

Index

Note: Page numbers followed by *f* and *t* indicate figures and tables, respectively.

- A**
- abstract design, reuse of, 7
 - `accept()`, for `ACE_SOCK_Acceptor`
 - acceptor and, 135
 - for connection requests, 136
 - interruption of, by signals, 136–137
 - `accept_handle()`, for
 - `ACE_Asynch_Acceptor`, 199
 - acceptor. *See also* `ACE_SOCK_Acceptor`
 - in Acceptor-Connector framework (*See* `ACE_Acceptor`)
 - definition of, 123
 - error handling and, 135
 - instantiating, 145
 - `open()` method of, 144–145, 146
 - port listening with, 135, 145
 - `register_handler()` for, 145
 - unicast mode and, 209
 - Acceptor-Connector framework, 169–182
 - `ACE_Acceptor` in, 169–171
 - `ACE_Svc_Handler` in, 171–172
 - classes of, 169, 169*f*
 - file I/O in, 213
 - SPIPE in, 214
 - ACE
 - benefits of, 5–6
 - building, 27–30
 - character types in, 19, 20*t*
 - developer forums for, 22
 - distribution of, 26–27
 - history of, 3–5
 - including, in applications, 30–31
 - memory allocation macros in, 19, 20*t*
 - organization of, 6–7
 - reference documentation for, 21
 - technical support services for, 22
 - versions of, 25–26
 - `ACE_Acceptor`
 - connection accepted by, 172–173, 172*f*
 - initialization of, 423
 - role of, 169–170
 - `ACE_Activation_Queue`. *See also* `Activation Queue`
 - in half-sync/half-async thread pool, 334
 - `ACE_Addr`, about, 125–126
 - `ACE_Addr::sap_any`, 131
 - `ACE_Allocator` interface
 - `ACE_Malloc` and, 350–351, 369
 - for containers, 115, 116–119
 - `ACE_Allocator_Adapter`, 359, 361

- ACE_ARGV, 85–86
- ACE_Array. *See* array
- ACE_ASSERT macro, 43*t*
- ACE_Asynch_Acceptor
 - about, 198–200
 - for passive connection establishment, 198
 - on POSIX systems, 202
- ACE_Asynch_Connector
 - about, 199–200
 - for active connection establishment, 198
 - on POSIX systems, 202
- ACE_Async_Timer_Queue_Adapter, 449–450
- ACE_Atomic_Op, 293
- ACE_At_Thread_Exit, 277
- ACE_Barrier, 307
- ACE_Based_Pointer_Basic, 358–359
- ACE_BINDING_SET, values in, 473
- ACE_Bounded_Stack. *See* bounded stack
- ACE_Cleanup, 15
- ACE_Condition, 259
- ACE_Configuration, 77
- ACE_Configuration_Heap, 83, 84
- ACE_Configuration_Win32Registry, 83, 84–85
- ACE_Connector, 177. *See also* ACE_SOCKET_Connector
- ACE_Data_Block, 401, 402
- ACE_DEBUG macro
 - about, 38–39, 43*t*
 - wrapping, 48–51
- ace directory, 27
- ACE_DLList container. *See* doubly linked list
- ACE_Dynamic_Message_Queue, 266, 266*t*
- ACE_Equal_To, specialization in, 89
- ACE_ERROR_BREAK macro, 43*t*
- ACE_ERROR_INIT macro, 43*t*
- ACE_ERROR macro, 38–39, 43*t*
- ACE_ERROR_RETURN macro, 43*t*
- ACE_Event_Handler. *See also* ClientService handler; event handler
 - ACE_Reactor pointer in, 146
 - I/O event handles in, 144
 - for process termination, 230
 - Reactor event handlers and, 142
 - for signal callbacks, 239
 - in timer event listener, 441
- ACE_FACTORY_DEFINE macro, 424, 427, 429
- ACE_FIFO classes, 214
- ACE_FILE_Addr, 125–126, 213, 392
- ACE_FILE_Connector, 214
- ACE_FILE_IO, 214
- ACE_Fixed_Stack, 95, 96–97, 98
- ACE_FlReactor extension, 186
- ACE_Future, 323
- ACE_Future_Observer, 323–324
- ACE_Get_Opt
 - altering behavior of, 80–81
 - command line arguments and, 78–82
 - getopt() vs., 78–82
 - parsing with
 - at arbitrary index, 80–81
 - error reporting during, 81
 - purpose of, 77
 - for string parsing, 85
- ACE_Guard, 255, 256, 256*t*. *See also* guards
- ACE_GUARD macro, 256
- ACE_GUARD_RETURN macro, 256
- ACE_Handler, 191, 192. *See also* completion handler
- ACE_Hash, 89
- ACE_Hash_Map_Manager, 108–111. *See also* hash map(s)
- ACE_HAS_LAZY_MAP_MANAGER, 104, 108
- ACE header files, including, 30–31
- ACE_HEX_DUMP macro, 43*t*
- ACE_INET_Addr. *See also* address
 - address extracted with, 137
 - for client, 125–126
 - as connect() parameter, 131
 - constructor of, 129
 - for server, 135
 - Reactor-based, 145
 - set() methods of, 129–130
 - for UDP/IP, 207, 209
- ACE_Ini_ImpExp, configuration information saved with, 85
- ACE kits, availability of, 21–22
- ACE_Less_Than, 111, 114–115
- ACE_Less_Than functor, 111, 114–115
- ACE library, 31, 36*t*
- ACE_Local_Mutex, as token, 297
- ACE_Log_Msg
 - flag values for, 57*t*

