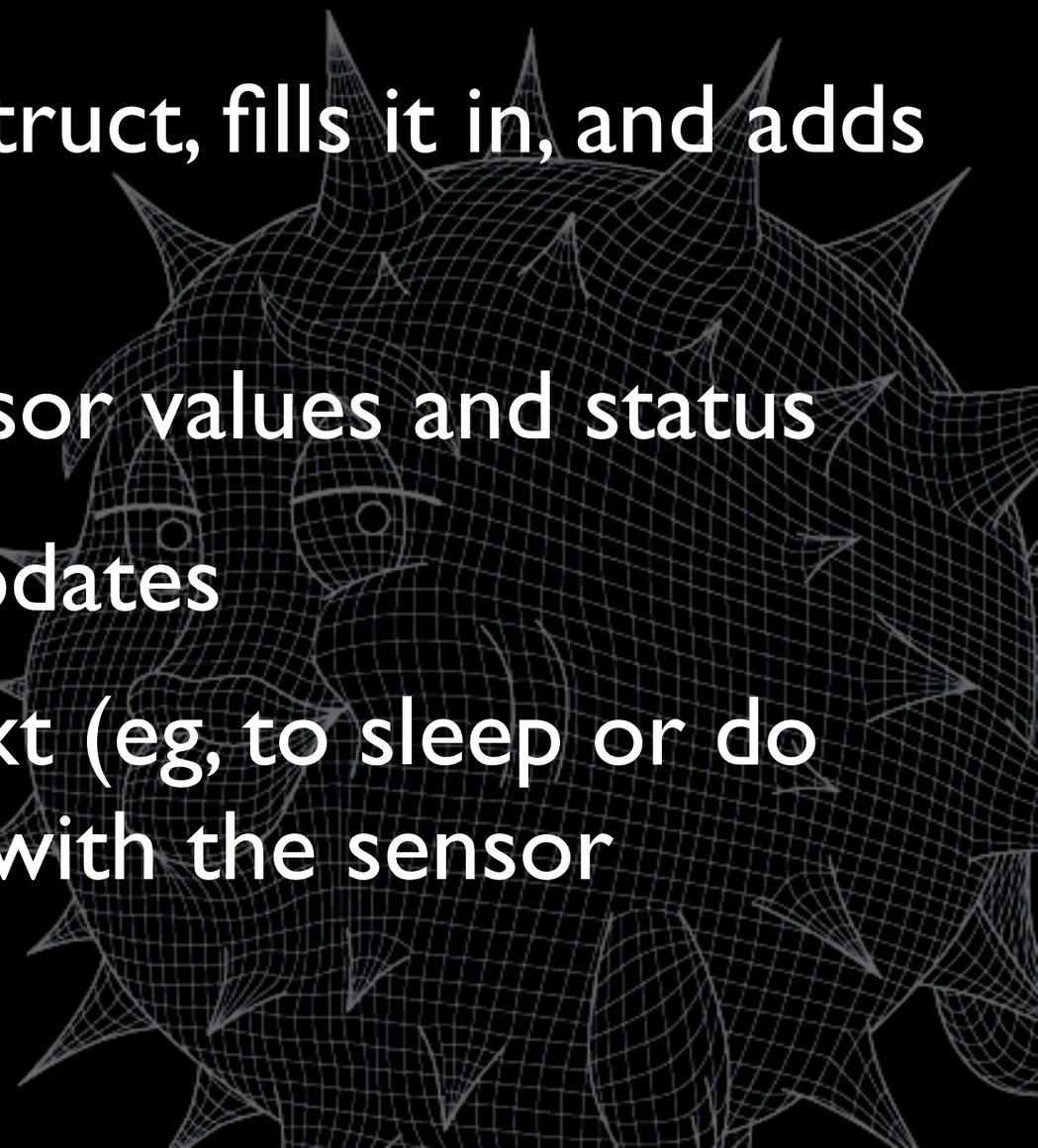
sensor hardware

- we've found a lot of sensors
  - SCSI enclosures: ses, safte
  - system management controllers: ipmi, esm
  - I2C and SMBus devices: adc, admcts, admlc, amdtemp, admtm, admtmp, admtt, adt, asbtm, asms, fcu, glenv, lmenv, lmtemp, maxds, maxtmp, pcfadc, tsl, ...



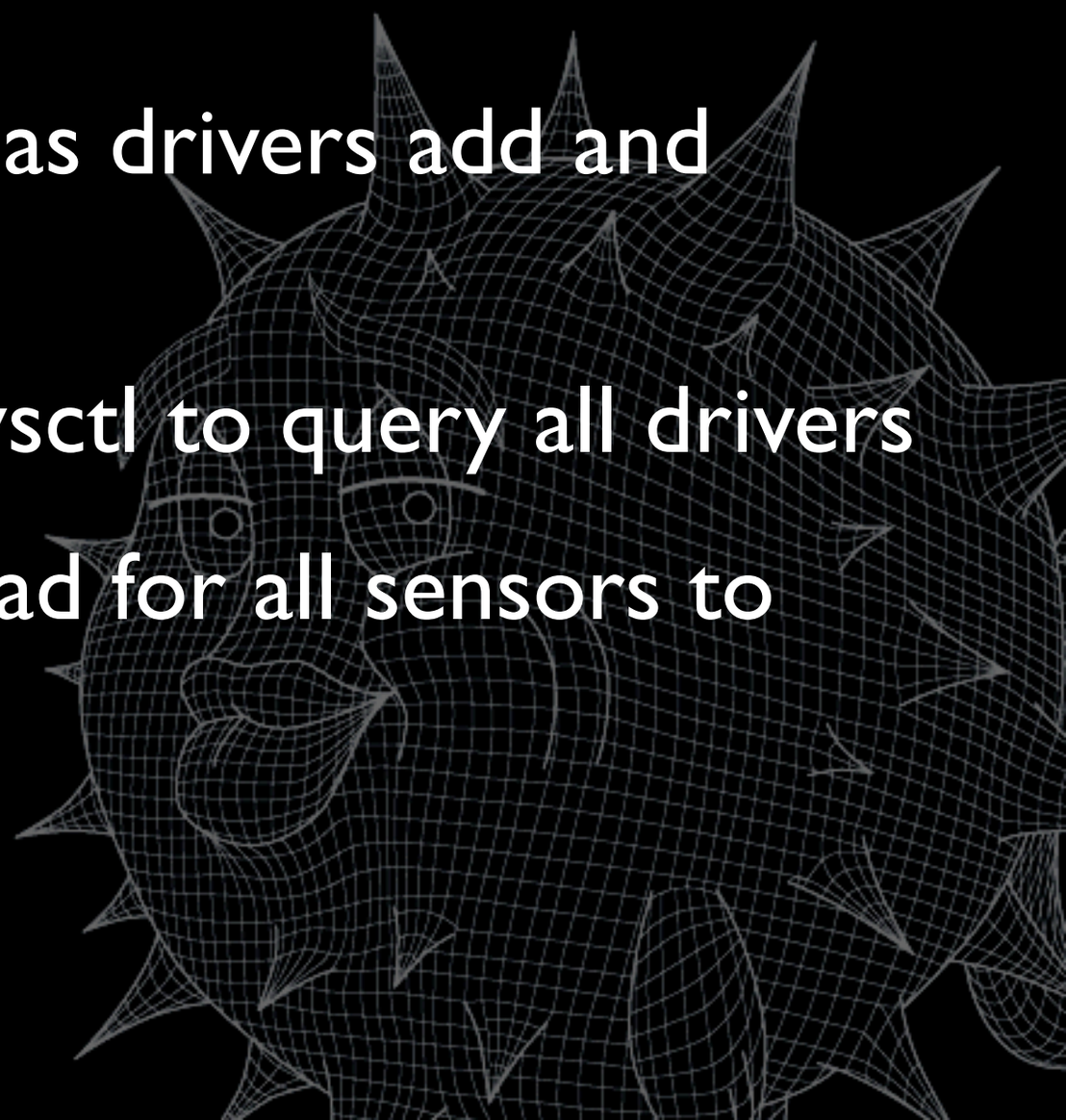
sensor drivers

- the driver is responsible for retrieving, interpreting, and normalising the sensor values off the hardware
- the driver allocates a sensor struct, fills it in, and adds it to the sensor-framework
- it periodically updates the sensor values and status
  - the driver can do its own updates
  - or if it needs process context (eg, to sleep or do DMA) it can register a task with the sensor framework



