Index

Abstract types, 30
accept(), 65, 66
Acceptor-Connector pattern, 15
Acceptors, nonblocking, 145
ACE (ADAPTIVE Communication Environment), 8, 12–17
  building, 19
  C++ wrapper facade layer, 14–15
  displaying classes, 50
  downloading, 12
  evolution of, 259–267
  framework layer, 15–16
  future for, 267–268
  layered architecture of, 12, 13
  network services, 16–17
  OS adaptation layer, 13–14
  web site for, 14
ACE::read_n(), 76
ACE::select(), 141, 151–152
ACE::write_n(), 76
ACE_Addr, 49–52
  hash(), 51
  operator!=(), 51
  operator==(), 51
ACE_Atomic_Op, 221
ACE_Condition_Thread_Mutex, 208, 229–230
  broadcast(), 229
  signal(), 229
  wait(), 229
ACE_const_cast, 176
ACE_Data_Block, 72
ACE_DEBUG, 93
ACE_dynamic_cast, 176
ACE_ERROR, 93
ACE_ERROR_RETURN, 92, 93
ACE_FILE_Connector, 85
ACE_FILE_IO, 85
ACE_Guard, 208–212, 216–217
ACE_Handle_Set, 140–147
  clr_bit(), 143
  fdset(), 143
  is_set(), 143
  max_set(), 143
  num_set(), 143
  reset(), 143
  set_bit(), 143
  sync(), 143
ACE_Handle_Set_Iterator, 140, 147–151
  operator(), 148
ACE_Hash_Map_Manager, 154, 155
ACE_INET_Addr, 49–52
  addr_to_string(), 52
  get_host_name(), 52
  get_port_number(), 52
  string_to_addr(), 52
ACE_InputCDR, 76–80

good_bit(), 78
operator>>(), 78
steal_contents(), 78
ACE_IPC_SAP, 52–53
disable(), 53, 62
enable(), 53, 62
get_handle(), 53
set_handle(), 53
ACE_Log_Msg::log(), 93
ACE_Mem_Map, 66
ACE_Message_Block, 72–76
clone(), 74
clr_flags(), 74
copy(), 74
duplicate(), 74
length(), 74
msg_priority(), 74
msg_type(), 74
next(), 74
prev(), 74
rd_ptr(), 74
release(), 74
set_flags(), 74
size(), 74
total_length(), 74
wr_ptr(), 74
ACE_Message_Queue, 72
ACE_Null_Condition, 208, 230–231
ACE_Null_Mutex, 208, 212–217
ACE_Null_Semaphore, 208, 222–229
ACE_Object_Manager, 217–218
ACE_OutputCDR, 76–80
begin(), 78
end(), 78
good_bit(), 78
operator<<(), 78
total_length(), 78
ACE_Process, 160–165
child(), 163
exit_code(), 162
getpid(), 162
kill(), 163
parent(), 163
prepare(), 162
spawn(), 162
terminate(), 162
unmanage(), 162
wait(), 162
ACE_Process_Manager, 160, 182
close(), 170
instance(), 170
open(), 170
spawn(), 170
spawn_n(), 170
wait(), 170
ACE_Process_Manager, 169
ACE_Process_Mutex, 208, 212–217
ACE_Process_Options, 160, 165–169
avoid_zombies(), 168
creation_flags(), 168
set_process_attributes(), 168
command_line(), 167
pass_handle(), 167
set_handles(), 167
setenv(), 167
seteuid(), 168
setruid(), 168
working_directory(), 167
ACE_Process_Semaphore, 208, 222–229
ACE_Read_Guard, 208–212, 219–221
ACE_Recursive_Thread_Mutex, 231–233
ACE_reinterpret_cast, 176
ACE_RW_Process_Mutex, 208, 219–221
ACE_RW_Thread_Mutex, 208, 219–221
ACE_Sched_Params, 186, 198–201
next_priority(), 199
prev_priority(), 199
priority_max(), 199

