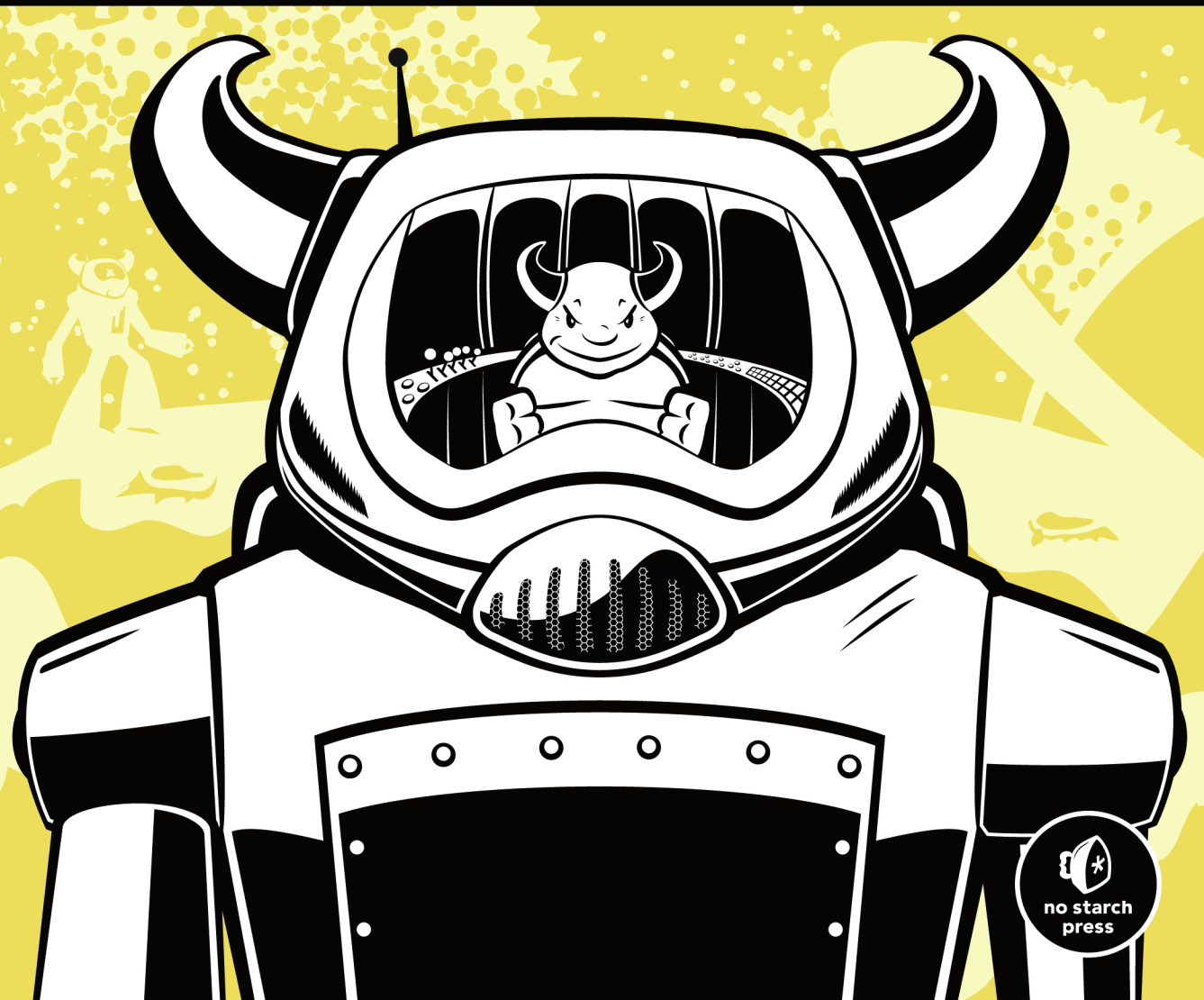


# FREEBSD<sup>®</sup> DEVICE DRIVERS

A GUIDE FOR THE INTREPID

JOSEPH KONG



# INDEX

## Numbers

00–04 bit, 291  
05–07 bit, 291  
08–15 bit, 291  
0 constant, 95  
0xFFFFFFFF, 197  
16–18 bit, 291  
19 bit, 291  
20–27 bit, 291  
28–31 bit, 291

## A

access argument, 46  
acpi\_sleep\_event event handler, 92  
acpi\_wakeup\_event event handler, 92  
action routines  
    XPT\_GET\_TRAN\_SETTINGS constant, 246–249  
    XPT\_PATH\_INQ constant, 243–245  
    XPT\_RESET\_BUS constant, 245–246  
    XPT\_RESET\_DEV constant, 255  
    XPT\_SCSI\_IO constant, 250–255  
    XPT\_SET\_TRAN\_SETTINGS constant, 249  
Advanced Technology Attachment Packet  
    Interface (ATAPI), 226  
ahc\_action function, 227  
ahc\_done function, 227, 240  
alignment argument, 23, 197  
alternate setting, 258  
arg argument, 46, 116  
at45d\_attach function, 217–218  
at45d\_delayed\_attach function, 218–219  
at45d\_get\_info function, 219–220  
at45d\_get\_status function, 220–221  
at45d\_strategy function, 221  
at45d\_task function, 221–223  
ATAPI (Advanced Technology Attachment Packet Interface), 226  
atomic\_add\_int function, 106  
autoconfiguration. *See* Newbus drivers