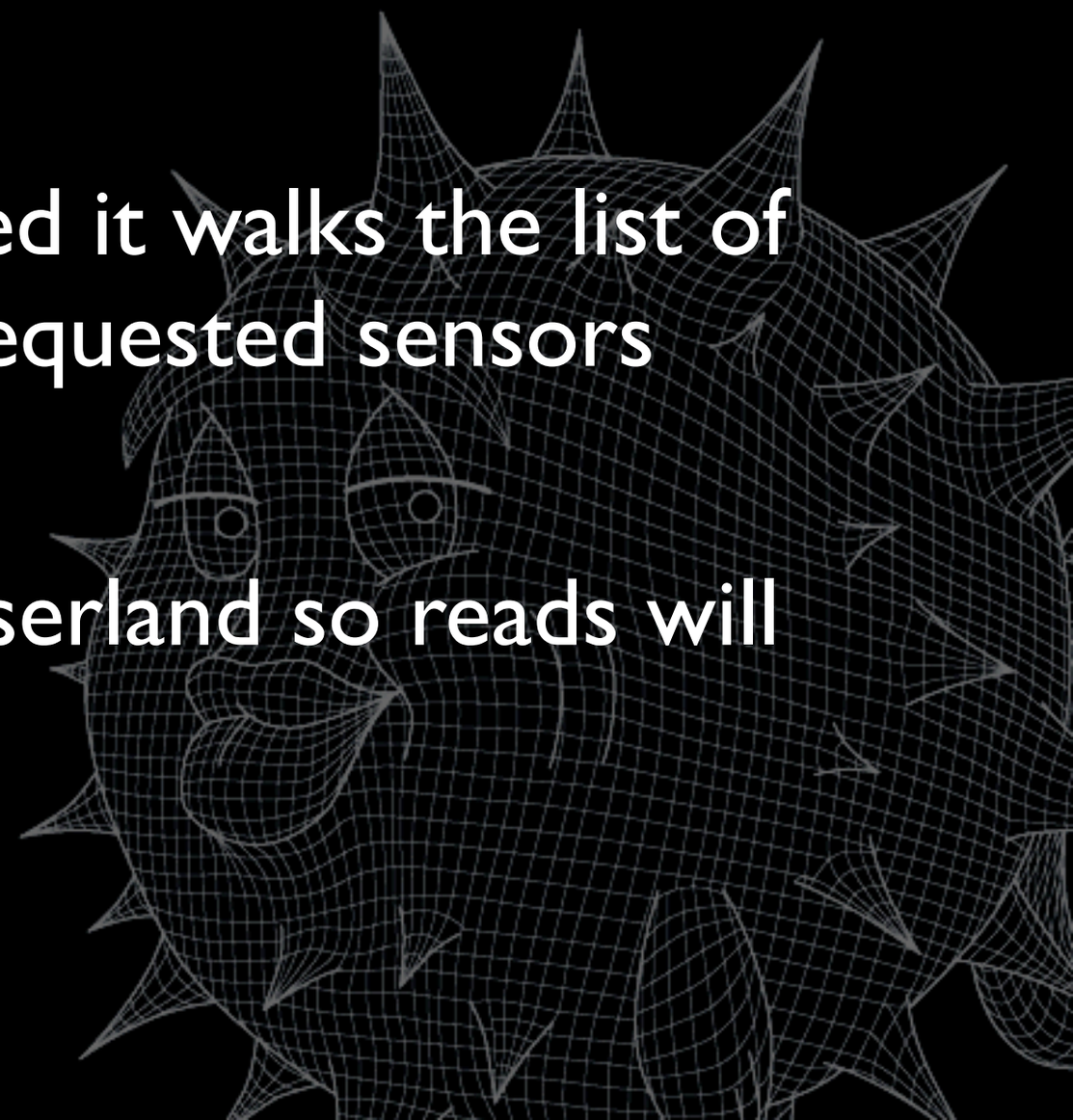
the sensor framework

- maintains the list of sensors as drivers add and remove entries
- provides a single place for sysctl to query all drivers
- provides a single kernel thread for all sensors to update out of via callbacks



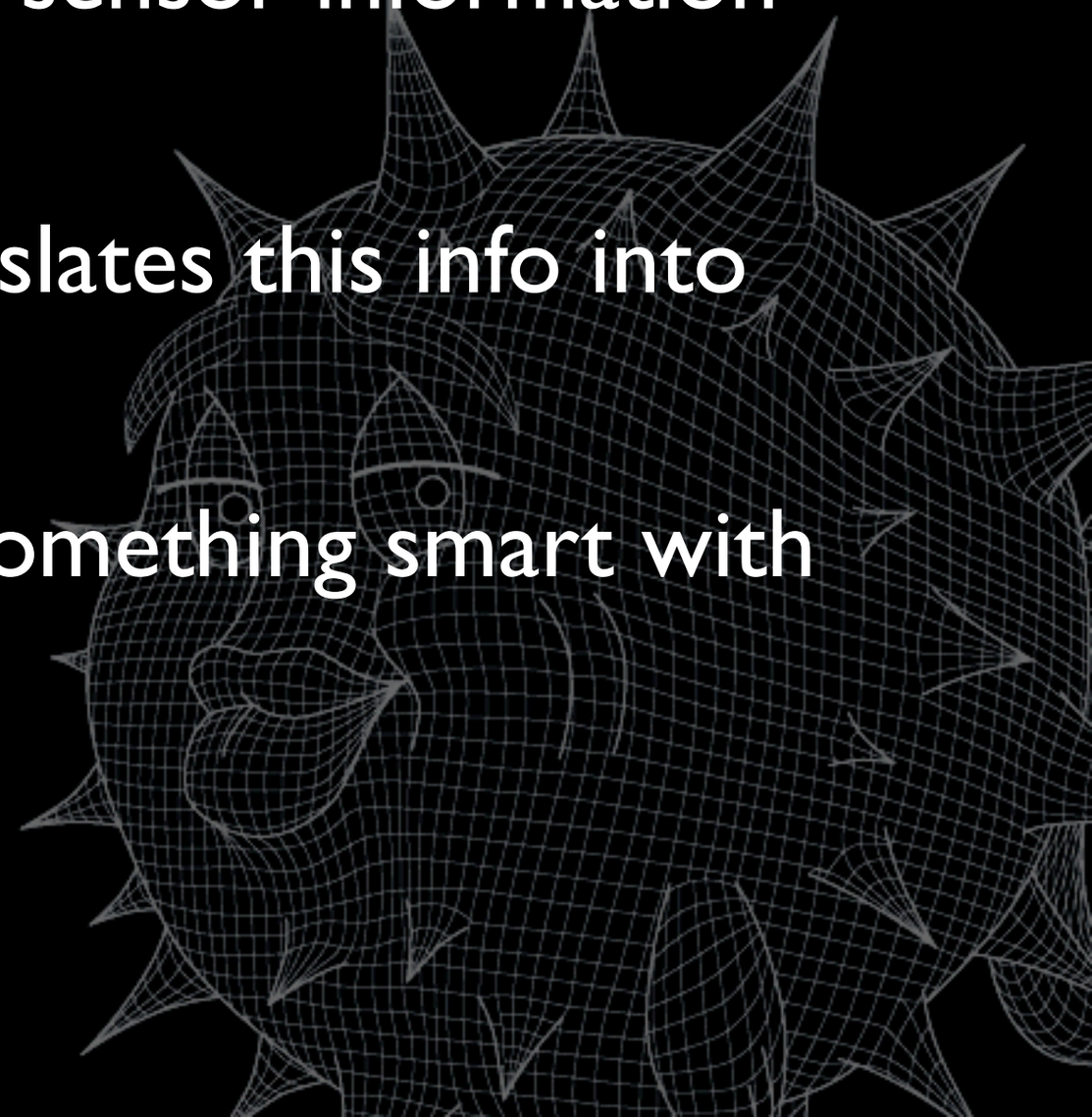
sysctl

- the sysctl interface is where userland and kernel meet
- when the kernel is queried it walks the list of sensors and copies the requested sensors struct out to userland
- decouples updates and userland so reads will not block



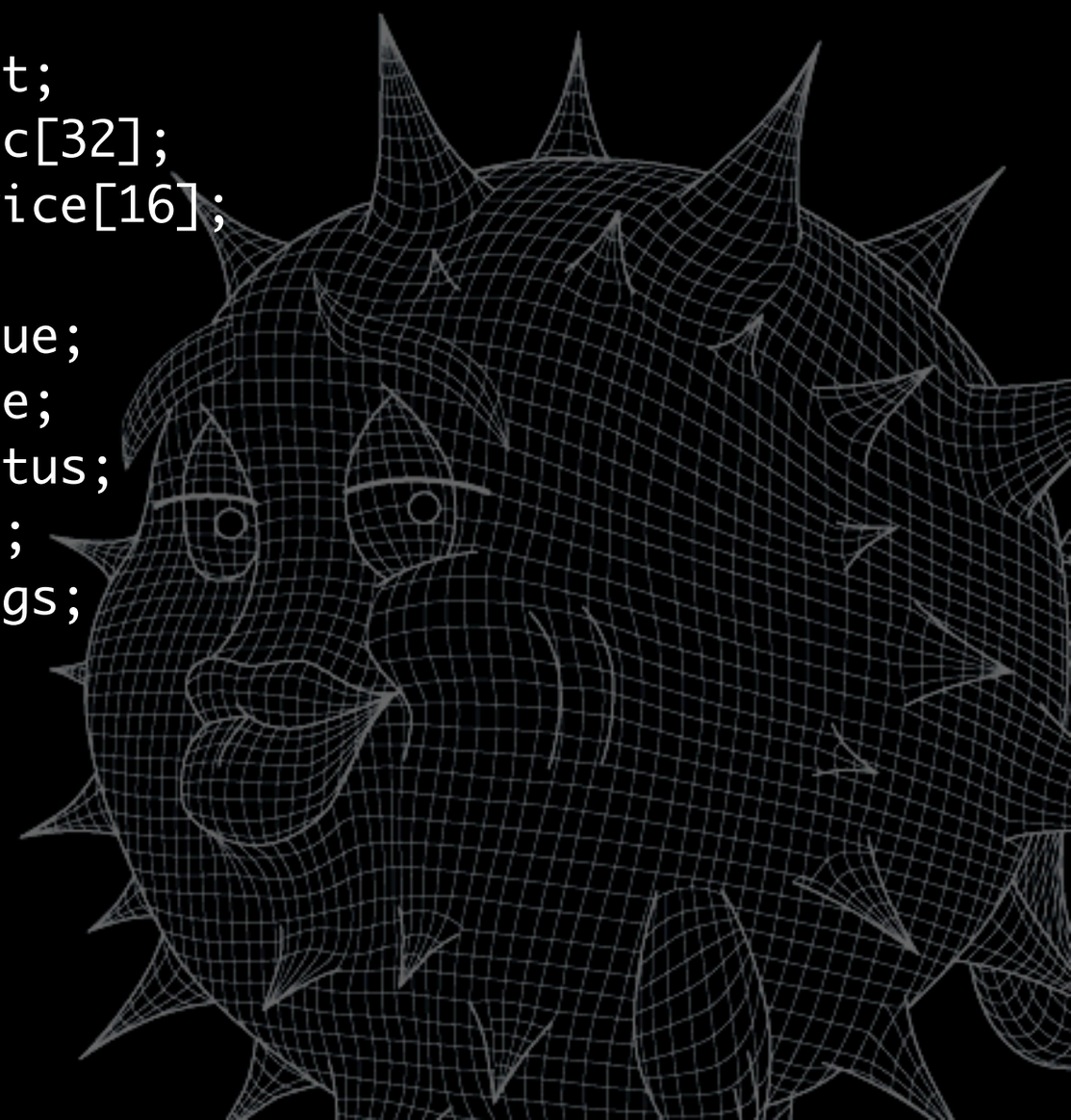
sensors in userland

- userland gets the kernels sensor information via sysctl(3)
- sysctl(8) fetches and translates this info into human readable output
- sensorsd(8) tries to do something smart with it