- log message format in, 45
- methods of, 47, 48*t*
- ACE_Log_Msg_Callback, 61–64
- ACE_Log_Record, 64, 65*t*
- ACE_Malloc
 - about, 350–351
 - ACE_Allocator and, 359
 - for containers, 119
 - map interface for, 351–352
 - memory protection interface for, 352
 - parameters for, 350
 - persistence with, 352–356
 - sync interface for, 352
- ACE_Malloc_T, 357
- ACE_Map_Manager. *See* map(s)
- ACE_MEM classes, for intrahost communication, 214
- ACE_Mem_Map, 375–376
- ACE_Message_Block
 - allocation of, in asynchronous I/O, 197
 - in asynchronous read operations, 193–194, 195, 196
 - in asynchronous write operations, 195, 196
 - data handled with, standardization on, 203–204
 - dequeuing, 154
 - dequeueing, 154
 - in leader/follower thread pool, 338
 - in message passing, 260
 - in one-way stream, 385
 - outstanding operations and, 197
 - queueing, 152–153, 180
 - queuing, 152–153, 180
 - releasing, 155, 195
 - with semaphore, 303
- ACE_Message_Queue
 - access to, with `msg_queue()`, 175
 - in event loop, 179
 - flushing, 155
 - in half-sync/half-async thread pool, 330–332
 - in input processing, 152–153
 - notification strategy on, 179
 - as shared data buffer, 303
 - in threads, 260
- ACE_MMAP_Memory_Pool, backing up, 352
- ACE_MMAP_Memory_Pool_Options, 358–359, 360*t*
- ACE_Module
 - command module from, 402–403
 - tasks in, 383
- ACE_Msg_WFMO_Reactor implementation, 183–185
- ACE_Mutex, 252. *See also* mutex
- ACE_Name_Binding. *See also* Name_Binding
 - memory management for, 464
- ACE_Name_Options, 458, 458*t*, 460
- ACE_Naming_Context. *See also* naming context
 - about, 457–459
 - binding in, 474–479
- ACE_Null_Mutex, 111
- ACE_Object_Manager. *See* Object Manager
- ACE_OSTREAM_TYPE, 59
- ACE_PI_Control_Block, 357
- ACE_Pipe, 214
- ACE_POSIX_Proactor implementation, 202
- ACE_Priority_Reactor implementation, 185
- ACE_Proactor. *See also* Proactor framework
 - in completion handling, 201
 - implementations of, 201–202
- ACE_Process. *See also* process
 - about, 219–220
 - spawning from, 220–221
- ACE_Process_Manager, 226–231
- ACE_Process_Mutex
 - for hash maps, 361
 - synchronization with, 231–234
- ACE_Process_Options, for slave process, 220–221
- ACE_QtReactor extension, 186
- ACE_RB_Tree. *See* self-adjusting binary trees
- ACE_Reactor. *See also* reactor
 - implementations of, 182–186
 - instance of, 146
 - pointer to
 - in event handler, 146
 - passing to event handler, 147
 - as timer dispatcher, 440
- ACE_Reactor_Notification_Strategy, 178–179
- ACE_READ_GUARD macro, 256
- ACE_READ_GUARD_RETURN macro, 257
- ACE_Recursive_Thread_Mutex, 16, 108, 298. *See also* recursive mutex
- ACE_Registry_ImpExp, 85
- ACE_RETURN macro, 43*t*

- ACE_RW_Mutex, 292
- ACE_RW_Thread_Mutex, 292
- ACE_Select_Reactor implementation, 183
- ACE_Service_Handler, 191
- ACE_Service_Object
 - dynamic services from, 427
 - static services from, 421
- ACE_Sig_Action
 - for action registration, 237
 - creation of, 238
- ACE_Sig_Guard, 246–247
- ACE_Sig_Handler
 - for event handler registration, 239
 - in reactor implementation, 247
 - in thread signaling, 279
- ACE_Sig_Handlers, for multiple handlers, 245
- ACE_Sig_Set, 158, 238
- ACE_SOCKET_Acceptor. *See also* acceptor
 - in ACE_Acceptor, 170
 - for connection acceptance, 136–138
 - for multiple connections, 143–144
 - port listening with, 135, 145
- ACE_SOCKET_CODgram, for UDP unicast, 210–211
- ACE_SOCKET_Connector. *See also* connector
 - connect () method in, 130
 - constructors for, 130–132
 - nonblocking connection operation with, 132
 - quality-of-service parameters with, 132
 - for socket connection, 126
- ACE_SOCKET_Dgram, 209, 210
- ACE_SOCKET_Dgram_Bcast, 212
- ACE_SOCKET_Dgram_Mcast, 212–213
- ACE_SOCKET_Stream. *See also* stream
 - access, with peer (), 175
 - in ACE_Svc_Handler, 172
 - arg () result in, 403
 - closing, 155
 - send and receive methods in, 127
 - server connection of, 126
 - timeout with, 132
 - wrapping, in ClientService, 147
- ACE_SPIPE, 214
- ACE_SPIPE_Addr, 125–126
- ACE_STATIC_SVC_DEFINE macro, 424
- ACE_STATIC_SVC_REGISTER macro, 426
- ACE_STATIC_SVC_REQUIRE macro, 425, 426
- ACE_Stream, 379*f. See also* stream
 - creation of, 378
 - as linked list, 385
 - modules in, 383
 - in one-way stream, 380, 381–386
- ACE_Svc_Handler. *See also* Client handler
 - about, 171–172
 - from ACE_Connector, 177
 - UDP classes with, 208
- ACE_SV_Semaphore_Complex, vs.
 - ACE_Process_Mutex, 234
- ACE_Synch_Read_Stream, 191
- ACE_Synch_Result, 191
- ACE_Synch_Write_Stream, 191
- ACE_Task
 - message queueing in, 334
 - message queuing in, 334
 - multithreaded queuing in, 330
 - multithreaded queuing in, 330
 - stream tasks from, 382
 - thread creation from, 260
- ACE_Task_Base
 - in half-sync/half-async thread pool, 334
 - Scheduler from, 320
 - thread creation from, 250
- ACE_Thread_Manager, 277
- ACE_Thread_Mutex
 - in ACE_Condition, 259
 - as consistency constraint, 252–254
 - for maps, 108
- ACE_Timer_Heap
 - for active timer, 447–448
 - memory allocation in, 439
 - for signal timer, 449–450
- ACE_Timer_Queue, 439
- ACE_Timer_Wheel, 439
- ACE_Time_Value, 132, 323
- ACE_TkReactor extension, 186
- ACE-Token
 - locking with, 297
 - strict ordering with, 254
- ACE_TP_Reactor implementation, 185, 343–345
- ACE_TRACE macro
 - about, 39–42, 44*t*
 - customizing, 51–55
 - features enabled in, 42
- ACE_TSS, 310