## B

biodone function, 223  
biofinish function, 223  
bio\_pblkno variable, 222  
bioq\_flush function, 213  
bioq\_insert\_head function, 212  
bioq\_insert\_tail function, 212  
bioq\_remove function, 213  
bio structure, 210  
bits\_per\_char function, 111–112  
block, defined, 207  
block-centric I/O requests, 222  
block devices, 2  
block drivers, 15–16  
block I/O queues, 212–213  
block I/O structures, 210–211  
boundary argument, 23, 197  
<bsd.kmod.mk> Makefile, 6  
bt.c source file, 255  
buffers, DMA, 195  
    bus\_dmamap\_load function, 200–201  
    bus\_dmamap\_load\_mbuf function, 201  
    bus\_dmamap\_load\_mbuf\_sg function, 201–202  
    bus\_dmamap\_load\_uio function, 202  
    bus\_dmamap\_unload function, 202  
    bus\_dma\_segment structures, 199–200  
buflen argument, 200  
bufsize field, 260  
bulk endpoints, 258  
bus\_alloc\_resource function, 123, 166  
bus\_deactivate\_resource function, 124  
BUS\_DMA\_ALLOCNOW constant, 198  
BUS\_DMA\_COHERENT constant, 199, 203  
bus\_dmamap\_create function, 195, 199  
bus\_dmamap\_destroy function, 199  
bus\_dmamap\_load function, 195, 200–201  
bus\_dmamap\_load\_mbuf function, 201  
bus\_dmamap\_load\_mbuf\_sg function, 201–202  
bus\_dmamap\_load\_uio function, 202  
bus\_dmamap\_unload function, 202

- bus\_dmamem\_alloc function, 202, 205
- bus\_dmamem\_free function, 203
- BUS\_DMA\_NOCACHE constant, 172, 201, 203
- BUS\_DMA\_NOWAIT constant, 201, 203
- bus\_dma\_segment structures, 199–200
- BUS\_DMASYNC\_POSTREAD constant, 205
- BUS\_DMASYNC\_PREWRITE constant, 205
- bus\_dma\_tag\_create function, 195
- bus\_dma\_tag\_destroy function, 198
- BUS\_DMA\_WAITOK constant, 203
- BUS\_DMA\_ZERO constant, 203
- busname argument, 116
- BUS\_PROBE\_SPECIFIC success code, 133
- bus\_read\_N functions, 167
- bus\_release\_resource function, 124
- bus\_setup\_intr function, 126
- BUS\_SPACE\_BARRIER\_READ constant, 172
- BUS\_SPACE\_BARRIER\_WRITE constant, 172
- BUS\_SPACE\_MAXADDR constant, 197
- bus\_teardown\_intr function, 126
- bus\_write\_multi\_N functions, 169
- bus\_write\_N functions, 169
- bus\_write\_region\_N functions, 169

## C

- callback2 argument, 201
- callback2 function, 201
- callbackarg argument, 200
- callback argument, 200
- callback field, 260
- callout\_drain function, 96
- callout\_init\_mtx function, 95
- callout\_init function, 94
- CALLOUT\_MPSAFE constant, 95
- callout\_reset function, 96
- CALLOUT\_RETURNUNLOCKED constant, 95
- callouts, 94–98
- callout\_schedule function, 96
- CALLOUT\_SHAREDLOCK constant, 95
- callout\_stop function, 95
- CAM (Common Access Method) standard
  - action routines
    - XPT\_GET\_TRAN\_SETTINGS constant, 246–249
    - XPT\_PATH\_INQ constant, 243–245
    - XPT\_RESET\_BUS constant, 245–246
    - XPT\_RESET\_DEV constant, 255
    - XPT\_SCSI\_IO constant, 250–255
    - XPT\_SET\_TRAN\_SETTINGS constant, 249
  - HBA (host bus adapter) driver example
    - mfip\_action function, 236–238
    - mfip\_attach function, 234–235
    - mfip\_detach function, 235–236
    - mfip\_done function, 240–241

- mfip\_poll function, 238
- mfip\_start function, 238–240
- overview, 225–227
- SIM registration routines
  - cam\_sim\_alloc function, 242–243
  - cam\_simq\_alloc function, 242
  - xpt\_bus\_register function, 243
- CAM Control Block (CCB), 227
  - camisr function, 227
  - cam\_sim\_alloc function, 235, 242–243
  - cam\_simq\_alloc function, 242
- CCB (CAM Control Block), 227
- ccb\_h.func\_code variable, 238
- ccb\_pathinq structure, 243, 245
- ccb\_scsiio structure, 253
- ccb\_trans\_settings structure, 249
- chan argument, 84
- change\_callback argument, 290
- character devices, 2
- character drivers
  - character device switch table, 8
  - destroy\_dev function, 9
  - DEV\_MODULE macro, 15
  - d\_foo function, 7–8
  - echo\_modevent function, 14
  - echo\_read function, 13–14
  - echo\_write function, 12–13
  - loading, 15
  - make\_dev function, 9
- ciss\_setup\_msix function, 294–295
- commands for ioctl interface, 29–30
- Common Access Method (CAM) standard. *See* CAM standard
- compiling KLDs, 6–7
- condition variables, 79–81
- configurations, 259
- configuration structures for USB drivers
  - management routines for, 264–265
  - mandatory fields, 260
  - optional fields, 260–261
  - transfer flags, 261–262
- contexts for sysctl interface, 44
- contigfree function, 22, 23
- contigmalloc function, 22–25
- contiguous physical memory, 22–25
- control endpoints, 258
- cookie argument, 126
- count argument, 123
- count value, 54
- ctx argument, 46
- cv\_broadcastpri function, 81
- cv\_destroy function, 81
- cv\_init function, 81
- cv\_timedwait function, 81
- cv\_timedwait\_sig function, 81

cv\_wait\_sig function, 81  
cv\_wait\_unlock function, 81  
cv\_wmesg function, 81