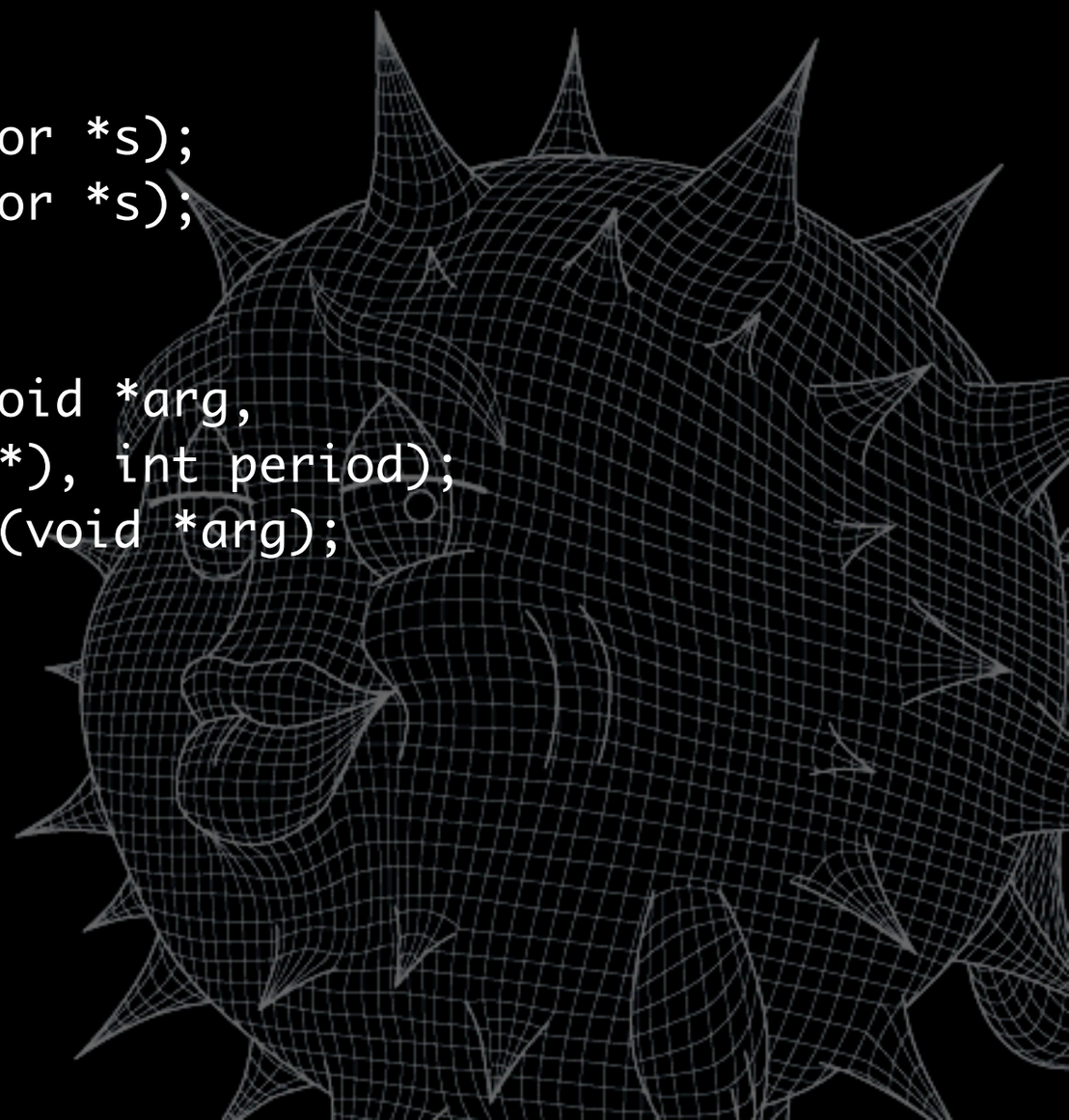
what a sensor looks like

```
struct sensor {
        SLIST_ENTRY(sensor)             list;
        char                            desc[32];
        char                            device[16];
        struct timeval                  tv;
        int64_t                         value;
        enum sensor_type                type;
        enum sensor_status              status;
        int                             num;
        int                             flags;
};
```



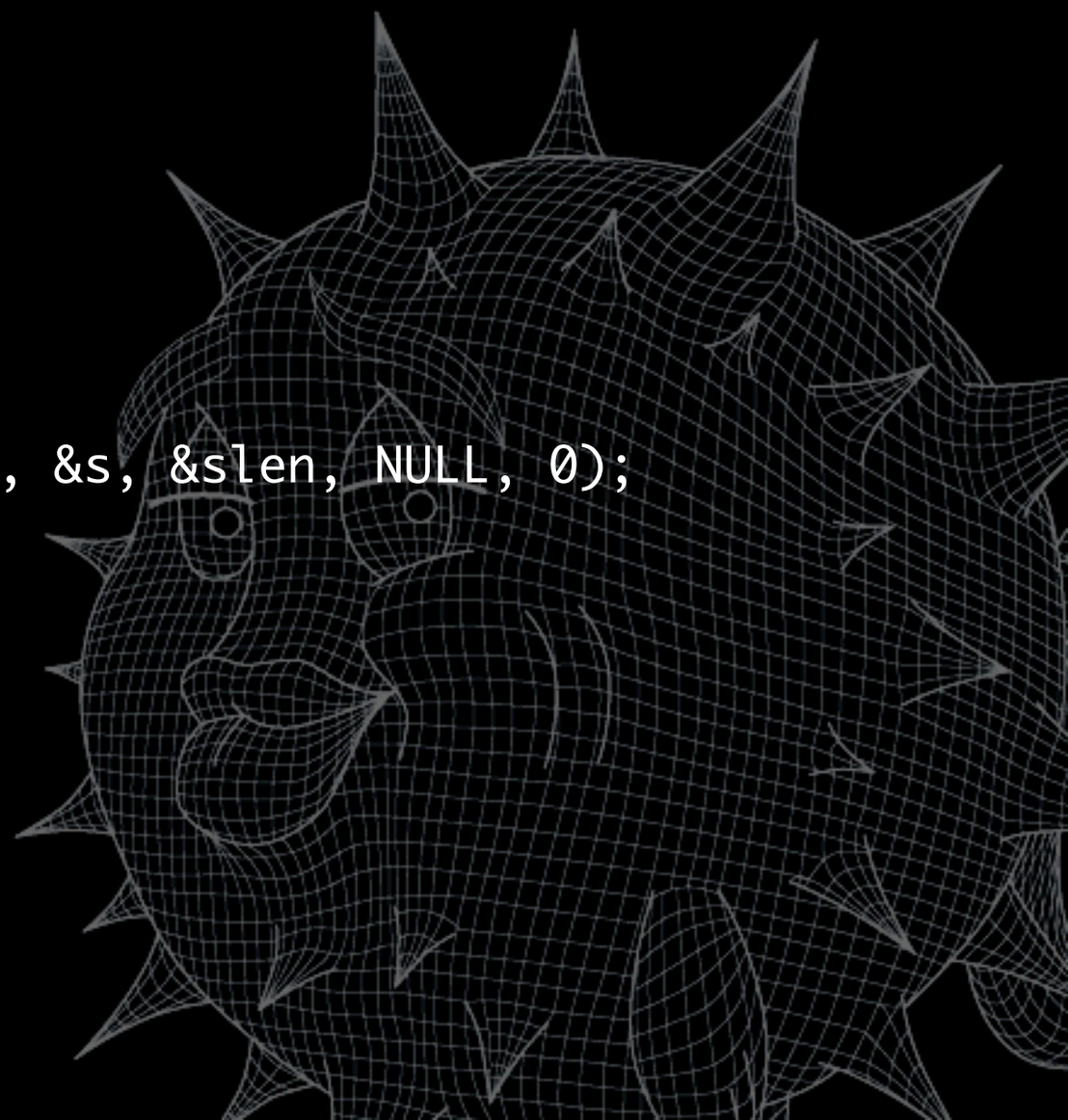
sensors in the kernel

```
void            sensor_add(struct sensor *s);
void            sensor_del(struct sensor *s);
struct sensor   *sensor_get(int id);

int             sensor_task_register(void *arg,
                    void (*func)(void *), int period);
void            sensor_task_unregister(void *arg);
```



sensors via sysctl(3)

```
int mib[] = { CTL_HW, HW_SENSORS, 0 };
struct sensor s;
size_t slen = sizeof(s);

sysctl(mib, sizeof(mib)/sizeof(mib[0]), &s, &slen, NULL, 0);
```