- ACE_Unbounded_Queue, 98–100, 329. *See also* queue(s)
- ACE_Unbounded_Stack, 94, 95–96, 97. *See also* unbounded stack
- ACE_UNIX_Addr, 125–126
- ACE_Unmanaged_Singleton, 435
- ACE_WFMO_Reactor implementation
 - about, 183–185
 - event loop integration with, 205
 - proactor integration with, 204
 - thread pool support in, 345
- ACE_Win32_Proactor, 201–202, 204–205
- ACE_wrappers directory, 26
- ACE_WRITE_GUARD macro, 256
- ACE_WRITE_GUARD_RETURN macro, 257
- ACE_WString, 462, 464
- ACE_XtReactor extension, 186
- acquire() function
 - in deadlock detection, 301
 - for mutex, 252, 258
 - semaphores and, 304
 - vs. guard, 255
- acquire_read(), for readers/writer lock, 292
- acquire_write(), for readers/writer lock, 292
- activate()
 - for active object, 314
 - for active-timer dispatcher, 449
 - priority specified in, 274
 - for thread of control, 320
 - thread started with, 251
- Activation Queue
 - in Active Object pattern, 316
 - in half-sync/half-async thread pool, 333–338
 - in Scheduler, 321
- Active Object pattern
 - about, 314, 314^f
 - collaboration in, 316, 317^f
 - for cooperative processing, 313
 - participants in, 315–316, 315^f
 - using, 317–324, 319^f
- Active Object thread, exit of, 321
- active timer dispatcher, 447–449
- active-timer queue, 447
- Adapter pattern, 16
- ADAPTIVE Communication Environment. *See* ACE
- address. *See also* ACE_INET_Addr
 - from ACE_INET_Addr, 137
 - addresses() for, 200
 - for client, 125–126
 - definition of, 123
 - for shared memory pool, 356–357, 369
 - in UDP broadcast, 211
 - in UDP multicast, 212
 - in UDP unicast, 209–210
- address space, protection of, 349
- addr_to_string(), 137
- agent implementation, aggregation of, 321
- algorithms
 - in C++ library, 90, 95
 - reuse of, 7
- allocators. *See also* ACE_Allocator; ACE_Malloc; shared memory allocator
 - about, 115–119, 116^t
 - cached, 116–119
 - type awareness and, 116
- answer_call()
 - for AnswerIncomingCall, 386–387
 - implementation of, 416
 - implentation of, 416
- AnswerCallModule, 409–411
- AnswerIncomingCall, 386–387
- answering machine application, one-way stream
 - for, 378–397
- API
 - in C, vs. C++, 5
 - finalize from, 18
 - initialize from, 18
 - of Naming Service, 459
 - vs. OS methods, 9–10
- applications
 - building, 31–35
 - networked, difficulty in writing, 5
- apply(), for exit handler, 277
- apps directory, 27
- arbitrary index, parsing at, 80–81
- architecture, layered, of ACE toolkit, 6–7
- arg(), for command module, 403, 409
- argument vector. *See* command line argument vector
- array, 97, 100–101
- associative containers, 104–115
- asynchronous cancelability, 285, 287–288
- asynchronous I/O model. *See also* Proactor frame-

- work
 - about, 187
 - benefits of, 188–189
 - steps in, 188
- asynchronous layer, of half-sync/half-async thread
 - pool, 326–327
 - code for, 327–329
- asynchronous signals, in multithreaded programs, 282
- asynchronous timer dispatcher, 449–450
- `at_exit()`, for exit handler registration, 277–278
- `atoi()`, for `PROC_LOCAL` context, 464
- atomic operation wrapper, 293–297
- attributes
 - of name options, 460
 - of threads, 267–268

B

- backlog argument, of `open()`, 200
- barriers, for thread synchronization, 301–303
- basic task, in one-way stream, 387–392
- `become_leader()`, 340, 343
- `begin()`, for maps, 106
- Berkeley Software Distribution (BSD) Sockets programming. *See* Sockets programming
- beta versions, 25
- BFO (bug fix only) versions, 26
- bidirectional stream, 397–418. *See also* command stream
- binary ordering functor, 111
- binary semaphore, *vs.* mutex, 303
- `bind()`
 - acceptor and, 135
 - allocator pointer in, 366
 - vs.* `rebind()`, 462
- binding set
 - iteration over, 473–474
 - name values in, 474
- bin directory, 27
- block
 - with `getq()`, 265
 - timed, on condition variable, 257
- block size, for cached allocators, 117
- bounded set, 101–103
- bounded stack, 94–96
- Bridge pattern, 182–183, 201

- broadcast connection, in UDP, 208, 211–212
- BSD (Berkeley Software Distribution) Sockets programming. *See* Sockets programming
- bucket size, in hash map, 109
- buffer
 - `ACE_Message_Queue` as, 303
 - for `addr_to_string()`, 137
 - allocation of, with `recvv()`, 134–135
 - counter for, 294
 - noncontiguous, 133
 - `recv_n()` method and, 127
 - `send_n()` method and, 127
- bug fix only (BFO) versions, 26
- build
 - of ACE, 27–30
 - of applications, 31–35
 - in Microsoft C++, 34–35
 - from multiple source files, 32
- `bytes_to_read` argument, 200

C

- C programming language
 - for APIs, 5
 - memory allocation in, 18–19
 - typeless pointers in, 88
- C++ programming language. *See also* Microsoft Visual C++
 - for APIs, 5–6
 - compilers for, differences among, 11–19
 - data types in, 14, 15†
 - in heap memory allocation, 18–19
 - templates in, 11–14
 - containers in, 87, 93
 - memory allocation in, 18–19
 - wide characters in, 19
- cached allocators, 116–119
- `calculate_timeout()`, on timer queue, 441–443
- callbacks. *See also specific methods*
 - deleting, 61
 - event handler (*See also* `handle_close()`; `handle_input()`)
 - in Reactor implementation, 141
 - return values of, 148, 149†
 - inheritance in, 61
 - for I/O operations, 194
 - for logging, 61–64
 - for logging server, direct communication with, 68–69

- for process termination, 229
 - queueing, with notifications, 159
 - queuing, with notifications, 159
 - for signals, 236–237, 238, 245
- cancel ()
 - for outstanding I/O operations, 197
 - for timer dispatcher, 441, 444
 - for timer queue, 451
- cancellation (), for upcall handler, 451, 453
- cancellation, of threads, 284–288
- cancel_task (), for thread cancellation, 286, 288
- cancel_wakeup (), 181
- cancel_wakup (), 181
- ChangeLog file, 26–27
- character sets, narrow vs. wide, 19–21
- char_rep (), memory allocation with, 464
- child(pid_t parent), 226
- child process. *See* slave process
- class(es)
 - in Acceptor-Connector framework, 169, 169f
 - reuse of, by layers, 7
 - template argument, types defined in, 13–14
 - vs. namespace, 10
- class libraries
 - extension of, 7–8
 - reuse by, 7
- class templates, in compilers, differences among, 11–14
- cleanup (), 16
- cleanup handlers, during cancellation, 285
- client. *See also* I/O sources
 - addressing in, 125–126
 - constructing, 124–129
 - querying with, 125, 129
 - with iovec structures, 133–134
 - send and receive in, 127, 137–138
- ClientAcceptor handler. *See also* connection-accepting handler
 - declaration of, 143
 - handle_input method of, 146–147
 - instantiation of, in Reactor-based server, 145
- Client handler. *See also* ACE_Svc_Handler
 - declaration of, 177–178
 - methods in, 178–181
- ClientService handler. *See also* service handler
 - in Acceptor-Connector framework, 170
 - creation of, 172
 - declaration of, 171
 - handle_output () method for, 175
 - open () method for, 172–173
 - in Reactor framework
 - creation of, 147
 - declaration of, 149–150
 - peer () method for, 147
 - queueing in, 152–153
 - queuing in, 152–153
- clone (), for message blocks, 263
- close ()
 - for command task, 405–406
 - for stream tasks, 388
 - for unmanaged singleton, 435
- close_writer (), for Client handler, 180
- code
 - conditionally compiled, 10
 - porting, to multiple operating systems, 8–10
 - private method for, 203–204
 - reuse of, templates for, 11–12
- collectCallerIdModule, 383
- CollectCallerId task, 392
- command line
 - processing, 79
 - static service configuration at, 425
- command line argument
 - ACE_Get_Opt and, 78–82
 - in one-way stream, 379–380
 - ordering, 81–82
 - runtime behavior altered with, 77
 - for slave process, 223
- command line argument vector
 - building, 85–86
 - conversion to string, 85
 - processing, 78–82
- command line options
 - arguments for, 79
 - defining, 79
 - long, 78
 - for naming context, 459
 - operator () for, 79
 - parsing, 78
 - + or - in, 82
 - short, 78
- command module
 - for command stream, 402–403
 - methods in, 403
 - retrieving, 410