## D

d (descriptor) argument, 38  
dadone function, 227  
dastart function, 227  
dastrategy function, 226  
data argument, 4  
Data Carrier Detect (DCD), 107  
data transfers for USB drivers, 262–264  
d\_close field, 209  
d\_close function, 57  
d\_drv1 field, 210  
debug.sleep.test sysctl, 90–92  
DECLARE\_MODULE macro  
  data argument, 4  
  name argument, 4  
  order argument, 4–5  
  sub argument, 4  
delaying execution  
  callouts, 94–96  
  event handlers for, 92–94  
  load function, 89–90  
  sleeping, 83–85  
  sleep\_modevent function, 88–89  
  sleep\_thread function, 90–91  
  sysctl\_debug\_sleep\_test function, 91  
  taskqueues  
    global, 97  
    management routines for, 97–98  
    overview, 96–97  
  unload function, 91–92  
  voluntary context switching, 83–85  
descr argument, 46  
descriptive fields for disk structures, 208–209  
descriptor (d argument), 38  
destroy\_dev function, 9, 72  
destroying tags for DMA, 198–199  
devclass argument, 116  
dev\_clone event handler, 92, 103  
DEV\_MODULE macro, 15  
device\_attach function, 114  
device\_detach function, 114, 115  
device\_foo functions, 114–115  
device\_identify function, 114  
device method table, 115  
device\_probe function, 114  
device\_resume function, 114  
devices  
  configuration of, 259  
  defined, 1  
  driver types, 1–2

device\_shutdown function, 114  
device\_suspend function, 114  
d\_flags field, 209  
d\_foo function, 7–8, 72, 121  
d\_fwheads field, 209  
d\_fwsectors field, 209  
d\_ident field, 209  
d\_ioctl field, 209  
d\_ioctl function, 28, 58  
direction field, 260  
Direct Memory Access (DMA). *See* DMA  
DISKFLAG\_CANDELETE constant, 208  
DISKFLAG\_CANFLUSHCACHE constant, 208  
DISKFLAG\_NEEDSGIANT constant, 208  
disk structures  
  descriptive fields, 208–209  
  driver private data, 210  
  management routines for, 210  
  mandatory media properties, 209  
  optional media properties, 209  
  storage device methods, 209  
dismantling transfers using DMA, 196  
DMA (Direct Memory Access)  
  buffers  
    bus\_dmamap\_load\_mbuf\_sg function, 201–202  
    bus\_dmamap\_load\_mbuf function, 201  
    bus\_dmamap\_load\_uio function, 202  
    bus\_dmamap\_load function, 200–201  
    bus\_dmamap\_unload function, 202  
    bus\_dma\_segment structures, 199–200  
    synchronizing, 205  
  example using, 203–205  
  maps, 199, 202–203  
  overview, 193  
  tags for  
    creating, 197–198  
    destroying, 198–199  
  transfers using  
    dismantling, 196  
    initiating, 196  
dmat argument, 198, 200, 205  
d\_maxsize field, 209  
d\_mediasize field, 209  
dontcare\_mask argument, 289  
d\_open field, 209  
d\_open function, 57  
driver argument, 116  
DRIVER\_MODULE macro  
  arg argument, 116  
  busname argument, 116  
  devclass argument, 116  
  driver argument, 116  
  evh argument, 116  
  name argument, 116

driver private data, 210  
ds\_addr field, 200  
d\_sectorsize field, 209  
d\_strategy field, 209  
d\_stripesize field, 209  
dump routines, 209  
dynamic node, 47  
dynamic sysctl, 44–47

## E

%eax value, 54  
ECHO\_CLEAR\_BUFFER command, 33  
echo\_ioctl function, 36  
echo\_modevent function, 14, 36–37  
echo\_read function, 13–14  
ECHO\_SET\_BUFFER\_SIZE command, 33  
echo\_set\_buffer\_size function, 35  
echo\_write function, 12–13, 34–35  
ECP (Extended Capabilities Port)  
    mode, 156  
em(4) driver, 299  
em\_handle\_rx function, 303  
em\_rxeof function, 300–303  
em\_start\_locked function, 304–305  
em\_txeof function, 305–307  
em\_xmit function, 305  
end argument, 123  
end of packet (eop), 301  
endpoint, 258  
endpoint field, 260  
Enhanced Parallel Port (EPP), 156  
ENXIO error code, 134  
eop (end of packet), 301  
ep\_index field, 261  
EPP (Enhanced Parallel Port), 156  
ether\_ifattach function, 287–288  
ether\_ifdetach function, 288  
EVENTHANDLER\_DEREGISTER macro, 93  
EVENTHANDLER\_INVOKE macro, 94  
EVENTHANDLER\_PRI\_ANY constant, 93  
EVENTHANDLER\_REGISTER macro, 93  
event handlers, 92–94, 98  
evh argument, 116  
exclusive holds, 73  
ext\_buffer flag, 262  
Extended Capabilities Port (ECP)  
    mode, 156  
Extended Message Signaled Interrupts  
    (MSI-X), 294  
extended mode, 156