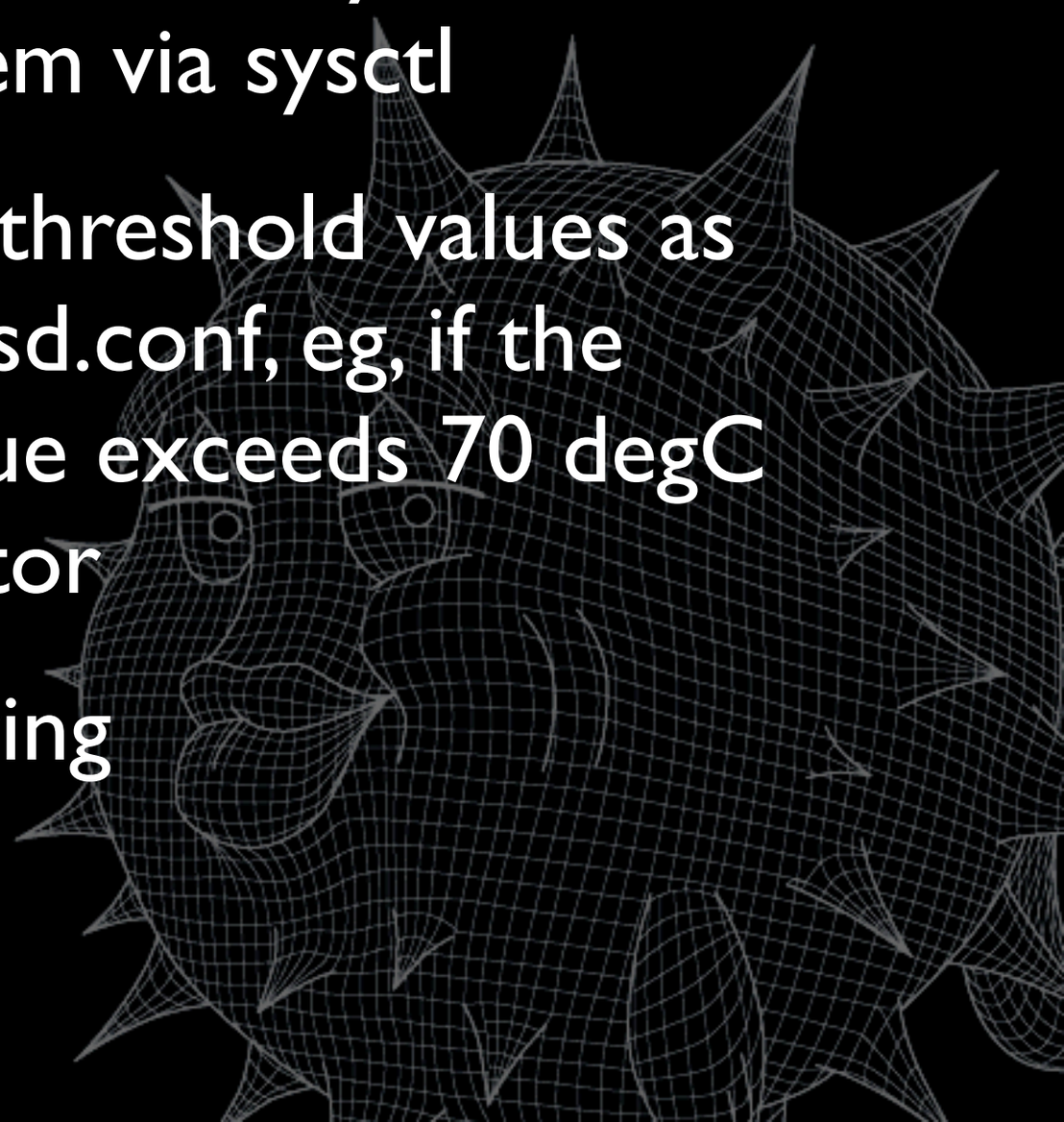


sensors via sysctl(8)

```
# sysctl hw.sensors
hw.sensors.0=ipmi0, Phys. Security, On, CRITICAL
hw.sensors.1=ipmi0, Baseboard 1.5V, 1.51 V DC, OK
hw.sensors.2=ipmi0, Baseboard 2.5V, 2.51 V DC, OK
hw.sensors.3=ipmi0, Baseboard 3.3V, 3.34 V DC, OK
hw.sensors.4=ipmi0, Baseboard 3.3Vsb, 3.49 V DC, OK
hw.sensors.5=ipmi0, Baseboard 5V, 5.10 V DC, OK
hw.sensors.6=ipmi0, Baseboard 12V, 12.10 V DC, OK
hw.sensors.7=ipmi0, Baseboard -12V, -12.30 V DC, OK
hw.sensors.8=ipmi0, Battery Voltage, 3.14 V DC, OK
hw.sensors.9=ipmi0, Processor VRM, 1.47 V DC, OK
hw.sensors.10=ipmi0, Baseboard Temp, 30.00 degC, OK
hw.sensors.11=ipmi0, Processor 1 Temp, 36.00 degC, OK
hw.sensors.13=ipmi0, Baseboard Fan 1, 1980 RPM, OK
hw.sensors.14=ipmi0, Baseboard Fan 2, 2100 RPM, OK
...
```

sensorsd

- sensorsd polls the sensor values by periodically retrieving them via sysctl
- sensorsd can react upon threshold values as configured in /etc/sensorsd.conf, eg, if the ambient temperature value exceeds 70 degC then page the administrator
- currently awful, it is evolving



sensors summary

- sensors are not magical, they're generally very simple, ie, read a value off hardware and stash it in a struct
- the same framework is enabled on all our archs
- sensors are easy (and fun, like blinkenlights) to implement and use

RAID management

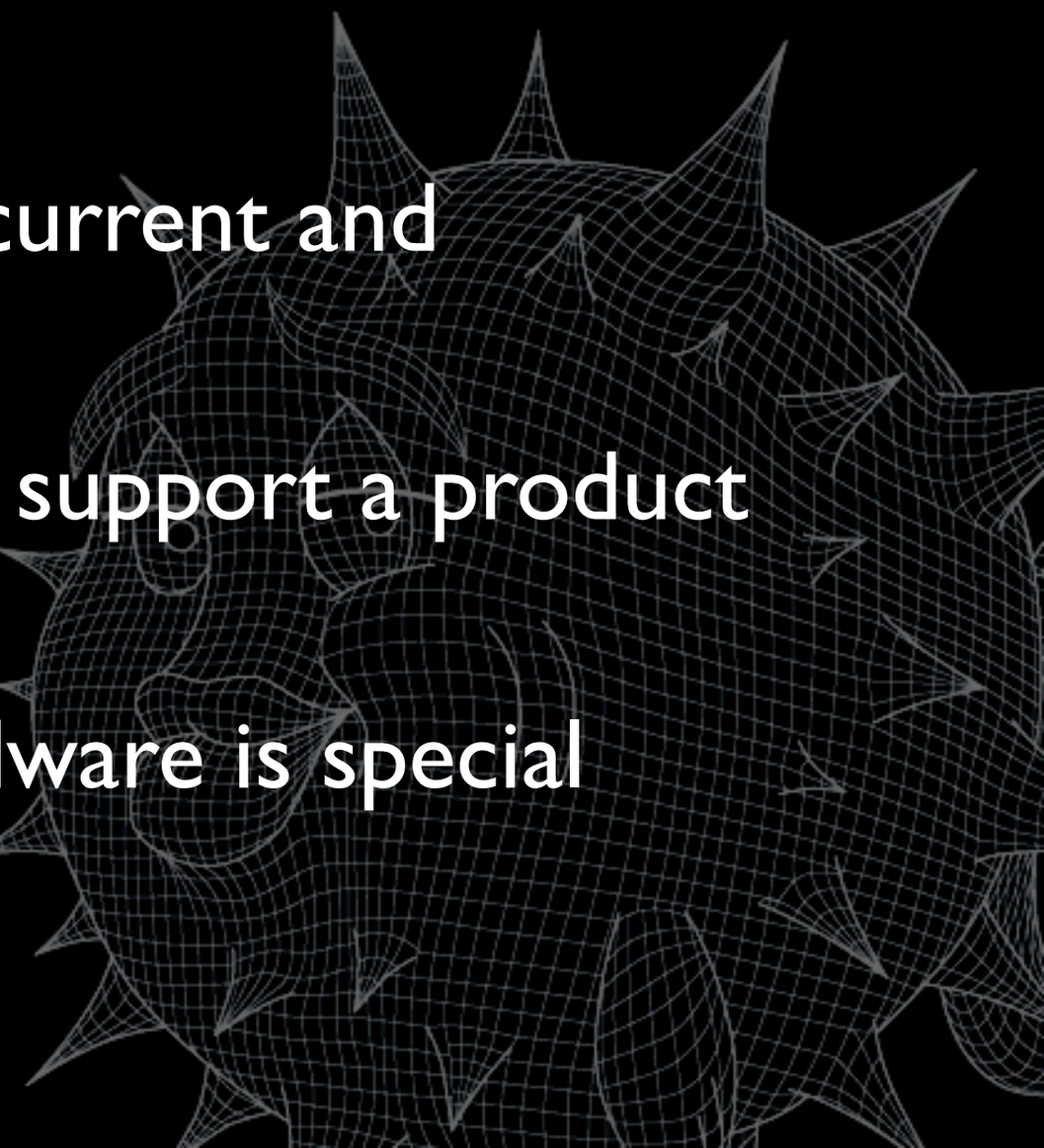
- similar to sensors in that we want to see the status of redundant components
- different to sensors in that we need to do more, eg, replace disks and add spares
- hard to do because vendors don't want to give up documentation
- vendors do provide tools, but...

vendor tools

- binary only, and limited to specific archs (i386, and whatever can run i386 binaries)
- requires us to open big holes in the kernel for userland to talk directly to the hardware (and hasn't that worked so well for X?)
- provided under incompatible licenses, so can't be shipped in the base system
- therefore not supported on OpenBSD

RAID documentation

- attempts to obtain documentation have failed for several reasons
  - Vendors do not posses current and accurate documentation
  - Vendors do not want to support a product beyond regular channels
  - Vendors think their hardware is special

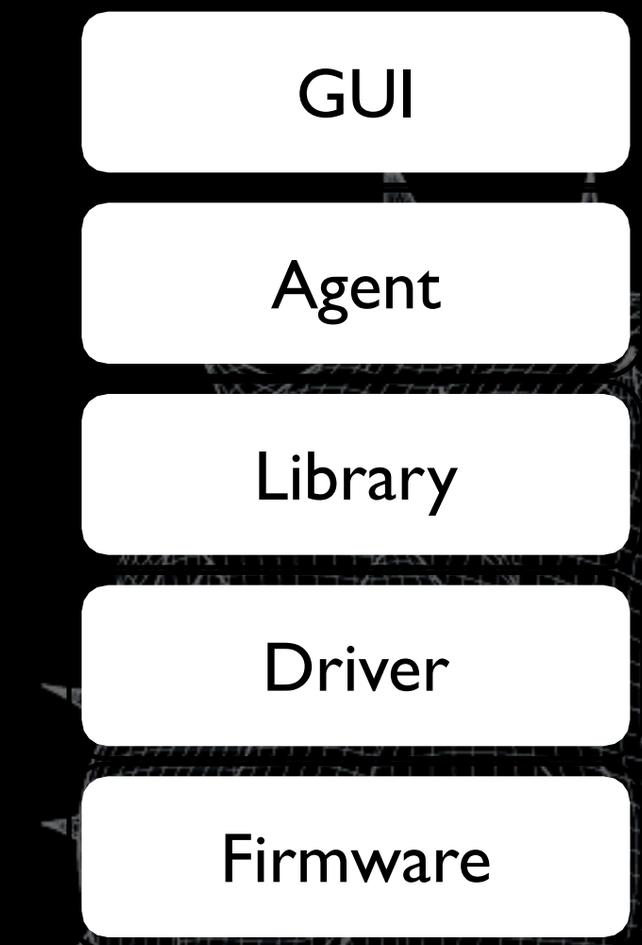


typical RAID management stack

- typically developed by different teams resulting in large amounts of abstraction

