Index

Symbols

/dev/zero, mapping, 98

A

address space of a process, 92, 103
advisory locking, 56
asynchronous I/O
    behavior, 111
    endpoint service, 140
    guaranteeing buffer state, 111
    listen for network connection, 142
    making connection request, 142
    notification of data arrival, 140
    opening a file, 143
    using aio_result_t structure, 111
    waiting for completion, 129
atomic updates to semaphores, 76

B

blocking mode
    defined, 120
    finite time quantum, 116
    opening a FIFO, 135
    priority inversion, 120
    time-sharing process, 109
    using the read() function, 135
bounded, 112
brk(2), 106

C

chmod(1), 62
class
    definition, 115
    priority queue, 118
    scheduling algorithm, 117
    scheduling priorities, 115
connectionless-mode
    asynchronous network service, 140
    definition, 139
connection-mode
    asynchronous network service, 141
    asynchronously connecting, 142
    definition, 139
    using asynchronous connection, 142
context switch
    preempting a process, 119
control
    semaphore set, 77
`creat()`, 56
creation flags, IPC, 67 to ??

D

dispatch
    priorities, 116
dispatch latency, 112
    under realtime, 112
dispatch table
    configuring, 124
    kernel, 119

F

F_GETLK, 59
fcntl(2), 56, 57
FIFO
    using as byte stream, 135
file and record locking, 55 to ??
file descriptor
    passing to another process, 143
    transferring, 143
file system
    contiguous, 112
    opening dynamically, 143
    using pipes, 135
files
    lock, 55 to ??
    memory-mapped, *See* mapped files
fork(2), ?? to 10
fsync(2), 93
functions
    advanced I/O, 53
    basic I/O, 52
    IPC, 65 to ??
    list file system control, 54
    signals, 17 to ??
    terminal I/O, 63
    user processes, 10

G

GETALL, 81
GETNCNT, 81
GETPID, 80
GETVAL, 80
GETZCNT, 81

I

I/O, *See* asynchronous I/O, or synchronous I/O
init(1M), scheduler properties, 47
Interprocess Communication (IPC)
    administering, 138
    creating pipes, 134
    memory mapped files, 137
    using fileless memory mapping, 137
    using memory mapping, 137
    using messages, 135
    using named pipes, 135
    using pipes, 133
    using semaphores, 136
    using shared memory, 136
    using the open() call, 135
IPC (interprocess communication), 65 to ??
    creation flags, 67 to ??
    functions, 67 to ??
    message header, 71
    message queue, 69
    messages, 68 to ??
    permissions, 66 to 67
    semaphore set, 77
    semaphores, 75 to 83
    shared memory, 83 to ??
IPC_NOWAIT, 83
IPC_RMID, 72, 81, 87
IPC_SET, 72, 81, 87
IPC_STAT, 72, 81, 87

K

kernel
    class independent, 117
    context switch, 119
    dispatch table, 119
    preempting current process, 119
    queue, 111

L

lockf(3C), 57, 60, 178
locking
    advisory, 56, 62
    F_GETLK, 59
    finding locks, 59
    mandatory, 56, ?? to 62
    memory in realtime, 127
    opening a file for, 58
    read, 56, 58, 178
    record, 58
    removing, 58 to ??
    setting, 58 to ??
    supported file systems, 55
    testing locks, 59
    with fcntl(2), 56 to 57
    with lockf(3C), 57
    write, 56, 58, 178
ls(1), 62
lseek(2), 57

M

mandatory locking, 56
mapped files, 94 to 100
    private, 95
    shared, 95
memory
    locking, 127
    locking a page, 127
    locking all pages, 128
    number of locked pages, 127
    sticky locks, 128
    unlocking a page, 128
memory management, 91 to 106
    address spaces, 92
    address-space layout, 103
    coherence, 93
    concepts, 91
    functions, 94
    heterogeneity, 93
    mapping, 91
    mlock(3C), 101 to ??
    mlockall(3C), 102 to ??
    mmap(2), 94 to 100
    mprotect(2), 103
    msync(3C), 102 to 103
    munmap(2), 100
    networking, 93
    pagesize, 103
    virtual memory, 91
memory-mapped files, *See* mapped files
message queue, 69
message, header, 71
messages, 65, 68 to ??
mlock(3C), 101 to ??
mlockall(3C), 102 to ??
mmap(2), 94 to 100
mprotect(2), 103
msgget(), 69
msqid, 71
msync(3C), 102 to 103
munmap(2), 100