## F

Fibre Channel (FC), 226  
filter argument, 126  
FILTER\_HANDLED constant, 127  
filter routine, 126  
FILTER\_SCHEDULE\_THREAD constant, 127  
FILTER\_STRAY constant, 127  
filtfuncarg argument, 197  
filtfunc argument, 197  
filtfunc function, 197  
FireWire (IEEE 1394), 226  
flags argument, 18–19, 126, 198–201  
flags field, 261  
flash memory driver example  
    at45d\_attach function, 217–218  
    at45d\_delayed\_attach function, 218–219  
    at45d\_get\_info function, 219–220  
    at45d\_get\_status function, 220–221  
    at45d\_strategy function, 221  
    at45d\_task function, 221–223  
foo bytes, 279  
foo\_callback function, 201  
foo lock, 65  
foo\_pci\_attach function, 120–121  
foo\_pci\_detach function, 121–122  
foo\_pci\_probe function, 120  
force\_short\_xfer flag, 262  
format argument, 46  
frames field, 261  
free function, 18

## G

g (group) argument, 29  
global taskqueues, 97

## H

handler argument, 46  
hardware resource management with  
    Newbus drivers, 122–124  
HBA (host bus adapter) driver  
    mfip\_action function, 236–238  
    mfip\_attach function, 234–235  
    mfip\_detach function, 235–236  
    mfip\_done function, 240–241  
    mfip\_poll function, 238  
    mfip\_start function, 238–240  
Hello, world! KLD, 5–6  
highaddr argument, 197  
host bus adapter (HBA) driver. *See* HBA  
    driver

- I
- IEEE 1394 (FireWire), 226
- if\* structures, 291
- ifaddr\_event event handler, 93
- \*ifattach function, 285
- if\_clone\_event event handler, 93
- if\_index field, 261
- if\_init field, 285
- if\_initname function, 287
- if\_input field, 285
- if\_ioctl field, 285
- ifmedia\_add function, 290
- ifmedia\_removeall function, 291
- ifmedia\_set function, 291
- ifmedia structure, 289
- IFM\_GMASK mask, 291
- IFM\_IMASK mask, 291
- IFM\_MMASK mask, 291
- IFM\_NMASK mask, 291
- IFM\_OMASK mask, 291
- IFM\_TMASK mask, 291
- ifnet\_arrival\_event event handler, 93
- ifnet\_departure\_event event handler, 93
- ifnet structure, 283, 286–287
- if\_output field, 285–286
- if\_qflush field, 286
- if\_reassign field, 286
- if\_resolvemulti field, 286
- if\_start field, 286
- if\_transmit field, 286
- if\_watchdog field, 286
- implementing MSI, 294–296
- initiating transfers using DMA, 196
- init routines, 285
- input/output (I/O) operations. *See* I/O operations
- input routine, 285
- Intelligent Platform Management Interface (IPMI) driver. *See* IPMI driver
- Intel PCI Gigabit Ethernet adapter driver, 291–293
- interfaces, 7, 258, 283
- interrupt, defined, 125
- interrupt endpoints, 258
- interrupt handlers
  - examples of
    - pint\_attach function, 133–134
    - pint\_close function, 135–136
    - pint\_detach function, 134
    - pint\_identify function, 132
    - pint\_intr function, 137–138
    - pint\_open function, 134–135
    - pint\_probe function, 132–133
    - pint\_read function, 136–137
    - pint\_write function, 136
  - overview, 125, 126–127
  - on parallel port, 138–139
  - registering, 125–126
- interrupt-request lines (IRQs), 122
- interval field, 261
- INTR\_ENTROPY constant, 126
- INTR\_MPSAFE constant, 126
- INVARIANTS option, 18
- I/O (input/output) operations. *See also* MMIO; PMIO
  - ioctl interface
    - commands for, 29–30
    - echo\_ioctl function, 36
    - echo\_modevent function, 36–37
    - echo\_set\_buffer\_size function, 35
    - echo\_write function, 34–35
    - invoking, 37–40
  - sysctl interface
    - contexts for, 44
    - dynamic sysctl, 44–47
    - overview, 40–44
    - SYSCTL\_CHILDREN macro, 47
    - sysctl\_set\_buffer\_size function, 50–52
    - SYSCTL\_STATIC\_CHILDREN macro, 47
- ioctl commands, 28
- ioctl interface
  - commands for, 29–30
  - echo\_ioctl function, 36
  - echo\_modevent function, 36–37
  - echo\_set\_buffer\_size function, 35
  - echo\_write function, 34–35
  - invoking, 37–40
- \_IO macro, 29
- i-Opener LEDs driver
  - led\_attach function, 178
  - led\_close function, 180
  - led\_detach function, 178–179
  - led\_identify function, 177
  - led\_open function, 179
  - led\_probe function, 177–178
  - led\_read function, 180–181
  - led\_write function, 181–182
- \_IOR macro, 29
- \_IOW macro, 29
- \_IOWR macro, 29
- IPMI (Intelligent Platform Management Interface) driver
  - ipmi2\_pci\_attach function, 189–191
  - ipmi2\_pci\_probe function, 189
  - ipmi\_pci\_attach function, 187–189

- IPMI driver, *continued*
  - `ipmi_pci_match` function, 186
  - `ipmi_pci_probe` function, 185–186
- `ipmi2_pci_attach` function, 189–191
- `ipmi2_pci_probe` function, 189
- `ipmi_attached` variable, 186
- `ipmi_identifiers` array, 186
- `ipmi_pci_attach` function, 187–189
- `ipmi_pci_match` function, 186
- `ipmi_pci_probe` function, 185–186
- IRQs (interrupt-request lines), 122
- isochronous endpoints, 258
- `ithread` argument, 126
- `ithread` routine, 127