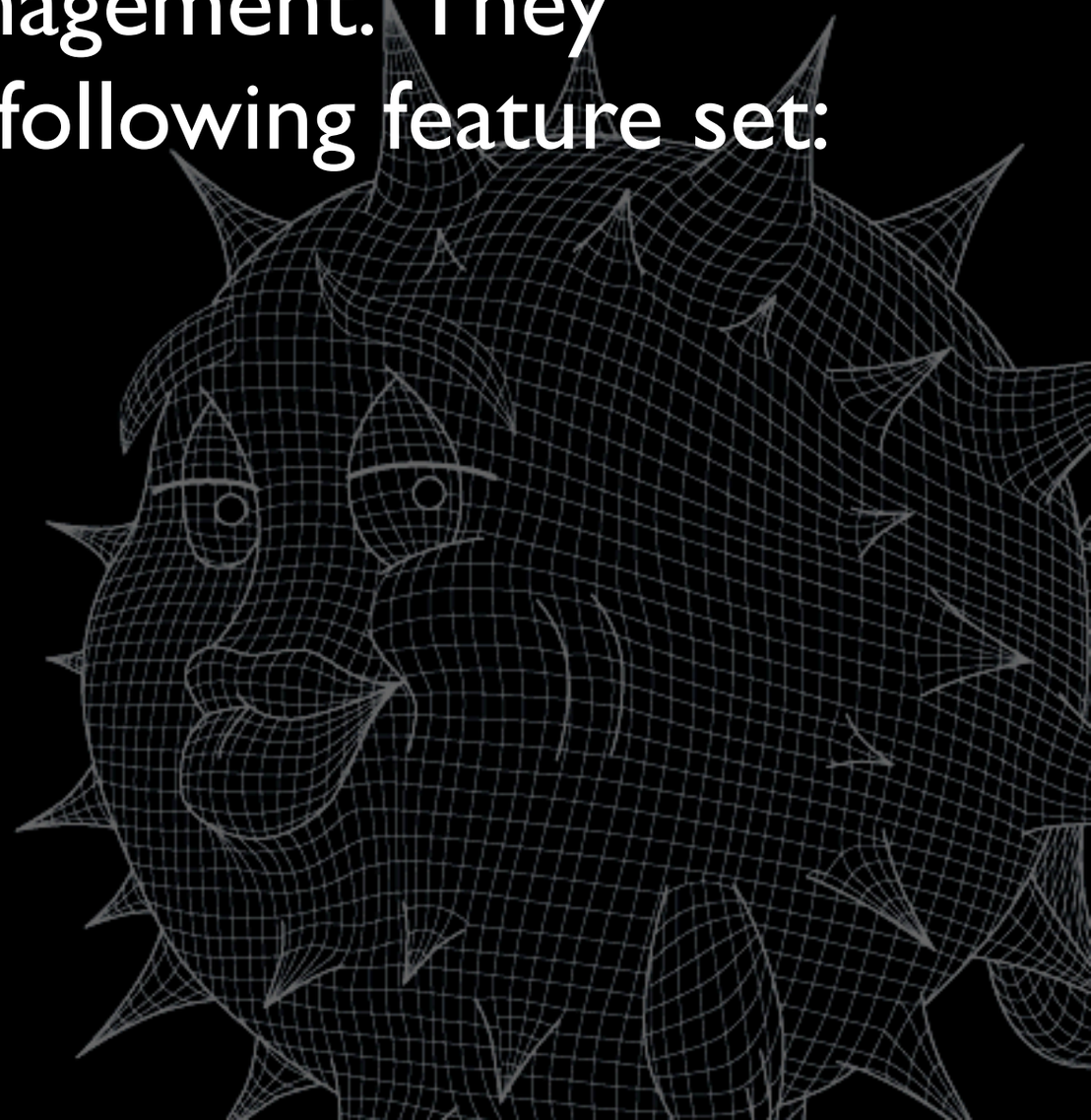
- the abstraction leads to bugs (more code always has more places for bugs to hide)

- different vendors have their own stacks



RAID management essentials

- production machines do not need complex tool chains for RAID management. They essentially only need the following feature set:
  - alerts
  - monitoring
  - inquiry
  - recovery operations



OpenBSD RAID management

bioctl

bio(4)