- socket pointer in, 400, 402–403, 409, 410
- CommandModule, 402–403
- Command object, 401–402
- Command pattern, 315
- command stream, 397–418, 397*f*
 - Command object for, 401–402
 - implementations of, 409–414
 - initialization of, 415
 - methods of, 399–401
 - pointer to, in peer attribute, 415
 - using, 414–418
- CommandStream task, 398–399
- CommandTask, 404–409
- compiler(s)
 - ACE build and, 29–30
 - differences among
 - data types in, 14, 15*t*
 - in heap memory allocation, 18–19
 - templates in, 11–14
 - in portability, 9
 - template applied in, 176
 - template instantiation and, 11–12, 71
- compiler macros, 10
- compile time, service handler classes derived at, 203–204
- completion handler
 - ACE_Handler and, 191
 - cleanup of, 198
 - deletion of, 194, 197
 - handle passed to, 191
 - in open() , 193
 - registration of, 194
- completion port, 201
- Component Configurator pattern, 420
- concrete design, reuse of, 7
- concurrency
 - in multithreaded I/O model, 188
 - in self-adjusting binary trees, 111
- condition variables
 - in half-sync/half-async thread pool, 329
 - in intertask communications, 257–260
 - mutex reference in, 259
 - semaphore vs., 303
- configuration, service. *See* service, configuration of
- configuration files
 - for ACE build, 28, 29*t*
 - for logging client proxy, 66
 - for logging server, 66
 - in Microsoft Visual C++, 34–35
 - reading, for runtime behavior, 77
 - service configuration without, 434
 - for services, reprocessing, 431–432
 - XML for, 432–433
- configuration information, accessing, 83–85
- connect()
 - for client, result from, 131
 - for socket connection, 126, 130
 - vs. constructor, 130
- connect() function, for client connection, 125
- connection(s)
 - accepting
 - with ACE_Acceptor, 172–173, 172*f*
 - with ACE SOCK_Acceptor, 137, 143–149
 - in bidirectional stream, 410
 - ACE SOCK_Connector for, 126
 - ACE SOCK_Stream and, 126
 - address in, 126
 - in Proactor framework, 198
 - processing, 138–139, 143
 - to Reactor-based server, 146–147
 - service handler for (*See* service handler)
 - in UDP, 208–213
- connection-accepting handler. *See also* ClientAcceptor handler
 - declaration of, 143
 - handle association of, removing from reactor, 143
 - separation of, 144
- connection requests
 - accepting, 143
 - accept() method for, 136
 - connect() method for, 126, 130
 - event handler for, 143
 - timeout on, 130–131
- connector. *See also* ACE SOCK_Connector
 - definition of, 123
 - unicast mode and, 209
- constructor
 - of ACE_INET_Addr, 126, 129
 - of ACE SOCK_Acceptor, 135
 - of ACE SOCK_Connector, 130–132
 - flexibility of, 129
 - vs. connect() , 130
- containers. *See also specific containers*
 - ACE, vs. STL, 87
 - ACE_Malloc for, 359–374
 - allocator reference in, 359–360

- associative, 104–112
- C++ algorithms in, 90, 94
- concepts for, 88–90
- design methods for, 88–90
- object-based, 89
- position-independent pointers and, 360
- subtype in, 88
- template-based, 88–89
- type in, 88
- typeless, error protection in, 88
- context object, in thread-specific storage, 310
- context switch
 - in half-sync/half-async thread pool, 327
 - in leader/follower thread pool, 338
- control block, position-independent, 357
- cooperative cancellation, 285–286
- copy() , for message blocks, 262
- copy constructor, for hash maps, 109
- counter, for buffer, 294
- critical sections
 - cancellation while executing, 285
 - guarding, from signal interrupts, 245–247
- C++ standard, compilers in, 11
- D**
- DataElement , 90–91
- data elements
 - in bounded stack, 95
 - deletion of
 - in map, 107
 - in self-adjusting binary trees, 112–114
 - unbind() method for, 114
 - in fixed stack, 97
 - insertions of
 - in maps, 104
 - in self-adjusting binary trees, 111, 112–114
 - in stack container, 98
 - iteration of, in self-adjusting binary trees, 112–114
 - locating
 - in map, 105
 - in self-adjusting binary trees, 112–114
 - number of active, 94
 - pointers to (*See* pointers)
- datagrams, in UDP, vs. streams, 208
- data order, in UDP, 208
- data population, in containers, 93
- data type
 - in compilers, 14, 15*t*
 - in porting, 9
- deactivate() , for worker thread pool shutdown, 332
- deadlock
 - detection of, 299–301
 - on mutex, 254–257
 - from mutex acquisition, 291
 - prevention of, 301
- debug statements
 - enabling and disabling, 37–38
 - usefulness of, 37
- deferred cancelability, 285
- deletion()
 - for timer queue, 451
 - for upcall handler, 453
- design patterns, 5, 7. *See also specific patterns*
- desired_threads() , 387
- destroyList() , 94
- developer forums, 22
- directory tree, of ACE distribution, 26–27
- disablecancel() , 285
- disable_debug_messages() , 47
- displayList() , 94
- Distributed Object Computing group, 4
- distribution, structure of, 26–27
- DllMain() function, 18
- DLLs, symbols in, 33–34
- docs directory, 27
- document type definition (DTD), for configuration files, 433
- DONT_CALL mask type, 155, 184
- Double-Checked Locking Optimization pattern, 16
- doubly linked list, 90–94
 - copying, 93–94
 - data population in, 93
 - element type in, 90–91
 - testing, 92–93
 - type definition for, 91–92
- downstream tasks
 - in command stream, 408
 - definition of, 378
 - message queue of, put() method in, 386
 - in module, 383
 - for PlayMessage, 412–413
 - for RetrieveCallerID, 412
- doWork() , for slave process, 224
- Doxygen, for reference documentation, 21