## K

- Keyboard Controller Style (KCS)
  - mode, 188
- KLDs (loadable kernel modules)
  - block drivers, 15–16
  - character drivers
    - character device switch table, 8
    - `destroy_dev` function, 9
    - `DEV_MODULE` macro, 15
    - `d_foo` function, 7–8
    - `echo_modevent` function, 14
    - `echo_read` function, 13–14
    - `echo_write` function, 12–13
    - loading, 15
    - `make_dev` function, 9
  - compiling and loading, 6–7
  - `DECLARE_MODULE` macro
    - data argument, 4
    - name argument, 4
    - order argument, 4–5
    - sub argument, 4
  - Hello, world! example, 5–6
  - module event handlers, 2–3
  - `kldunload -f` command, 61

## L

- LED driver
  - `led_attach` function, 178
  - `led_close` function, 180
  - `led_detach` function, 178–179
  - `led_identify` function, 177
  - `led_open` function, 179
  - `led_probe` function, 177–178
  - `led_read` function, 180–181
  - `led_write` function, 181–182
- `len` argument, 46
- loadable kernel modules (KLDs). *See* KLDs

- load function, 89–90
- loading
  - character drivers, 15
  - KLDs, 6–7
- `lockfuncarg` argument, 198
- `lockfunc` argument, 198
- locks, 65
- `longdesc` argument, 19
- `lowaddr` argument, 197
- LP\_BUSY flag, 157
- LP\_BYPASS flag, 156
- `lpt_attach` function, 148–150
- `lpt_close` function, 159–160
- `lptcontrol(8)` utility, 162
- `lpt_detach` function, 150
- `lpt_detect` function, 147–148
- `lpt_identify` function, 146
- `lpt_intr` function, 156–157
- `lpt_ioctl` function, 160–162
- `lpt_open` function, 151–153
- `lpt_port_test` function, 147, 148
- `lpt_probe` function, 146
- `lpt_push_bytes` function, 158–159
- `lpt_read` function, 153–154
- `lpt_release_ppbus` function, 162–163
- `lpt_request_ppbus` function, 162
- `lpt_timeout` function, 158
- `lpt_write` function, 154–156

## M

- `make_dev` function, 9
- Makefiles, 6
- `MALLOC_DECLARE` macro, 20
- `MALLOC_DEFINE` macro, 19
- `malloc` function, 18
- `malloc_type` structures
  - `MALLOC_DECLARE` macro, 20
  - `MALLOC_DEFINE` macro, 19
- management routines
  - for condition variables, 80–81
  - for disk structures, 210
  - for DMA maps, 199, 202–203
  - for MSI (Message Signaled Interrupts), 297
  - for mutex locks, 66–68
  - for network interface media structures, 289–291
  - for network interface structures, 286–288
  - for rw (reader/writer) locks, 78–79
  - for sx (shared/exclusive) locks, 73–75
  - for taskqueues, 97–98

- mandatory fields for USB drivers, 260
- mandatory media properties for disk structures, 209
- manual\_status flag, 262
- maps, DMA, 199, 202–203
- masks, for ignoring bits, 291
- max\_dev\_transactions argument, 242, 243
- MAX\_EVENT constant, 88
- maxsegsz argument, 198
- maxsize argument, 198
- max\_tagged\_dev\_transactions argument, 243
- mbuf argument, 201
- mbuf chain, 293
- mbuf structures, 293–294
- M\_ECHO structure, 22
- media properties for disk structures
  - mandatory, 209
  - optional, 209
- memory allocation
  - contiguous physical memory, 22–25
  - malloc\_type structures
    - MALLOC\_DECLARE macro, 20
    - MALLOC\_DEFINE macro, 19
  - overview, 17–19
- memory barriers, 172
- memory-mapped I/O (MMIO). *See* MMIO
- Message Signaled Interrupts (MSI), 294
  - implementing, 294–296
  - management routines for, 297
- methods structure for USB drivers, 265
- mfi(4) code base, 241
- mfi\_intr function, 240
- mfi\_action function, 236–238
- mfi\_attach function, 234–235
- mfi\_detach function, 235–236
- mfi\_done function, 240–241
- mfi\_poll function, 238
- mfi\_start function, 238–240
- mfi\_startio function, 239, 252
- MMIO (memory-mapped I/O). *See also*
  - I/O operations; PMIO
  - and memory barriers, 172
  - reading from, 166–167
  - stream operations, 169–172
  - writing to, 167–169
- M\_NOWAIT constant, 19, 23
- modem drivers. *See* virtual null modem
- modeventtype\_t argument, 3
- MOD\_QUIESCE constant, 61
- module event handlers, 2–3
- MSI (Message Signaled Interrupts), 294
  - implementing, 294–296
  - management routines for, 297
- MSI message, 294

- MSI-X (Extended Message Signaled Interrupts), 294
- MSI-X message, 294
- msleep\_spin function, 85
- MTX\_DEF constant, 67
- mtx\_destroy function, 68
- MTX\_DUPOK constant, 67
- mtx\_init function, 67
- MTX\_NOPROFILE constant, 67
- MTX\_NOWITNESS constant, 67
- MTX\_QUIET constant, 67
- MTX\_RECURSE constant, 67
- MTX\_SPIN constant, 67
- mtx\_trylock function, 68
- mtx\_unlock\_spin function, 68
- mutex locks
  - management routines for, 66–68
  - race\_modevent function, 71–72
  - sleep mutexes, 66
  - spin mutexes, 65–66
- M\_WAITOK constant, 19, 23
- mword value, 290–291
- M\_ZERO constant, 19, 23