N

named pipe
    defined, 135
    FIFO, 133
    using, 135
named pipes, limitations, 65
network
    asynchronous connection, 138
    asynchronous service, 140
    asynchronous transfers, 140
    asynchronous use, 140
    connectionless-mode service, 139
    connection-mode service, 139
    programming models for
        realtime, 139
    services under realtime, 138
    synchronous use, 139
    using STREAMS
        asynchronously, 138
    using Transport-Level Interface
        (TLI), 138
nice(1), 47
nice(2), 47

non-blocking mode
configuring endpoint connections, 142
defined, 138
endpoint bound to service address, 142
network service, 140
polling for notification, 140
service requests, 140
Transport-Level Interface (TLI), 138
using the t_connect() function, 142

O

`open()`, 56

P

page 0, 106
performance, scheduler effect on, 47
permissions
IPC, 66 to 67
pipe
defined, 135
non-blocking read, 135
pipes, limitations, 65
polling
for a connection request, 142
notification of data, 140
using the poll(2) function, 140
priocntl(1), 30 to ??
priocntl(2), ?? to 44
priocntlset(2), ?? to 46
priority inversion
defined, 109
synchronization, 120
priority queue
linear linked list, 119
process
defined for realtime, 107
dispatching, 119
highest priority, 108
preemption, 119
residence in memory, 127
runaway, 111
scheduling for realtime, 116
setting priorities, 123
process address space, 92, 103
process priority
global, 25
real-time, 28
setting and retrieving, 30 to 46
system, 28
time-sharing, 28
process, spawning, ?? to 10
processes, cooperating, locking, 56

R

read
blocking, 135
read lock, 56, 58, 178
`read()`, 56
real-time, scheduler class, 27
removing record locks, 58 to ??
response time
blocking processes, 110
bounds to I/O, 109
degrading, 108
inheriting priority, 109
servicing interrupts, 109
sharing libraries, 109
sticky locks, 110
reversing operations for semaphores, 77

S

sbrk(2), 106
scheduler, 13 to 14, 23 to 50
classes, 117
configuring, 124
effect on performance, 47
priority, 115
realtime, 112
real-time policy, 27
scheduling classes, 115
system policy, 27
time-sharing policy, 26

using system calls, 121
using utilities, 122
scheduler, class, 27
SEM_UNDO, 83
semaphores, 75 to 83
advantages, 65
arbitrary simultaneous updates, 76
atomic updates, 76
reversing operations and SEM_UNDO, 77
set structure, 77
undo structure, 76
semget(), 75
semop(), 76
SETALL, 81
setting record locks, 58 to ??
SETVAL, 80
shared memory, 65, 83 to ??
SHM_LOCK, 87
SHM_UNLOCK, 87
shmget(), 84
signals
code blocking, 20
handlers, 17, 20
limitations, 65
process control, 17
resource limits, 17
sending, 18 to 19
stacks, 21 to ??
structure, semaphore set, 77
synchronization, 93, 113
shared memory, 138
synchronous I/O
blocking, 129
critical timing, 109

T

time slice, real-time process, 40
timers
for interval timing, 145
for realtime applications, 145
timestamping, 145
using one-shot, 146
using periodic type, 146
time-sharing
scheduler class, 26
scheduler parameter table, 27
Transport-Level Interface (TLI)
asynchronous endpoint, 140
connectionless-mode, 138
connection-mode, 138

U

undo structure for semaphores, 76
updates, atomic for semaphores, 76
user priority, 29

V

virtual memory, 91 to 106
See also memory management

W

write lock, 56, 58, 178
`write()`, 56

Z

zero(7), 98