sensors

Inquiry
&
Recovery

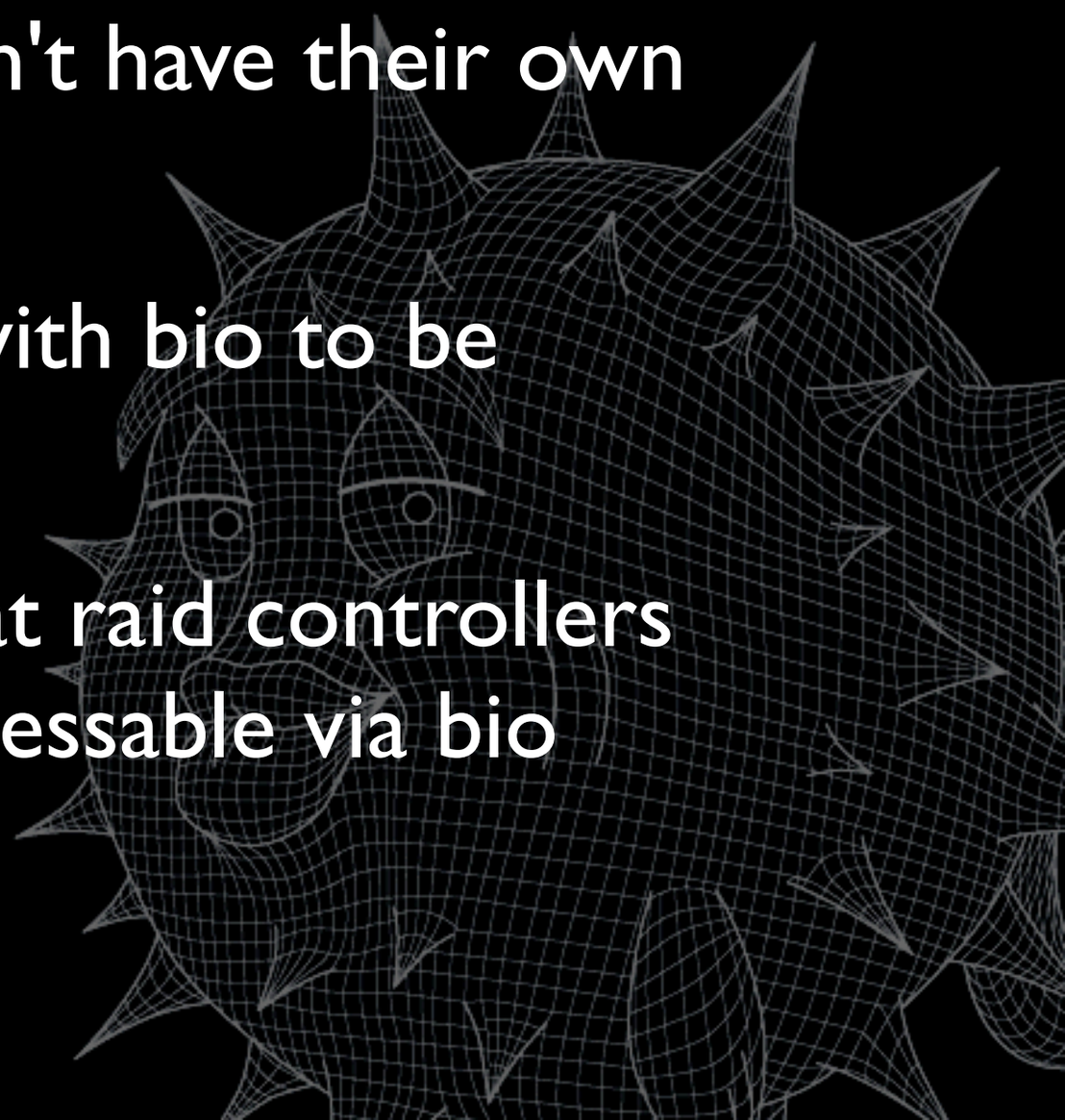
Driver

Firmware

Alerting
&
Monitoring

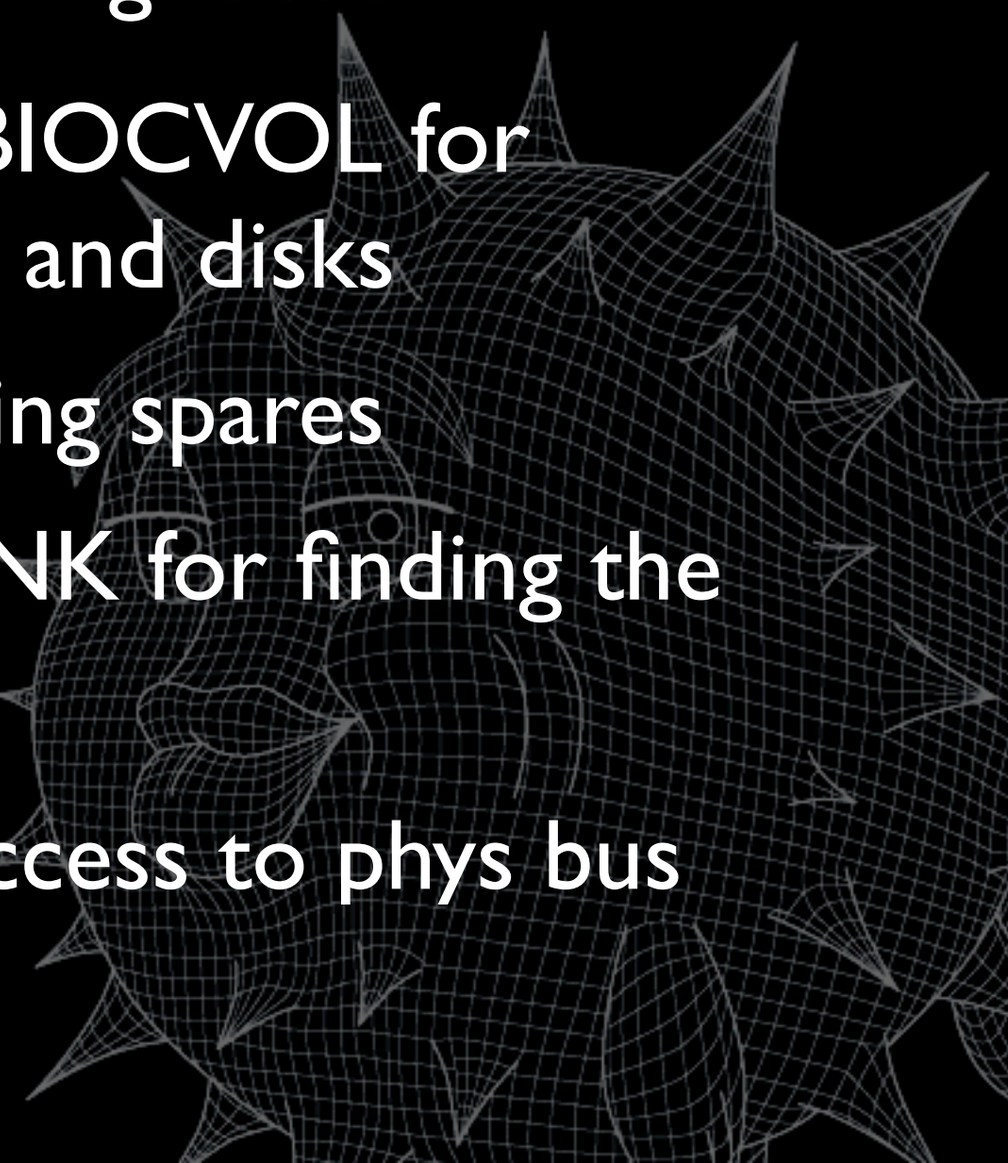
bio(4)

- technically it is a pseudo device that tunnels ioctls for devices that don't have their own /dev entry
- drivers have to register with bio to be accessible via bio
- we define some ioctls that raid controllers can implement that is accessable via bio



bio inside drivers

- In order to support bio drivers need to support some of the following ioctls:
  - BIOCINQ, BIOCDISK, BIOCVOL for enumeration of volumes and disks
  - BIOCSETSTATE for adding spares
  - BIOCALARM, BIOCBLINK for finding the computer and the disks
- need a pass thru bus for access to phys bus



bioctl

- bioctl is the userland half of our RAID management tool
- intended as the ifconfig of RAID controllers
- it translates the bio ioctls into something humans can grok

bioctl

- inquiry functions:
  - display RAID setup and status
  - blink enclosure slot so you can find it
- recovery functions:
  - alarm management
  - create hot-spare
  - rebuild to hot-spare

bioctl in action

```
# bioctl ami0
Volume  Status     Size           Device
 ami0 0 Online       366372454400 sd0     RAID5
      0 Online        73403465728 0:0.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:2.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:4.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Online        73403465728 0:8.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:10.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:12.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 1 Online       366372454400 sd1     RAID5
      0 Online        73403465728 0:1.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:3.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:5.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Online        73403465728 1:9.0   ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:11.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:13.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 2 Unused         73403465728 1:14.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 3 Hot spare      73403465728 1:15.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
```

bioctl when we pull a disk

```
# bioctl ami0
Volume  Status     Size           Device
 ami0 0 Online       366372454400 sd0     RAID5
      0 Online        73403465728 0:0.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:2.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:4.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Online        73403465728 0:8.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:10.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:12.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 1 Degraded     366372454400 sd1     RAID5 11% done
      0 Online        73403465728 0:1.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:3.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:5.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Rebuild       73403465728 1:15.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:11.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:13.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 2 Unused         73403465728 1:14.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
```

bioctl when we return the disk

```
# bioctl ami0
Volume  Status     Size           Device
 ami0 0 Online       366372454400 sd0     RAID5
      0 Online        73403465728 0:0.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:2.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:4.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Online        73403465728 0:8.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:10.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:12.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 1 Degraded     366372454400 sd1     RAID5 57% done
      0 Online        73403465728 0:1.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:3.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:5.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Rebuild       73403465728 1:15.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:11.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:13.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 2 Unused        73403465728 1:9.0   ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 3 Unused        73403465728 1:14.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
```

bioctl when we make it a spare

```
# bioctl -H 1:9 ami0
# bioctl ami0
Volume  Status     Size            Device
 ami0 0 Online       366372454400 sd0     RAID5
      0 Online        73403465728 0:0.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:2.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:4.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Online        73403465728 0:8.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:10.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:12.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 1 Degraded     366372454400 sd1     RAID5 60% done
      0 Online        73403465728 0:1.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      1 Online        73403465728 0:3.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      2 Online        73403465728 0:5.0   ses0   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      3 Rebuild       73403465728 1:15.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      4 Online        73403465728 1:11.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
      5 Online        73403465728 1:13.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 2 Hot spare      73403465728 1:9.0   ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
ami0 3 Unused         73403465728 1:14.0  ses1   <MAXTOR  ATLAS15K2_73SCA JNZ6>
```

other bioctl magic

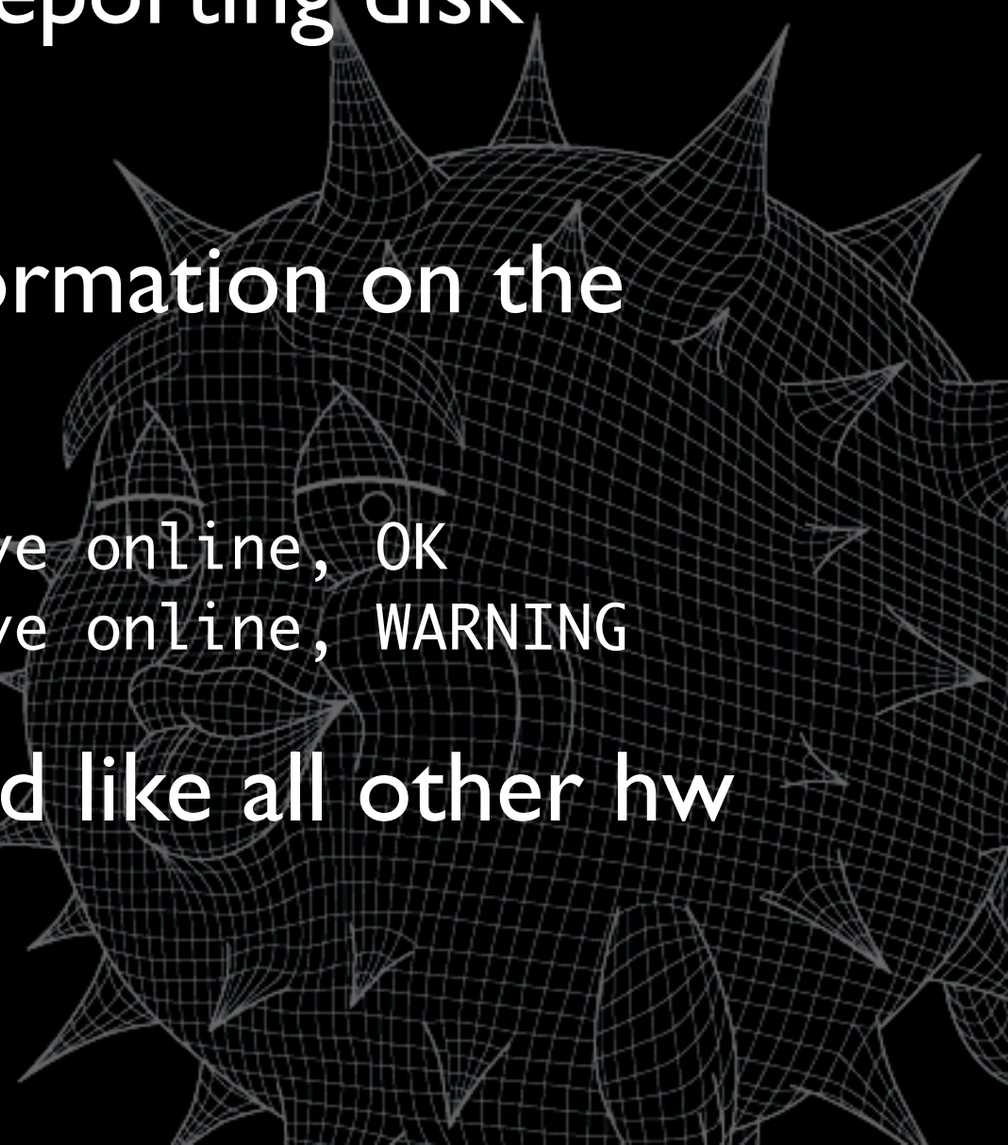
- help! i am bleeding from the ears (or waking people up when testing at 1am)!
  - Disable the alarm with:
    ```
    # bioctl -a quiet ami0
    ```
- help! show me the disk i need to replace!
  - Blink it with:
    ```
    # bioctl -b 1.9 ami0
    ```

RAID and sensors

- along with temperatures and voltages, we have a type of sensor for reporting disk status
- provides near realtime information on the health of a RAID disk:

```
hw.sensors.0=ami0, sd0, drive online, OK
hw.sensors.1=ami0, sd1, drive online, WARNING
```