## N

- name argument
  - for DECLARE\_MODULE macro, 4
  - description of, 46
  - for DRIVER\_MODULE macro, 116
- n argument, 30
- network devices, 2
- network drivers
  - example of, 291–293
  - mbuf structures, 293–294
  - MSI (Message Signaled Interrupts), 294
    - implementing, 294–296
    - management routines for, 297
  - network interface media structures, 289–291
  - network interface structures
    - ether\_ifattach function, 287–288
    - ether\_ifdetach function, 288
    - management routines for, 286–288
  - packets.
    - post transmitting, 307–308
    - receiving, 299–303
    - transmitting, 304–307
- network interface media structures, 289–291
- network interface structures
  - ether\_ifattach function, 287–288
  - ether\_ifdetach function, 288
  - management routines for, 286–288



- Newbus drivers
  - device\_foo functions, 114–115
  - device method table, 115
  - DRIVER\_MODULE macro
    - arg argument, 116
    - busname argument, 116
    - devclass argument, 116
    - driver argument, 116
    - evh argument, 116
    - name argument, 116
  - example of
    - d\_foo functions, 121
    - foo\_pci\_attach function, 120–121
    - foo\_pci\_detach function, 121–122
    - foo\_pci\_probe function, 120
    - loading, 122
  - hardware resource management with, 122–124
  - overview, 113–114
- nibble mode, 154
- nmdm(4) driver, 99–100, 102
- nmdm\_alloc function, 105–106
- nmdm\_clone function, 104
- nmdm\_count variable, 103
- nmdm\_inwakeup function, 108
- nmdm\_modem function, 108–109
- nmdm\_modevent function, 103
- nmdm\_outwakeup function, 106
- nmdm\_param function, 109–110
- nmdm\_task\_tty function, 106–107
- nmdm\_timeout function, 110, 111
- no\_pipe\_ok flag, 262
- np\_rate variable, 110
- nsegments argument, 198
- ns\_part variables, 106
- number argument, 46

## O

- optional fields for USB drivers, 260–261
- optional media properties for disk structures, 209
- order argument, 4–5
- output routines, 285–286

## P

- packets
  - post transmitting, 307–308
  - receiving
    - em\_handle\_rx function, 303
    - em\_rxeof function, 300–303
  - transmitting
    - em\_start\_locked function, 304–305
    - em\_txeof function, 305–307

- parallel port
  - interrupt handlers on, 138–139
  - printer driver example
    - lpt\_attach function, 148–150
    - lpt\_close function, 159–160
    - lpt\_detach function, 150
    - lpt\_detect function, 147–148
    - lpt\_identify function, 146
    - lpt\_intr function, 156–157
    - lpt\_ioctl function, 160–162
    - lpt\_open function, 151–153
    - lpt\_port\_test function, 148
    - lpt\_probe function, 146
    - lpt\_push\_bytes function, 158–159
    - lpt\_read function, 153–154
    - lpt\_release\_ppbus function, 162–163
    - lpt\_request\_ppbus function, 162
    - lpt\_timeout function, 158
    - lpt\_write function, 154–156
- parent argument, 46, 197
- pause function, 85
- pci\_alloc\_msi function, 297
- pci\_alloc\_msix function, 297
- pci\_msi\_count function, 297
- pci\_msix\_count function, 297
- PCIR\_BAR(x) macro, 189
- pci\_release\_msi function, 297
- \_pcsid structures, 120
- physical memory, contiguous, 22–25
- pint\_attach function, 133–134
- pint\_close function, 135–136
- pint\_detach function, 134
- pint\_identify function, 132
- pint\_intr function, 137–138
- pint\_open function, 134–135
- pint\_probe function, 132–133
- pint\_read function, 136–137
- pint\_write function, 136
- pipe, defined, 257
- pipe\_bof flag, 262
- PMIO (port-mapped I/O). *See also* I/O operations; MMIO
  - i-Opener LEDs driver example
    - led\_attach function, 178
    - led\_close function, 180
    - led\_detach function, 178–179
    - led\_identify function, 177
    - led\_open function, 179
    - led\_probe function, 177–178
    - led\_read function, 180–181
    - led\_write function, 181–182
  - and memory barriers, 172
  - reading from, 166–167
  - stream operations, 169–172
  - writing to, 167–169

- poll routines, 238
- port-mapped I/O (PMIO). *See* PMIO
- power\_profile\_change event handler, 93
- ppb\_release\_bus function, 136
- ppb\_sleep function, 137
- printer driver
  - ulpt\_close function, 276
  - ulpt\_detach function, 273
  - ulpt\_ioctl function, 276
  - ulpt\_open function, 273–274
  - ulpt\_probe function, 270–273
  - ulpt\_read\_callback function, 280–281
  - ulpt\_reset function, 274
  - ulpt\_start\_read function, 277
  - ulpt\_start\_write function, 278
  - ulpt\_status\_callback function, 281–282
  - ulpt\_stop\_read function, 278
  - ulpt\_stop\_write function, 278
  - ulpt\_watchdog function, 277
  - ulpt\_write\_callback function, 279–280
  - unlpt\_open function, 275–276
- priority argument, 84
- process\_exec event handler, 93
- process\_exit event handler, 93
- process\_fork event handler, 93
- proxy\_buffer flag, 262
- pseudocode, 194–195
- pseudo-devices, 2

## Q

- qflush routines, 286

## R

- race conditions, 65
- race\_destroy function, 59
- race\_find function, 58–59
- RACE\_IOC\_ATTACH operation, 60
- RACE\_IOC\_DETACH operation, 60
- RACE\_IOC\_LIST operation, 60
- RACE\_IOC\_QUERY operation, 60
- race\_ioctl function, 59–61, 70
- race\_ioctl.h header, 57
- race\_ioctl\_mtx function, 70
- race\_modevent function, 60–61, 71–72
- race\_new function, 58
- race\_softc structure, 57, 58, 59, 64
- r argument, 126
- readers, defined, 78
- reader/writer (rw) locks, 78–79
- reading
  - from MMIO (memory-mapped I/O), 166–167
  - from PMIO (port-mapped I/O), 166–167