- DTD (document type definition), for configuration files, 433
- `duplicate()`, for message blocks, 263
- dynamic memory allocation
- from runtime heap, 18–19
 - for service handler, 155, 182
- dynamic stack, definition of, 94
- E**
- `elect_new_leader()`, 340
- `empty_set()` routine, 158
- `enable_debug_messages()`, 47
- EncodeMessage task, 395
- `end()`, for maps, 106
- EndTask, for special conditions, 391
- environment variable, in parent process, 223
- `equal()`, specializing, 107–108
- equality operator
- in hash map, 109, 110–111
 - in map manager, 105
 - in sets, 101
- `errno`, global, in thread-specific storage, 309
- error checking, on `acquire()` and `release()`, 256
- error handling
- acceptor and, 135
 - in `handle_input()`, 152–153
- error protection, in typeless containers, 88
- event demultiplexer, 142. *See also* `poll()`; `select()`; `WaitForMultipleObjects()`
- event handle(s), 144, 204. *See also* `handle(s)`; `I/O handle`
- event handler. *See also* `ACE_Event_Handler`; `completion handler`
- `ACE_SOCKET_Acceptor` wrapped in, 143
 - in `ACE_TP_Reactor` implementation, 343
 - in `ACE_WFMO_Reactor` implementation, 345
 - for connection accepting (*See* `connection-accepting handler`)
 - for connection processing, 143
 - for connection requests, 143
 - for connection servicing (*See* `service handler`)
 - dynamically allocated, 184
 - for I/O (*See* `I/O event handlers`)
 - notifications for, 162–163
 - reactor pointer in, 146
 - removal of, 184
 - for signals (*See* `signal event handler`)
 - state data passed to, 163–166
 - for timers (*See* `timer event handler`)
 - XML, 433
- event handling
- demultiplexer for, 142
 - in process management, 229
 - Reactor framework for, implementing, 141
- Event Log
- in mixed environment, 65
 - output to, 58
- event loop
- `ACE_Message_Queue` in, 179
 - in `ACE_TP_Reactor` implementation, 185
 - active-timer dispatcher and, 449
 - function of, 141
 - proactor-based
 - integrating with reactor, 204
 - for I/O completion processing, 201
 - reactor-based, 146
 - integrating with proactor, 204
 - stopping, 156–157
- event notifications, intertask communication on, 257
- examples directory, 27
- exception-handling, for pool growth handling, 369–374
- `execute()`, for command stream, 400–401, 416
- execution context, in `ucontext_t`, 245
- exit functions
- cancellation and, 284
 - Object Manager and, 17
 - for threads
 - number of, 277
 - registration of, 277–278
- `expire()`
- for timer dispatcher, 443
 - for timer queue, 451, 453
- F**
- façade pattern, 9
- factory classes, in Proactor framework, 202–203. *See also* *specific classes*
- failure code, from `process()`, in command stream, 406, 407–408
- failure status, fetching, 467
- FastLight reactor, 186
- `fetch()`

- for NODE_LOCAL context, 473
 - for PROC_LOCAL context, 462, 466, 467
- FIFO scheduling policy, 271
- FIFO sequences, 98, 213–214
- FIFO thread order, tokens for, 297
- file(s), direct operations on, in shared memory, 349, 375–376
- file(s), direct operations on, in shared memory, 349, 375–376
- file I/O, for intrahost communication, 213, 214
- fill_set() routine, 158
- find() function
 - allocator argument in, 369
 - in shared memory, 363
- fini() function
 - Object Manager initialization with, 18
 - for service removal, 431
 - for static services, 421–424
- fixed stack, 95–96, 97
- Fix Kits, 26
- follower thread. *See* leader/follower thread pool
- fork() vs. spawn(), 220
- format, for logging, 38, 41*t*–42*t*
- forums, developer, 22
- framework(s). *See also specific frameworks*
 - in ACE, 5
 - class libraries extended with, 7–8
 - definition of, 6
 - reuse by, 7
 - at runtime, 8
 - vs. patterns, 7
- framework layer, 6
- function tracing, macros for, 53–55
- Future
 - in Active Object pattern, 316, 322–323
 - in half-sync/half-async thread pool, 333
- Future Observer, 323–324
 - in Active Object pattern, 316
 - in half-sync/half-async thread pool, 333
- G**
- get()
 - for Future object, 323
 - for return data, 401
 - for stream tasks, 378
- get_handle()
 - ACE_Svc_Handler and, 172
 - handle access through, 144–145, 150
- get_message_destination(), 395
- getopt(), vs. ACE_Get_Opt, 78–82
- get_process_attribute(), for process ID, 226
- getq()
 - block with, 265
 - for stream tasks, 378, 389–390
- global errno, in thread-specific storage, 309
- GNU Autotools, for build configuration, 28
- GNU Make tool
 - for application building, 31–33
 - for compiling, 9
 - options for, 30*t*
- Graph, 474
- Graphable_Element, 473–475
- Graphable_Element_List, 474, 475–486
- group ID, for thread pool, 275
- grp_id() accessor, 275
- guards
 - for critical sections, 246–247
 - for mutexes, 254–257
 - classes of, 256, 256*t*
 - macros for allocation of, 256
 - vs. acquire and release, 255
- GUI integrated reactors, 185–186
- H**
- half-sync/half-async thread pool, 326–338
 - ACE_Task queuing in, 330
 - ACE_Task queuing in, 330
 - advantages and disadvantages of, 326–327
 - structure of, 326
- handle(), 192, 193
- handle(s). *See also* event handle(s); I/O handle
 - from ACE_Asynch_Acceptor, 199
 - in ACE_WFMO_Reactor implementation, 184
 - in ACE_Win32_Proactor implementation, 201
 - direct use of, 140, 144
 - obtaining, 192, 199
 - for Proactor factory classes, 202–203
 - saving, 192
 - signalable, 204
 - for slave process, 223
 - in Sockets API, 124
 - stored in handler, 191–192
 - value of, getting, 145, 150
- handleCancel(), for upcall handler, 453, 454