- raid disks can be monitored like all other hw

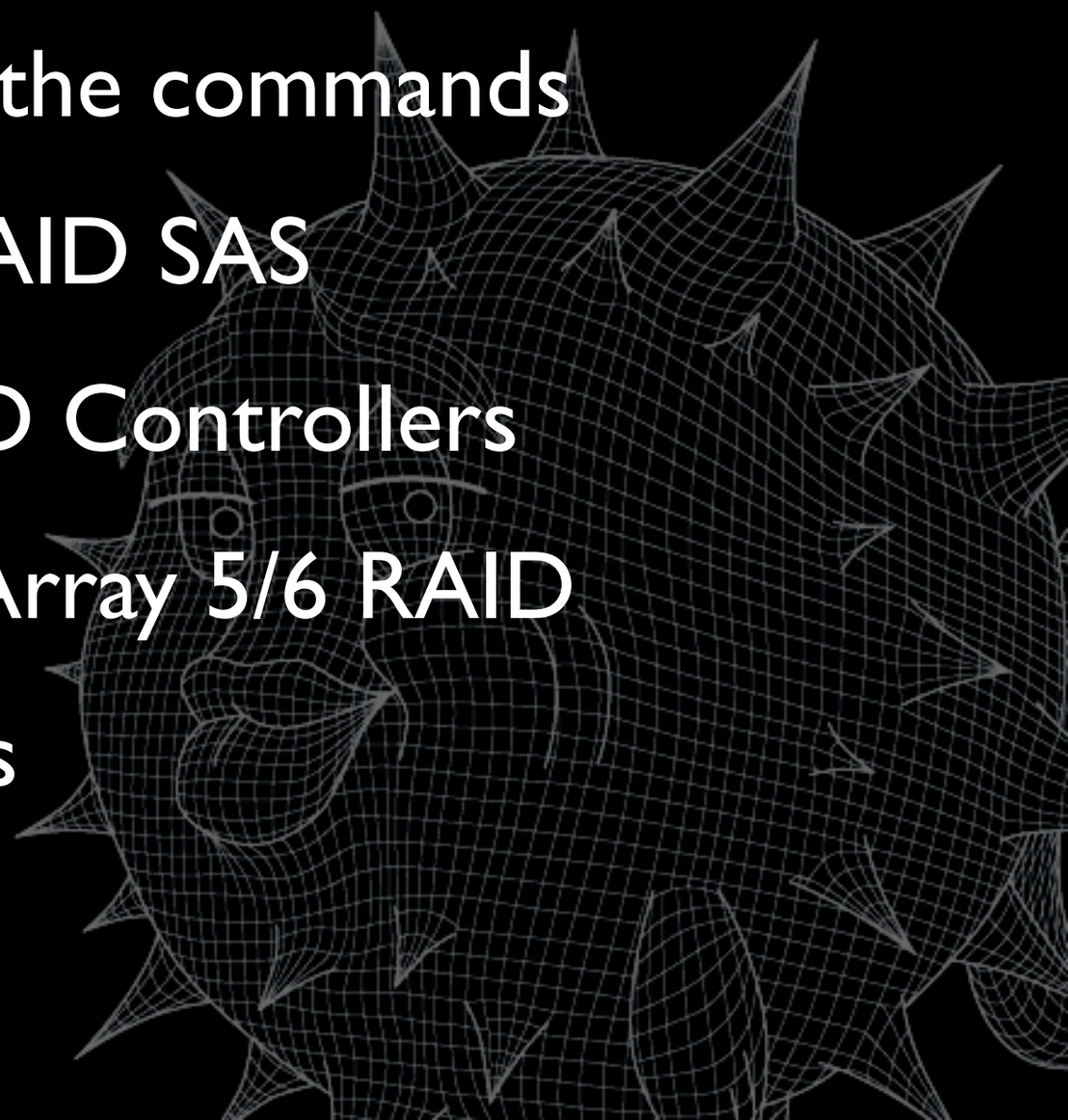


SES and SAF-TE

- short for "SCSI Enclosure Services" and "SCSI Accessed Fault-Tolerant Enclosures"
- they're needed for one main reason
    - SCSI does not support hot-plug without either one of these devices. in the above example the insertion of the disk in slot 1:9 would go undetected without an enclosure
- also provide normal temp/volt/etc sensors

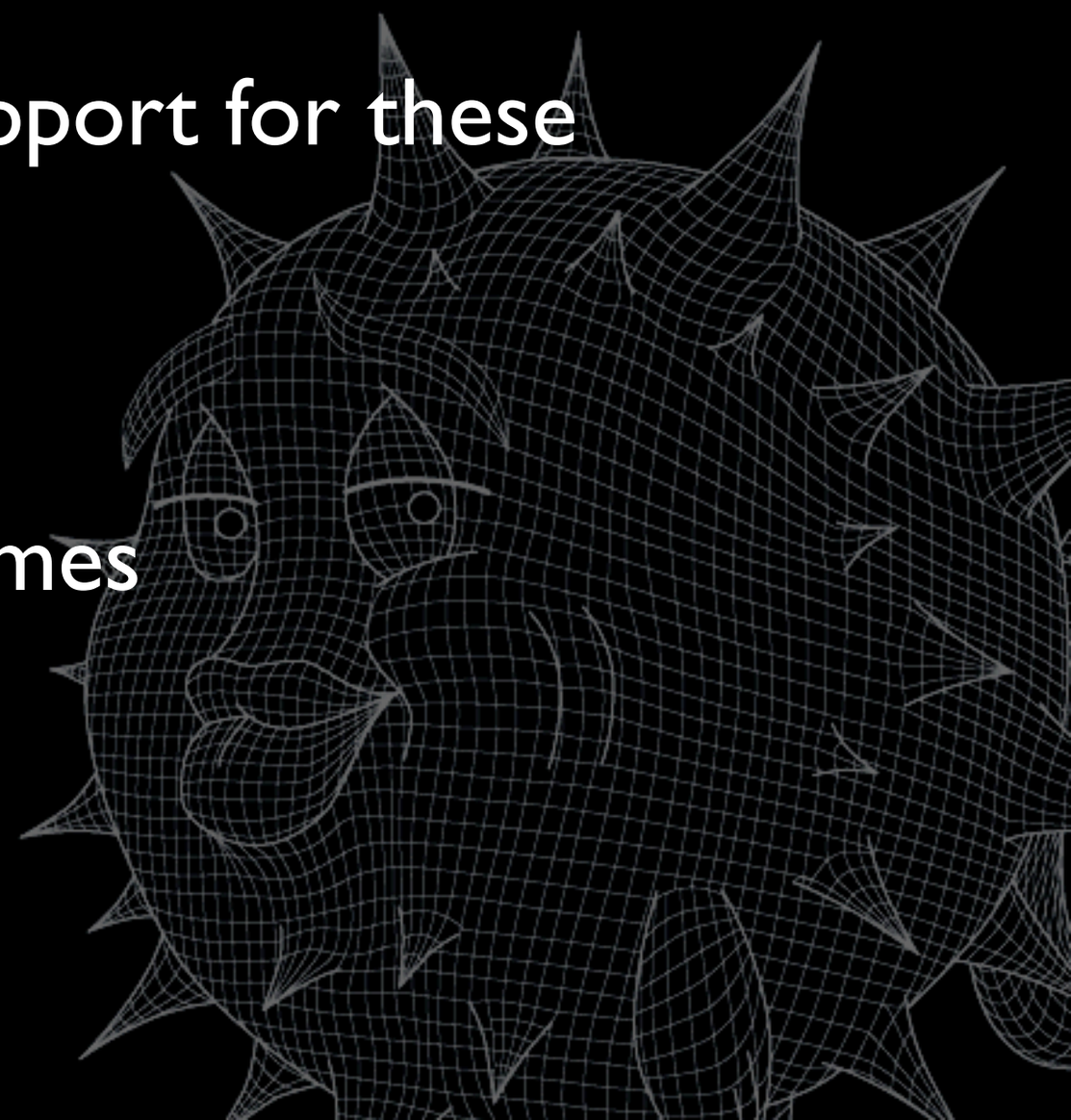
supported hardware

- ami(4) - LSI Logic MegaRAID ATA/SCSI/SATA
  - older cards don't grok the commands
- mfi(4) - LSI Logic MegaRAID SAS
- arc(4) - Areca SATA RAID Controllers
- ciss(4) - Compaq Smart Array 5/6 RAID
- ses(4), safte(4) enclosures



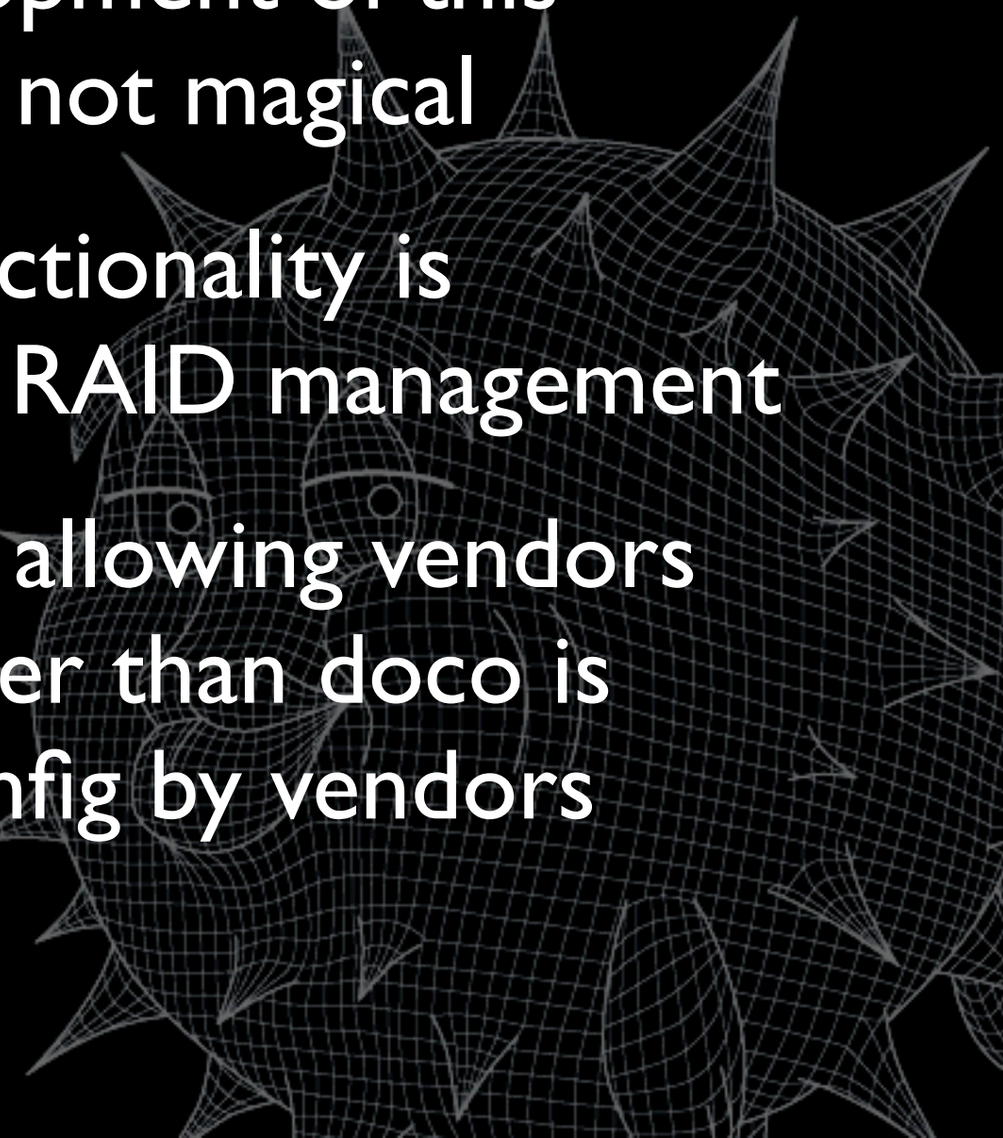
what's new in 4.0

- mfi(4), arc(4), plus bio support for these controllers
- bio on ciss(4)
- rebuild progress for volumes