priority_min(), 199
ACE::select(), 140
ACE_SOCK, 54–55
close(), 55
get_local_addr(), 55
get_option(), 55
get_remote_addr(), 55, 85, 88
open(), 55
set_option(), 55
ACE_SOCK_Acceptor, 64–67
accept(), 66
open(), 66
ACE_SOCK_Connector, 56–60
complete(), 58
connect(), 58
ACE_SOCK_IO, 60–64
ACE_SOCK_Stream, 60–64
recv(), 62
recv_n(), 62
recvv_n(), 62
send(), 62
send_n(), 62
sendv_n(), 62
ACE_static_cast, 176
ACE_Task, 205
ACE_Thread_Manager, 186–198
cancel_all(), 189
close(), 189
exit(), 189
instance(), 189
join(), 189
spawn(), 189
spawn_n(), 189
testcancel(), 189
wait(), 189
ACE_Thread_Mutex, 208, 212–217
ACE_Thread_Semaphore, 208, 222–229
ACE_Time_Value, 58, 62
ACE_TSS, 186, 187, 201–205
cleanup(), 203
operator->(), 203
ACE_Write_Guard, 208–212, 219–221
Active Object, 205
Active Object pattern, 16, 133
addr_to_string(), 52
Address family, 37
Arrays of primitive types, 77
Asynchronous I/O, 108
Asynchronous message exchange, synchronous versus, 26–28
avoid_zombies(), 168

Barrier synchronization, 197
begin(), 78
Blocking, 56, 58, 61, 62, 65
broadcast(), 229

C++ code, displaying, 50
C++ iostreams, 76
C++ wrapper facade layer, 14
cancel_all(), 189
Casts, 176
child(), 163
cleanup(), 203
clone(), 74
close(), 55, 170, 189
clr_bit(), 143
clr_flags(), 74
COM+, 9
command_line(), 167
Commercial off-the-shelf (COTS), 9, 10
Common middleware services layer, role of, 9
Communication design
connectionless versus connection-oriented protocols, 23–26
message passing versus shared memory, 28–31
synchronous versus asynchronous message exchange, 26–28

Communication dimensions, 6
Communication domain, 36
`complete()`, 58
Component, 16
Component Configurator Pattern, 16–17
Concrete class, 30
Concurrency design, 6
  concurrent servers, 105–106
  iterative servers, 103–105
  limitations with OS, 135–136
  process/thread spawning strategies, 112–114
  processes versus threads, 109–112
  reactive servers, 106–108
  real-time scheduling, 119–121
  task- versus message-based architectures, 121–122
  threading models, 114–119
  time-shared scheduling, 119–121
Concurrency framework, 15
Concurrent servers, 105–106
Condition variables, 133–134
Configuration dimensions, 6
`connect()`, 56, 58
Connection establishment and termination, Socket API, 35
Connection establishment framework, 15
Connectionless versus connection-oriented protocols, 23–26
`const_cast`, 176
Container classes, 155
Contention scope, 114
Cooperative cancelation, 190–191, 197
`copy()`, 74
CORBA, 9, 29
  Common Data Representation (CDR), 77
Cost containment, 11
`CreateProcess()`, 109, 161
`CreateThread()`, 110, 129
`creation_flags()`, 168
`CreateProcess()`, 128

Data framing strategies, 24
Data transfer mechanism, Socket API, 35
Data-mode socket, 56
Deadlocks, 107
Debugging macros, 93
Demarshaling, 9, 11, 76–77
`dequeue_head()`, 227–228
Descriptor, *see* Handles
`disable()`, 53, 62
Dispatching framework, 15
Distributed shared memory (DSM), 30–31
Distribution middleware, role of, 8–9
Domain analysis, 5
Domain-crossing penalty, 91
Domain-specific middleware services layer, role of, 9
Double-Checked Locking Optimization pattern, 191, 192, 203
`duplicate()`, 74
`dynamic_cast`, 176

Eager spawning, 112
`echo_server()`, 237
`echo_server`, 37–38
Efficiency issues, 46
`enable()`, 53, 62
`end()`, 78
Endpoints, 34
`enqueue_tail()`, 226–227
`errno`, 130, 202
Error macros, 93
Error propagation strategies, 137
Escape hatches, 237–238
Event demultiplexing framework, 15

synchronous, 125–127
Event loops, 125
exit(), 111, 128, 189
exit_code(), 162
ExitProcess(), 111, 128
ExitThread(), 129

FD_CLR(), 127
FD_ISSET(), 127
FD_SET(), 127
fd_set, 126, 141–143, 148–150
FD_ZERO(), 127
fdset(), 143
First-in, first-out (FIFO), 120, 198
for loop, 147
fork(), 109, 128, 161, 164
Framework layer, 15–16

get_handle(), 53, 238
get_host_name(), 52
get_local_addr(), 55
get_option(), 55
get_port_number(), 52
get_remote_addr(), 55, 85, 88
gethostbyname(), 84
getpid(), 162
GetThreadPriority(), 130
good_bit(), 78