- handle_close()
 - in Acceptor-Connector framework, 175–176, 176f
 - in ACE_WFMO_Reactor implementation, 184
 - calling, 148–149
 - handle access through, 144
 - return value from, 154–155
 - handleClose(), for upcall handler, 453, 454
 - handleEvent(), for timer expiration, 453
 - handle_exception()
 - in ACE_WFMO_Reactor implementation, 184
 - control dispatched to, 159–160
 - handle_exit(), 230–231
 - handle_input()
 - in ACE_WFMO_Reactor implementation, 184
 - for Client handler, 179–180
 - error handling in, 152–153
 - handle access through, 144, 146–147
 - return value from, 148
 - for service handler, 150–152
 - in Acceptor-Connector framework, 173–175
 - handle_output()
 - in Acceptor-Connector framework, 175, 180–181
 - in Reactor framework, 153–155
 - handler. *See* event handler
 - handle_read_stream(), 194, 195
 - handler threads
 - barrier and, 307–308
 - updating by, 253–254
 - handle_signal(), 156–157, 239–240
 - in ACE_WFMO_Reactor implementation, 184
 - parameters of (*See* siginfo_t; ucontext_t)
 - signal state and, 158–159
 - for thread signaling, 279
 - handle_timeout()
 - for active-timer dispatcher, 448
 - for Client handler, 180
 - current time and, 162
 - for signal timer, 450
 - state data passed to, 163–166
 - for timer dispatcher, 443
 - for timer event handler, 444–445
 - for timer event listener, 441
 - handle_write_stream(), 195
 - HA_Proactive_Acceptor, 198–200
 - HA_Proactive_Service handler
 - deletion of, 194
 - handle passed to, 191
 - hashing function, 109, 110–111
 - hash map(s)
 - about, 109–111
 - allocator reference in, 366
 - record deletion from, 366–367
 - in shared memory, 361–369
 - head module
 - in command stream, 399
 - in one-way stream, 381
 - heap memory
 - allocation of
 - in compilers, 18–19
 - configuration information and, 83
 - queue on, 99
 - helper class, for self-adjusting binary trees, 111–112
 - Hollywood Principle, 8
 - host name, in ACE_INET_Addr, 126
- I**
- IBM mainframes, asynchronous I/O in, 188
 - implementations
 - of ACE_Proactor, 201–202
 - of ACE_Reactor, 182–186
 - of command stream, 409–414
 - reuse of, 7
 - include/makeinclude directory, 27
 - info(), for static services, 421
 - inheritance, in callbacks, 61
 - init()
 - Object Manager initialization with, 18
 - for static services, 421–423
 - initialization
 - of ACE_Acceptor, 423
 - of command stream, 415
 - of name options, 460
 - of Object Manager, 17–18
 - of reader object, in Proactor framework, 193
 - at runtime, platform and, 14–18
 - of semaphores, 307
 - of static service, 423, 425
 - of writer object, in Proactor framework, 193
 - input, handling, 149–153
 - instance(), for reactor instance, 146
 - instantiation
 - allocators passed during, 117
 - of Object Manager, 18

- Institute for Software Integrated Systems (ISIS), 4
- interface, iterators and, 95
- interprocess communication (IPC)
 - interhost, 207–213
 - intrahost, 213–214
 - shared memory for, 349
 - in wrapper facades, 123
- interval timer
 - resetting, 441
 - timer queue and, 438
- `int_value()`, for `PROC_LOCAL` context, 464, 467
- inversion of control, in runtime architecture, 8
- I/O
 - completion of, 194–197
 - initiation of, in Proactor framework, 193–194
- I/O event handlers. *See also* completion handler; connection-accepting handler; service handler
 - registration of, 144
 - for multiple handles, 147
 - at reactor shutdown, 155
 - removing from reactor, 148–149, 155
- I/O handle, association of
 - in `ACE_Event_Handler`, 144
 - removing from reactor, 148–149, 155
- I/O operations, asynchronous, 196*f*
 - completing, 194–195
 - guidelines for, 195–197
 - initiating, 191–194
 - outstanding, 197
- I/O sources. *See also* client; connection(s); input; output; server
 - handling multiple, 142–155
- iostream formatting, 38–39
- `iovec` structures, 132–133
 - receiving data with, 134–135
 - sending data with, 133–134
- ISIS (Institute for Software Integrated Systems), 4
- `is_member()` routine, 158
- iterator(s)
 - about, 89–90
 - in `ACE_Malloc` map interface, 351
 - in array, 101
 - in C++ algorithms, 90
 - dereferencing, 106, 109
 - in doubly linked list, 94–95
 - in lazy map managers, 104
 - in maps, 105–107
 - in self-adjusting binary trees, 112–114
- iterator APIs, in ACE, 89–90
- K**
- kernel-level threads, 268–269
- key(s)
 - `ACE_Less_Than` functor specialization for, 114–115
 - in associative containers, 103
 - comparability of, in map manager, 104, 107
 - grouping, 457
 - hashing function for, 110
 - in hash maps, 108–109, 363
 - in maps, 103
- key/value pairs, in naming context, 457
- L**
- layered architecture, of ACE toolkit, 6–7
- leader/follower thread pool, 338–343
 - in `ACE_TP_Reactor`, 343–345
 - advantages and disadvantages of, 338
 - becoming leader in, 340, 342
 - follower created in, 341
 - `svc()` for, 339–340, 342
- less-than operator, in `NODE_LOCAL` context, 475
- libraries
 - flexibility from, 5
 - linking, in application build, 32
 - shared (*See* shared libraries)
- licensing, for ACE, 4
- LIFO sequences, stacks as, 94
- linked list. *See also* doubly linked list
 - `ACE_Stream` as, 385
- `list_name_entries()`, for `NODE_LOCAL` context, 473
- lock(s)
 - in `ACE_Malloc`, 350
 - in `ACE_Svc_Handler`, 172
 - in guard classes, 255
 - in hash map, 109
 - in map manager, 108
 - readers/writer, 292–293
- log files, rotation of, 75
- logger key, definition of, 66
- logging. *See also* `ACE_DEBUG` macro; `ACE_ERROR` macro
 - basic, 38–42
 - format for, 38–39, 45

- macros for, 43*t*–44*t*
 - customizing, 47–55
 - output of
 - to output streams, 58–59
 - redirecting, 55–60
 - to standard error stream, 55–56
 - to system logger, 56–58
 - runtime configuration of, 73–75
 - switching, with signals, 157–158
 - thread-specific storage and, 309–310
 - logging client proxy, 65–70
 - configuration files for, 66
 - port value for, 66–67
 - logging server, 65–70
 - configuration file for, 66
 - direct communication with, 67–68
 - starting, 66
 - logging strategy
 - configuration options for, 75*t*
 - definition of, 66
 - for runtime configuration, 73–75
 - LogManager, 70–73
 - long command line options
 - alternative specification for, 81
 - definition of, 78
 - long_only parameter for, 81
 - without short options, 79–80
- M**
- macro(s)
 - for function tracing, 53–55
 - for logging, 43*t*–44*t*
 - customizing, 47–55
 - for memory allocation, 19, 20*t*
 - for mutex allocation, 256
 - for service configuration, 424–425
 - for thread priority, 272*t*
 - macro files, for ACE build, 29*t*
 - main()
 - dynamic service configuration and, 429
 - exit handler in, 278
 - for NODE_LOCAL context, 469, 471–472
 - Object Manager instantiation with, 18
 - for one-way stream, 379–380
 - process thread in, 250
 - for PROC_LOCAL context, 465
 - signal set in, 245
 - vs. exit() function, 17
 - main thread, in process, 250
 - Makefile, for application building, 31–33
 - make_handler(), for
 - ACE_Asynch_Acceptor, 199
 - map(s), 104–108
 - bindings in, 105
 - deletion from, 107
 - insertions in, 104
 - iterators in, 106–107
 - lazy, 104
 - locks in, 108
 - operations on, 105
 - retrievals from, 105
 - map interface, for ACE_Malloc, 351–352, 355
 - MapViewOfFileEx(), 375
 - master process
 - dump by, 224–225
 - environment variable in, 223
 - mutex shared with, 231–232
 - slave result and, 224
 - MB_HANGUP message type, 263, 264, 266
 - checking for, 390, 406–407
 - in stream task close, 389
 - memory, shared. *See* shared memory
 - memory allocation
 - for ACE_WString pointer, 464
 - configuration information and, 83
 - pool growth and, 369
 - position-independent, 356–359
 - for record additions, 355
 - for record deletions, 362
 - resolve() method and, 463
 - in timer queue, 439
 - memory allocation macros, 19, 20*t*
 - memory allocators. *See* allocators
 - memory-mapped files, 119, 458
 - memory ordering properties, 293
 - memory pool
 - in ACE_Malloc, 350
 - growth of, handling, 369–374
 - insert values in, 351
 - shared (*See* shared memory pool)
 - types of, 351*t*
 - memory protection interface, for ACE_Malloc, 352
 - message blocks, 262–263. *See also*
 - ACE_Message_Block
 - in command stream, 401
 - downstream tasks and, 408