- read operations, 29
- reallocf function, 18
- realloc function, 18
- reassign routines, 286
- receiving packets
  - em\_handle\_rx function, 303
  - em\_rxeof function, 300–303
  - overview, 299–300
- recurring on exclusive locks, avoiding, 81
- registering interrupt handlers, 125–126
- resolvemulti routines, 286
- RF\_ACTIVE constant, 123
- RF\_ALLOCATED constant, 123
- RF\_SHAREABLE constant, 123
- RFSTOPPED constant, 90
- RF\_TIMESHARE constant, 123
- rid argument, 123
- rw (reader/writer) locks, 78–79
- rw\_destroy function, 79
- rw\_init\_flags function, 79
- rw\_init function, 79
- rw\_runlock function, 79
- rw\_try\_rlock function, 79
- rw\_try\_wlock function, 79
- rw\_wunlock function, 79

## S

- scatter/gather segment, 198
- sc\_open\_mask value, 179
- sc\_open\_mask variable, 178
- sc\_read\_mask variable, 178
- sc->sc\_state value, 135–136, 152
- SCSI Parallel Interface (SPI), 226
- Server Management Interface Chip (SMIC) mode, 188
- shared/exclusive (sx) locks. *See* sx locks
- shared holds, 73
- shortdesc argument, 19
- short\_frames\_ok flag, 262
- short\_xfer\_ok flag, 262
- shutdown\_final event handler, 92, 93
- shutdown\_post\_sync event handler, 93
- shutdown\_pre\_sync event handler, 93
- sigoff argument, 109
- sigon argument, 109
- SIM queues, 235
- SIM registration routines for CAM (Common Access Method)
  - cam\_sim\_alloc function, 242–243
  - cam\_simq\_alloc function, 242
  - xpt\_bus\_register function, 243
- SIMs (software interface modules), 225
- size argument, 18
- \*sleep function, 66

- sleeping, 66, 83–85, 98
- sleep\_modevent function, 88–89
- sleep mutexes, 66
- sleep\_thread function, 90–91
- SMBIOS (System Management BIOS), 188
- SMIC (Server Management Interface Chip) mode, 188
- software interface modules (SIMs), 225
- SPI (SCSI Parallel Interface), 226
- spi\_command structure, 220, 223
- spin, defined, 65
- spin mutexes, 65–66
- stall\_pipe flag, 262
- start argument, 123
- start routines, 286
- static node, 47
- status\_callback argument, 290
- storage device methods for disk structures, 209
- storage drivers
  - block I/O queues, 212–213
  - block I/O structures, 210–211
  - disk structures
    - descriptive fields, 208–209
    - driver private data, 210
    - management routines for, 210
    - mandatory media properties, 209
    - optional media properties, 209
    - storage device methods, 209
  - flash memory driver example
    - at45d\_attach function, 217–218
    - at45d\_delayed\_attach function, 218–219
    - at45d\_get\_info function, 219–220
    - at45d\_get\_status function, 220–221
    - at45d\_strategy function, 221
    - at45d\_task function, 221–223
- strategy routines, 209
- stream operations, 169–172
- struct usb\_xfer \* argument, 262–264
- sub argument, 4
- sx (shared/exclusive) locks
  - avoid holding exclusive locks for long periods of time, 82
  - avoid recursing on exclusive locks, 81
  - example of, 75–78
  - management routines for, 73–75
- sx\_destroy function, 75
- SX\_DUPOK constant, 74
- sx\_init\_flags function, 74
- sx\_init function, 74
- SX\_NOADAPTIVE constant, 74
- SX\_NOPROFILE constant, 74
- SX\_NOWITNESS constant, 74
- SX\_QUIET constant, 74
- SX\_RECURSE constant, 74
- sx\_slock\_sig function, 74
- sx\_unlock function, 75
- sx\_xlock\_sig function, 74
- sx\_xunlock function, 74
- synchronization primitives, 65
- synchronizing DMA buffers, 205
- SYSCTL\_ADD\_\* macros, 44, 46–47
- SYSCTL\_ADD\_INT macro, 43
- SYSCTL\_ADD\_LONG macro, 42
- SYSCTL\_ADD\_NODE macro, 42, 43, 47
- SYSCTL\_ADD\_OID macro, 46
- SYSCTL\_ADD\_PROC macro, 43
- SYSCTL\_ADD\_STRING macro, 43
- SYSCTL\_CHILDREN macro, 47
- sysctl contexts, 42
- sysctl\_ctx\_init function, 44
- sysctl\_debug\_sleep\_test function, 90, 91
- SYSCTL\_HANDLER\_ARGS constant, 51
- sysctl interface
  - contexts for, 44
  - dynamic sysctl, 44–47
  - overview, 40–44
  - SYSCTL\_CHILDREN macro, 47
  - sysctl\_set\_buffer\_size function, 50–52
  - SYSCTL\_STATIC\_CHILDREN macro, 47
- sysctl\_set\_buffer\_size function, 50–52
- SYSCTL\_STATIC\_CHILDREN macro, 47
- sysinit\_elem\_order enumeration, 4
- <sys/malloc.h> header, 20
- <sys/module.h> header, 4
- SYS\_RES\_IOPORT constant, 123
- SYS\_RES\_IRQ constant, 123
- SYS\_RES\_MEMORY constant, 123
- System Management BIOS (SMBIOS), 188