Half-Sync/Half-Async pattern, 16, 112, 206
handle_connections(), 84, 91, 93, 156
handle_connections, 177–179
handle_data(), 84, 91, 94, 146, 156–157, 196–197
Handles, 34
errors and, 37–40
hash(), 51
Hook methods, Logging_Server, 83–84
Host infrastructure middleware layer, role of, 8, 10–13

Hybrid-threading model, 116–117

instance(), 170, 189
Internet Protocol (IP), 24
Interprocess communication (IPC), local and remote, 33
iovec structure, 63
is_set(), 143
Iterative servers, 103–105
Iterative_Logging_Server, 91–95
Iterator pattern, 148

Java Packages, 8
Java RMI, 9
JAWS, 265
Jitter, 11
join(), 189

Kernel-threading model, 115–116
kill(), 161, 163

Last-in, first-out (LIFO), 218
Leader/Followers pattern, 112
length(), 74
Lightweight processes (LWPs), 116–117
Linearization, 76
Local context management, Socket API, 35
Local shared memory, 29
Lock-step sequence, 27
Locking, 203, 210–212, 218–224
log_record(), 91
Logging service
asynchronous request/response protocol, 28
client application, 95–98
example of, 17–19
initial, 80–95
message framing protocol, 86
message passing, 31
TCP/IP connection, 25
Logging service, implementing

ACE_InputCDR, 72–80
ACE_OutputCDR, 76–80
Logging_Client::send(), 95–97
Logging_Handler, 86–91
log_record(), 91
recv_log_record(), 87–90
write_log_record(), 90–91
Logging_Process, 180–182
Logging_Server, 81–86
handle_connections(), 84, 91
handle_data(), 84, 91
hook methods, 83–84
make_log_file(), 85–86
open(), 83–84
run(), 83
wait_for_multiple_events(), 84

Macros
debugging and error, 93
guard, 216
main(), 94, 157
make_log_file(), 85–86, 92, 156
Marshaling, 9, 11, 76–77
max_set(), 143
Memory management unit (MMU), 109
Memory-mapped files, 29–30
Message exchange, synchronous versus asynchronous, 26–28
Message passing versus shared memory, 28–31
Message(s)
composite, 73
framing protocol, 86
simple, 73
Message-based concurrency architecture, 121–122
Message-oriented middleware (MOM), 29
Message_Queue, 223–229
Microsoft Windows, 34
Middleware standards, 263–264
Monitor Object pattern, 133, 224
msg_priority(), 74
msg_type(), 74
Multiplexing connections, 24–25
Multiprocessing
advantages and disadvantages of, 109–110
mechanisms, 127–128
spawning strategies, 112–114
Multiprocessing wrapper facades
ACE_Process, 161–165
ACE_Process_Manager, 169–182
ACE_Process_Options, 165–169
overview of, 159–161
Multithreading
advantages and disadvantages of, 110–112
mechanisms, 129–130
models, 114–119
spawning strategies, 112–114
Multithreading wrapper facades
ACE_Sched_Params, 186, 198–201
ACE_Thread_Manager, 186–198
ACE_TSS, 186, 187, 201–205
overview of, 185–187
Mutual exclusion (mutex) locks, *see also* ACE_Condition_Thread_Mutex, ACE_Null_Mutex, ACE_Process_Mutex, ACE_RW_Process_Mutex, ACE_RW_Thread_Mutex, and ACE_Thread_Mutex, 105, 132, 134

N:1 user-threading model, 114–115
N:M hybrid-threading model, 116–117
Nagle's algorithm, 55, 63, 64
Network addressing, Socket API, 36
Network services, library of, 16–17
Networked applications
challenges of, 1–4

design dimensions, 5–7
example of, 2–3
next(), 74
next_priority(), 199
Nonblocking, 56, 58, 61, 62, 65
Nonmultiplexing connections, 25
num_set(), 143