- releasing, 409
 - read pointer from, 390
 - sending, in one-way stream, 391
 - `svc()` method for, in stream tasks, 389
 - type field in, 263
 - message passing, 257, 260–266
 - message processing, 273, 275
 - message queue
 - for message passing, 260–266, 266*t*
 - in one-way stream, 385, 387
 - priority in, 266
 - for thread pool, 275
 - in thread pool asynchronous layer, 329
 - types of, 266, 266*t*
 - using, 263–266
 - metadata, saving, 396
 - method. *See specific methods*
 - method, private, for processing code, 203–204
 - method request
 - in Active Object pattern, 315–316
 - creation of, 319–320
 - enqueueing, 321, 338
 - enqueuing, 321, 338
 - in half-sync/half-asynch thread pool, 333, 335
 - microarchitecture, reuse of, 7
 - Microsoft Visual C++, build configuration in, 34–35, 36*t*
 - middleware, flexibility from, 5
 - (minus), in command line options, 82
 - `mmap()`, 375
 - mnemonic, for DLL porting, 34
 - `module()`, 410, 412
 - module(s), in Streams framework
 - in `ACE_Stream`, 383
 - for command stream, 399–400
 - instantiation of, 383
 - in one-way stream, 381
 - in `open()`, 382–383
 - ordering on stream, 400
 - overview of, 377–378
 - pushing onto one-way stream, 383–384
 - tasks in, 383
 - `monitor()`
 - for `NET_LOCAL` context, 477
 - for `NODE_LOCAL` context, 470, 472
 - for `PROC_LOCAL` context, 460, 466
 - `msg_queue()`
 - `ACE_Message_Queue` accessed with, 175
 - queue type specified with, 266
 - `msg_type()`, 263
 - multicast connection, in UDP, 208, 212–213
 - multicast groups, 208, 212–213
 - multiple threads
 - in `ACE_POSIX_Proactor` implementation, 202
 - in `ACE_WFMO_Reactor` implementation, 184
 - handlers in, registering and unregistering, 159
 - multithreaded I/O model, 187, 188
 - multithreaded programming, 249, 282–283
 - multithreaded server, 325
 - mutex
 - acquiring, 252, 255
 - automatic, 258
 - in hash map, 365
 - twice, 291
 - binary semaphore vs., 303
 - condition variable in, 257–258
 - deadlock on, 254–257
 - named, `ACE_Process_Mutex` for, 231–232
 - recursive, 291–292
 - releasing, 252–254, 255
 - automatic, 258
 - in thread synchronization, 233
 - shared, 231–232
 - for thread safety, 252–254
 - type of, in `ACE_Condition`, 259
 - `mutex()`, lock reference obtained with, 108
- ## N
- `Name_Binding`
 - memory management for, 464
 - releasing, 464–465
 - `resolve()` result in, 463, 466
 - values extracted from, 474–475
 - named mutex, `ACE_Process_Mutex` for, 231–232
 - name options, 460
 - namespace, vs. class, 10
 - naming context. *See also*
 - `ACE_Naming_Context`
 - binding in, 460–465
 - key/value pairs in, 457
 - shared, 469–476
 - reading data from, 471–476
 - saving data from, 469–471
 - types in, 457
 - types of, 458

- uses of, 457
- values stored in, 459–468
- Naming_Context, 465
- Naming Service
 - about, 457
 - API of, 459
 - context types of, 458
 - starting, 476–477
- narrow character sets, vs. wide, 19–21
- nested type definition, 106
- NET_LOCAL context, 458–459, 476–478
- netsvcs logging framework, 65–70
- networked applications, difficulty in writing, 5
- networked services layer, about, 6
- network software, timers in, 437
- next_step(), for stream tasks, 391
- NODE_LOCAL context, 469–476
 - access in, 458
 - modifying, for NET_LOCAL context, 477
- nonblocking connection operation, with
 - ACE_SOCKET_Connector, 132
- notification
 - for callback queuing, 159
 - for callback queuing, 159
 - control returned by, 238, 247
 - in process event handling, 230
- notification strategy, on ACE_Message_Queue, 179
- notify(), logging switching with, 159
- NotifySomeone task, 396–397
- Null Mutex, 16, 111
- O**
- object, runtime initialization of, 14–18
- object-based containers, 89
- Object Manager
 - about, 14–15
 - initialization of, 17–18
 - instantiation of, 18
 - rules for, 17
 - termination of, 17–18
- object type, in containers, 88
- one-way stream, 378–397
 - initializing, 381–386
 - main program for, 379–380
 - modules in, 383
 - stream in, 381–387
 - tasks in, 386–397
- open()
 - for ACE_Asynch_Acceptor, 200
 - for ACE_SOCKET_Acceptor, 135
 - for Client handler, 168–179
 - for ClientService handler, in Acceptor-Connector framework, 172–173
 - for command stream, 399
 - for command task, 405
 - for connection-accepting handler, 144–145, 146
 - for HA_Proactive_Service, 192
 - for logging service, 67
 - for one-way stream, 380, 382–383
 - for PROC_LOCAL context, 460
 - service configuration with, 426
 - for service handler, 148–149
 - for stream tasks, 387
 - for system logger output, 56–58
- Open VMS, asynchronous I/O in, 187
- operating system, OS adaptation layer in, 10
- operating systems
 - multiple, porting code to, 8–10
 - priorities defined in, 271
 - system loggers of, 58
- operator(), for command line options, 79
- operator->(), for thread-specific storage access, 310
- op_status() methods, 47
- ordering parameter, for argument ordering, 82
- ordering properties, of memory, 293
- OS adaptation layer, 6, 9, 10
- OS methods, 9–10
- output, handling, 154–155
- output streams
 - deleting, 60
 - for logging, 58–59
 - thread-specific, 309–310
- owner(), for ACE_Select_Reactor, 183
- P**
- parallel processing, in Streams framework, 378
- parent(pid_t child), 226
- parent process. *See* master process
- parsing
 - at arbitrary index, 80–81
 - error reporting during, 81
- pass_addresses argument, 200
- peer()
 - ACE_SOCKET_Stream accessed with, 175