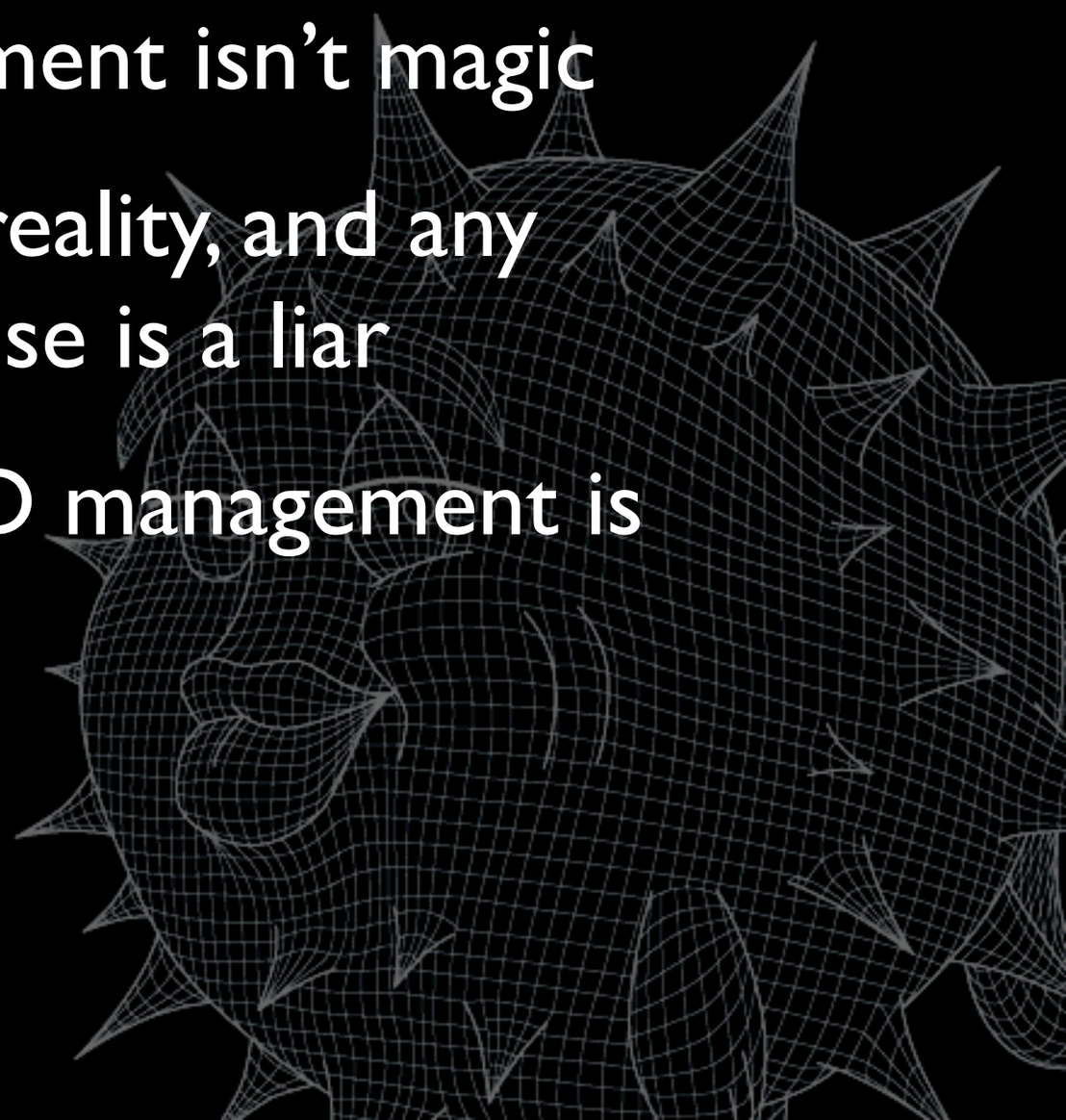
conclusion

- RAID isn't some arcane voodoo (no chickens were harmed in the development of this software), and sensors are not magical
- only a small amount of functionality is necessary to create useful RAID management
- if we can do it, so can you. allowing vendors to provide their tools rather than doco is hurting users. imagine ifconfig by vendors



conclusion again

- RAID and RAID management isn't magic
- it is extremely simple in reality, and any vendor who says otherwise is a liar
- we have shown that RAID management is easier than ifconfig

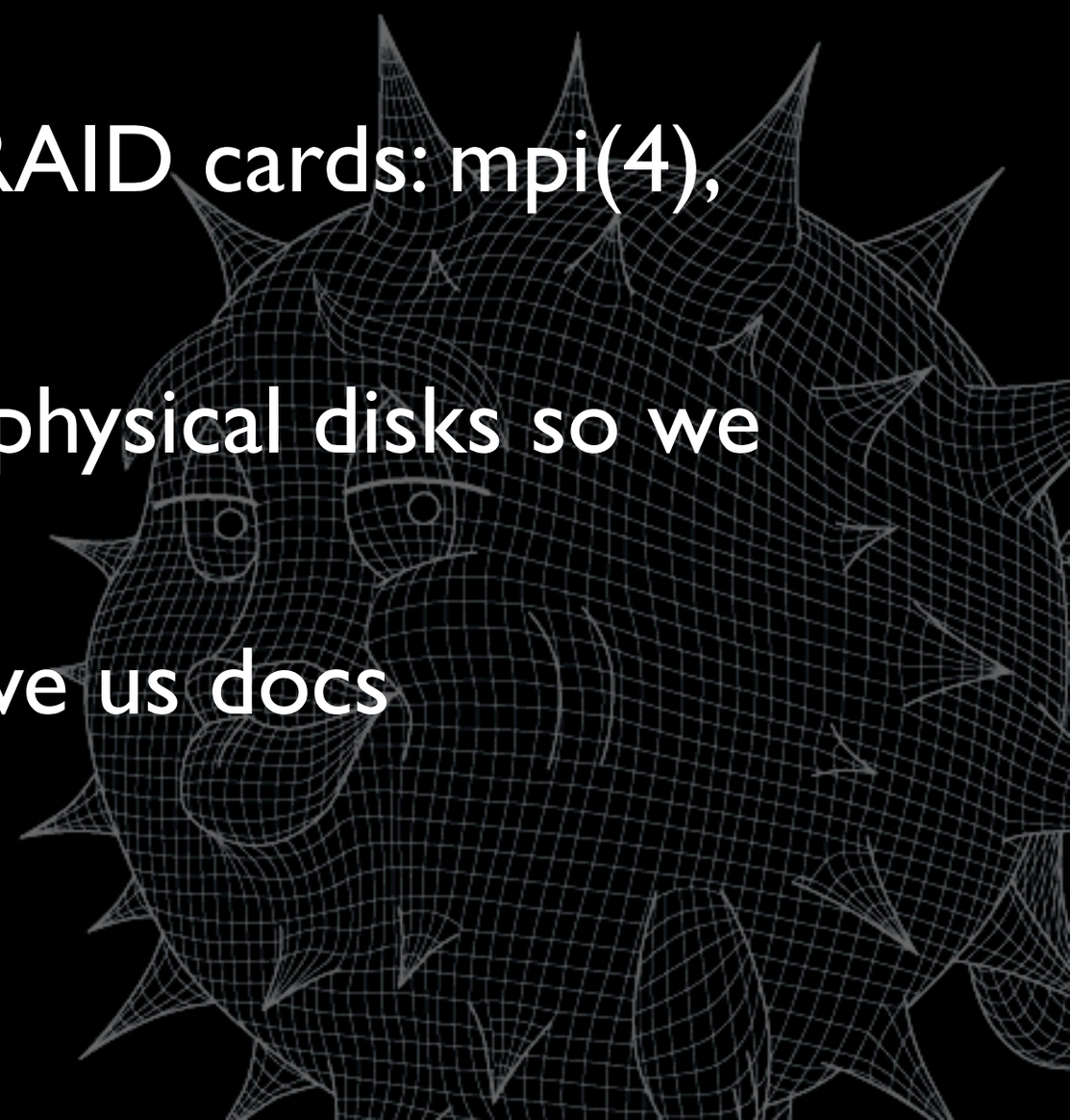


future work

- both sensors and bio have been around for a while now. we intend to go back and rework these a bit based on our experience. still works in progress
- for sensors
  - a new sensorsd with a hetero config file
  - new sensor types and drivers for new hw

i have a dream... (the future)

- for bio
- add support to other RAID cards: mpi(4), gdt(4), ips(4)
- S.M.A.R.T. support for physical disks so we can predict failure
- convince vendors to give us docs



thx

- marco, krw, pascoe, deraadt for putting up with my stupid questions
- marco and deraadt for giving me the freedom to play around with this stuff
- donators for giving me toys to play with
- grange, for being so talented