Object Lifetime Manager pattern, 218
Object Request Brokers (ORBs), 9
Object, differences between a thread and an, 194
Object-oriented middleware
benefits of host infrastructure, 10–13
layers, 7–10
role of, 7, 9–10
On-demand spawning, 113
1:1 kernel-threading model, 115–116
open(), 55, 66, 155, 170
Logging_Server, 83–84, 92
operator!=(), 51
operator(), 148
operator->(), 203
operator<<(), 78–79
operator==(), 51
operator>>(), 78–80
Options management, Socket API, 36
OS adaptation layer, 13–14

parent(), 163
pass_handle(), 167
Passive-mode socket, 56
Pattern
Acceptor-Connector, *see* Acceptor-Connector pattern
Active Object, *see* Active Object pattern
Component Configurator, *see* Component Configurator pattern
Double-Checked Locking Optimization, *see* Double-Checked Locking Optimization pattern
Half-Sync/Half-Async, *see* Half-Sync/Half-Async pattern
Iterator, *see* Iterator pattern
Leader/Followers, *see* Leader/Followers pattern
Monitor Object, *see* Monitor Object pattern
Object Lifetime Manager, *see* Object Lifetime Manager pattern
Pipes and Filters, *see* Pipes and Filters pattern
Proactor, *see* Proactor framework
Reactor, *see* Reactor framework
Singleton, *see* Singleton pattern
Thread-Safe Interface, *see* Thread-Safe Interface pattern
Wrapper Facade, *see* Wrapper Facade pattern
PEER_ADDR, 57
PEER_STREAM, 57
Pipes and Filters pattern, 16
poll(), 126
Portability, 46, 52, 135, 136, 152, 164, 176
lack of, Socket API, 41–43
Ports, ephemeral, 51
POSIX, 161, 163
prepare(), 162
prev(), 74
prev_priority(), 199
Primitive types, 77
arrays of, 77
printf(), 93
Priority inversion, 25
priority_max(), 199
priority_min(), 199
Proactive servers, 108

Proactor framework, 15
Process, *see also* Multiprocessing
  contention scope, 114
  lifetime operations, 128
  pool, 106
  property operations, 128
  synchronization operations, 128
Process-per-connection, 171–180
Protocol stacks, 7
Protocols
  connectionless versus connection-oriented, 23–26
  defined, 23
  family, 36
`pthread_cancel()`, 129, 187
`pthread_create()`, 110, 129
`pthread_exit()`, 129
`pthread_getschedparam()`, 130
`pthread_getspecific()`, 130
`pthread_join()`, 129
`pthread_key_create()`, 130
`pthread_kill()`, 188
`pthread_setschedparam()`, 130
`pthread_setspecific()`, 130
`pthread_testcancel()`, 187

Quality of service (QoS) requirements, 10–11

Race conditions, 130–132
`rd_ptr()`, 74
Reactive servers, 106–108
Reactor framework, 15
`read_n()`, 76
Readers/writer locks, 132–133
Real-time scheduling, 119–121
`recv()`, 56, 62
`recv_log_record()`, 87–90
`recv_n()`, 62
`recvv_n()`, 62
`reinterpret_cast`, 176
`release()`, 74
Request/response protocols, asynchronous and synchronous, 26–28
`reset()`, 143
Reuse, 10
Round-robin, 120, 198–199
RPC, 29
`run()`, 83, 173, 174
`run_master()`, 174–175
`run_svc()`, 194–196
`run_worker()`, 175–176

Scheduler activations, 116
Scoped Locking, 203, 210–212, 224
`select()`, 107, 126–127, 140, 151–152
Semantic variations, 187
Semaphores, *see also* `ACE_Null_Semaphore`, `ACE_Process_Semaphore`, and `ACE_Thread_Semaphore`, 105, 133–134
`send()`, 56, 62
`send_n()`, 62
`sendv_n()`, 62
Serialization, 30
Service configurator framework, 16–17
Service dimensions, 6
Service initialization framework, 15
`set_bit()`, 143
`set_flags()`, 74
`set_handle()`, 53, 238
`set_handles()`, 167
`set_option()`, 55
`set_process_attributes()`, 168
`setenv()`, 167
`seteuid()`, 168
`setruid()`, 168
`SetThreadPriority()`, 130
Shared memory
  C++ objects and, 30
  distributed, 30–31