## T

- tags for DMA
  - creating, 197–198
  - destroying, 198–199
- t argument, 30
- TASK\_INIT macro, 98
- taskqueue\_drain function, 98
- taskqueue\_enqueue function, 98
- taskqueue\_run function, 98
- taskqueues
  - global, 97
  - management routines for, 97–98
  - overview, 96–97

- tasks, 96
- TF\_NOPREFIX flag, 105
- threads, context switches by, 84
- thread synchronization
  - example of
    - problem in, 61–65
    - race\_destroy function, 59
    - race\_find function, 58–59
    - race\_ioctl function, 59–60
    - race\_modevent function, 60–61
    - race\_new function, 58
  - locks, 65
  - mutex locks
    - management routines for, 66–68
    - race\_modevent function, 71–72
    - sleep mutexes, 66
    - spin mutexes, 65–66
  - reasons for, 54
  - rw (reader/writer) locks, 78–79
  - sx (shared/exclusive) locks
    - avoid holding exclusive locks for long periods of time, 82
    - avoid recursing on exclusive locks, 81
    - example of, 75–78
    - management routines for, 73–75
- timeout field, 260
- timo argument, 85
- transfer flags for USB drivers, 261–262
- transfers using DMA, 196
- transmit routines, 286
- transmitting packets
  - em\_start\_locked function, 304–305
  - em\_txeof function, 305–307
  - post transmitting, 307–308
- tsleep function, 84
- tty\_alloc\_mutex function, 100
- TTY device, 100
- tty\_makedev function, 100
- tty\_softc function, 100
- tx\_buffer variable, 306
- tx\_desc variable, 306
- type field, 260

**U**

- UE\_BULK endpoint type, 261
- UE\_CONTROL endpoint type, 261
- UE\_DIR\_ANY constant, 260
- UE\_DIR\_IN constant, 260
- UE\_DIR\_OUT constant, 260
- UE\_INTERRUPT endpoint type, 261
- UE\_ISOCHRONOUS endpoint type, 261
- ulpt\_close function, 276
- ulpt\_detach function, 273
- ulpt\_ioctl function, 276
- ulpt\_open function, 273–274
- ulpt\_probe function, 270–273
- ulpt\_read\_callback function, 280–281
- ulpt\_reset function, 274
- ulpt\_start\_read function, 277
- ulpt\_start\_write function, 278
- ulpt\_status\_callback function, 281–282
- ulpt\_stop\_read function, 278
- ulpt\_stop\_write function, 278
- ulpt\_watchdog function, 277
- ulpt\_write\_callback function, 279–280
- UMASS (USB Mass Storage), 226
- Universal Serial Bus (USB) drivers. *See* USB drivers
- unload function, 89, 91–92
- unlpt\_open function, 275–276
- USB (Universal Serial Bus) drivers
  - configuration structures
    - management routines for, 264–265
    - mandatory fields, 260
    - optional fields, 260–261
    - transfer flags, 261–262
  - data transfers, 262–264
  - methods structure, 265
  - overview, 257–259
  - printer driver example
    - ulpt\_close function, 276
    - ulpt\_detach function, 273
    - ulpt\_ioctl function, 276
    - ulpt\_open function, 273–274
    - ulpt\_probe function, 270–273
    - ulpt\_read\_callback function, 280–281
    - ulpt\_reset function, 274
    - ulpt\_start\_read function, 277
    - ulpt\_start\_write function, 278
    - ulpt\_status\_callback function, 281–282
    - ulpt\_stop\_read function, 278
    - ulpt\_stop\_write function, 278
    - ulpt\_watchdog function, 277
    - ulpt\_write\_callback function, 279–280
    - unlpt\_open function, 275–276
- usb\_config structures, 259
- usbd\_transfer\_drain function, 265
- usbd\_transfer\_setup function, 264
- usbd\_transfer\_start function, 264
- usbd\_transfer\_stop function, 265
- usb\_fifo\_attach function, 265
- usb\_fifo\_detach function, 265
- usb\_fifo\_methods structure, 265
- USB frames, 261
- USB Mass Storage (UMASS), 226
- USB packets, 261
- USB\_ST\_SETUP constant, 264

## V

- variable declarations, 88
- virtual null modem
  - bits\_per\_char function, 111–112
  - loading, 112
  - nmdm\_alloc function, 105–106
  - nmdm\_clone function, 104
  - nmdm\_inwakeup function, 108
  - nmdm\_modem function, 108–109
  - nmdm\_modevent function, 103
  - nmdm\_outwakeup function, 106
  - nmdm\_param function, 109–110
  - nmdm\_task\_tty function, 106–107
  - nmdm\_timeout function, 111
  - overview, 99–100
- vm\_lowmem event handler, 93
- voluntary context switching, 83–85

## W

- Wake-on-LAN (WOL), 293
- wakeup function, 85
- watchdog\_list event handler, 93
- WOL (Wake-on-LAN), 293
- wmesg argument, 85
- write operations, 29
- writing
  - to MMIO (memory-mapped I/O), 167–169
  - to PMIO (port-mapped I/O), 167–169

## X

- xpt\_action function, 227
- xpt\_bus\_register function, 243
- xpt\_done function, 227
- XPT\_GET\_TRAN\_SETTINGS constant, 246–249
- XPT\_GET\_TRAN\_SETTINGS operation, 248
- XPT\_PATH\_INQ constant, 243–245
- XPT\_PATH\_INQ operation, 244
- XPT\_RESET\_BUS constant, 245–246
- XPT\_RESET\_DEV constant, 255
- xpt\_run\_dev\_allocq function, 227
- xpt\_schedule function, 226, 227
- XPT\_SCSI\_IO constant, 250–255
- XPT\_SET\_TRAN\_SETTINGS constant, 249