local, 29–30
message passing versus, 28–31
signal(), 229
Singleton pattern, 191, 192
size(), 74
Sleep locks, 133
sockaddr, 49
Socket API
address family, 37
Connection establishment and termination, 35
Data transfer mechanisms, 35
limitations of, 37–43
local context management, 35
network addressing, 36
nonportable and nonuniform, 41–43
options management, 36
protocol family, 36
role of, 34
Socket wrapper facades
ACE_Addr, 49–52
ACE_INET_Addr, 49–52
ACE_IPC_SAP, 52–53
ACE_SOCK, 54–55
ACE_SOCK_Acceptor, 64
ACE_SOCK_Connector, 56–60
ACE_SOCK_IO, 60–64
ACE_SOCK_Stream, 60–64
benefits of, 46
overview of, 45–49
relationships between, 47
structure of, 46
using traits for, 57
Sockets, 34
spawn(), 162, 170, 189
spawn_n(), 170, 189
Spawning
of threads, 195
of worker processes, 175, 178
strategies for processes and threads, 112–114
Spin locks, 132
Stand-alone applications, example of, 2
Standards, open, 10
static_cast, 176
steal_contents(), 78
Strategic focus, 9
Strategized Locking pattern, 215
Streams framework, 16
string_to_addr(), 52
structtimeval, 127
sync(), 143
Synchronization mechanisms, 130–134
Synchronization wrapper facades
ACE_Condition_Thread_Mutex, 208, 229–230
ACE_Guard, 208–212, 216–217
ACE_Null_Condition, 208, 230–231
ACE_Null_Mutex, 208, 212–217
ACE_Null_Semaphore, 208, 222–229
ACE_Process_Mutex, 208, 212–217
ACE_Process_Semaphore, 208, 222–229
ACE_Read_Guard, 208–212, 219–221
ACE_Recursive_Thread_Mutex, 231–233
ACE_RW_Process_Mutex, 208, 219–221
ACE_RW_Thread_Mutex, 208, 219–221
ACE_Thread_Mutex, 208, 212–217
ACE_Thread_Semaphore, 208, 222–229
ACE_Write_Guard, 208–212, 219–221
overview of, 207–209

Synchronous event demultiplexing, 125–127
Synchronous event demultiplexing, wrapper facades
ACE::select(), 141, 151–152
ACE_Handle_Set, 141–147
ACE_Handle_Set_Iterator, 147–151
ACE_Handle_Set, 140
ACE_Handle_Set_Iterator, 140
overview of, 139–141
Synchronous versus asynchronous message exchange, 26–28
Syntactic variations, 187
System contention scope, 114
System V STREAMS, 73
System V UNIX shared memory, 29

TAO, 264–265
Task, 205
Task framework, 15
Task-based concurrency architecture, 121
Template method, Logging_Server::run(), 83
terminate(), 162
TerminateProcess(), 128, 161
TerminateThread(), 129
testcancel(), 189
THR_DETACHED, 190
THR_JOINABLE, 190
thr_kill(), 188
THR_NEW_LWP, 190
THR_SCOPE_PROCESS, 190
THR_SCOPE_SYSTEM, 190
Thread, *see also* Multithreading
difference between an object and a, 194
lifetime operations, 129
pool, 106
property operations, 130
spawning of, 195
specific storage, 130
synchronization operations, 129
Thread-per-connection, 106, 191–198
Thread-per-request concurrent server, 105–106
Thread-Safe Interface pattern, 224, 226
Thread-specific storage (TSS), *see also* ACE_TSS, 130
Timed socket operations, 56, 58, 60–62, 65
Timeouts, 62–64, 152
timeval, 127
TlsAlloc(), 130
TlsGetValue(), 130
TlsSetValue(), 130
total_length(), 74, 78
TP4, 24
Traits, 57
Transmission Control Protocol (TCP), 24–26
TSS, *see* Thread-specific storage
Type errors, 46

UNIX, 34, 35, 161, 163
unmanage(), 162
User Datagram Protocol (IP), 24
User-threading model, 114–115

Virtual methods, 30

wait(), 128, 162, 170, 189, 229
wait_for_multiple_events(), 84, 93, 144–145, 155–156
WaitForMultipleObjects(), 107, 126, 128, 129, 161
WaitForSingleObject(), 128, 129, 161
waitpid(), 128, 161
Wildcard, 51
Win32, 161
working_directory(), 167
wr_ptr(), 74

Wrapper Facade pattern, 14, 45, 85
    hide platform differences, 248–254
    hierarchies to enhance clarity and extensibility, 246–248
    optimize for efficiency, 255–257
    simplify for common case, 238–246
    to enhance type safety, 236–238
`write_log_record()`, 90–91
`write_n()`, 76
`writev()`, 63
`WSASend()`, 63

XTP, 24