- ACE_Svc_Handler and, 172
 - for ClientService, 147
 - for command module, 403, 409, 410
- peer, in UDP unicast, 209, 210
- perform(), method request enqueued by, 338
- performance, in multithreaded I/O, 188
- PERMUTE_ARGS, for argument ordering, 82
- pipes, for intrahost communication, 213–214
- Pipes and Filters pattern, in Streams framework, 377
- platform. *See* operating systems
- PlayMessage, 412–413
- play_message(), 417
- PlayOutgoingMessage task, 392–393
- + (plus), in command line options, 82
- pointers. *See also specific pointers*
 - in ACE_Message_Block, 154
 - in ClientService::open, 173
 - copying, in queues, 100
 - data population with, 93
 - in fixed stack, 97
 - iterators and, 89
 - position-independent, 358–359, 360
 - in queue, 99–100
 - to reactor, in ACE_Acceptor, 173
 - in reference containers, 93
 - in sets, 101, 103
 - to shared memory, 354, 356–357
 - typeless, in C, 88
- poll(), for event handling, 142
- port
 - in ACE_INET_Addr, 126, 145
 - choosing, 131
 - for client, 131
 - for server, 135, 145
 - in UDP, 207, 209, 211
- portability, standards and, 9
- position-independent allocation, 356–359, 361
- position-independent control block, 357
- position-independent pointers, 358–359, 360
- POSIXLY_CORRECT environment variable, 82
- POSIX systems
 - asynchronous I/O in, 188
 - proactor implementation on, 202
 - signals on, response to, 156
- prepare(), for process spawning, 222–223
- primitives
 - for consistency, 289
 - for thread safety, 251
 - types of, 290*t*
- priorities, thread scheduling classes and, 271–274, 272*t*
- priority(), in ACE_Priority_Reactor implementation, 185
- private method, for code, 203–204
- private thread of control. *See* thread of control
- proactive I/O model. *See* asynchronous I/O model
- proactor argument, 200
- proactor event loop
 - integrating with reactor, 204
 - for I/O completion processing, 201
- Proactor framework
 - classes in, 189–191, 190*f*
 - completion handling in, 201–202
 - connection establishment in, 198–201
 - I/O operations in, 191–197
 - Reactor framework and, combining, 203–205
 - UDP and, 208
- PROBLEM-REPORT-FORM file, 26
- process. *See also* ACE_Process; master process; slave process
 - address space protection between, 349
 - event handling with, demultiplexer for, 142
 - logging severities of, 45–47
 - main thread in, 250
 - in shared memory, 365
 - signaling, in multithreaded programs, 284–285
 - spawning, 219–226
 - spawning multiple, 226–231
 - synchronization of, mutex for, 252
 - terminating, 227, 229, 243–245
- process()
 - for command task, 406, 407–408, 410
 - for stream tasks, 387, 390–391
- process_directive(), for service configuration, 434
- process_file(), for service configuration, 434
- process ID
 - in signal handling, displaying, 242
 - for slave process, in process termination, 228
- process_message(), priorities in, 273
- process-per-connection model, 142
- processRecord(), 372–373
- processWin32Record(), 372–373
- PROC_LOCAL context, 459–468

- access in, 458
 - binding in, 460–465
 - `protect()`, for `ACE_Malloc`, 352
 - protection mode, for memory-mapped files, 376
 - protocols, for multicast group management, 212
 - Proxy
 - in Active Object pattern, 315–316
 - in method request creation, 319, 322
 - proxy, as Active Object, 313
 - Proxy pattern, in Active Object pattern, 315
 - `put()`
 - for command task, 405
 - for downstream task message queue, 386, 401
 - for stream tasks, 378, 385, 388
 - `put_next()`, for stream tasks, 378
 - bidirectional, 407
 - one-way, 391
 - `putq()`
 - for message enqueueing, 264
 - for message enqueueing, 264
 - for stream tasks, 388, 411
- Q**
- Qt reactor, 186
 - quality-of-service parameters, with
 - `ACE_SOCK_Connector`, 132
 - queue(s), 98–100. *See also*
 - `ACE_Message_Queue`
 - for `handle_input()` errors, 152–153
 - for message passing (*See* message queue)
 - `putq()` method for, 175
 - shared memory allocator specified for, 371
 - as shared resource, 260
 - queueing layer, of half-sync/half-async thread pool, 326–327
 - queuing layer, of half-sync/half-async thread pool, 326–327
- R**
- `rd_ptr()`, 154, 262
 - reactive I/O model, 142, 187–189
 - reactor. *See also* `ACE_Reactor`
 - in `ACE_Acceptor`, 171
 - handlers registered with, for state data, 166
 - I/O event handlers registered with, 144
 - removing, 148–149, 155
 - at shutdown, 155
 - notifications in, 159–160
 - control returned by, 238, 247
 - shared memory registered with, 214
 - signal handler registered with, 238
 - signal management with, 247
 - signal registered with, 157–158
 - timers handled by, 162–163
 - `reactor()`, for `ACE_Event_Handler`, 146
 - Reactor event, handlers for, from
 - `ACE_Event_Handler`, 144
 - reactor event loop, 146
 - service reconfiguration and, 431
 - stopping, 156–157
 - Reactor framework
 - `ACE_Message_Block` in, data handling with, 203–204
 - callbacks in, 141, 148
 - overview of, 142
 - Proactor framework and, combining, 203–205
 - process management and, 229
 - purpose of, 141
 - server based on, 145–149
 - signal-handling in, 235, 247
 - UDP classes in, 208
 - reactor pointer
 - in `ACE_Acceptor`, 173
 - in `ACE_Connector`, 179
 - `read()`, for `ACE_Synch_Read_Stream`, 192
 - reader object, initialization of, in Proactor framework, 193
 - readers/writer locks, 292–293
 - reader task, in Streams framework, 377
 - read operations
 - asynchronous, 194–195, 196
 - on memory-mapped files, 376
 - read pointer
 - in asynchronous write operation, 195
 - automatic update of, 196
 - in message block, in one-way stream, 390
 - `readv()`, 132–133
 - `rebind()`, in `PROC_LOCAL` context, 462, 466, 468
 - receive methods, in `ACE_SOCK_Stream`, 127
 - Receiver Implementation, in Active Object pattern, 315–316
 - `reconfigure()`, for services, 432
 - record(s)
 - adding, 355

- binding, 355
- copying to shared memory, 366
- deletion of
 - from hash map, 366–367
 - memory pool growth and, 367
- inserting, into shared memory allocator, 352, 353–354
- memory allocation for, deletion of, 362
- record(), for one-way stream, 384–385, 386
- recorder(), for AnswerIncomingCall, 386–387
- record_failure(), for PROC_LOCAL context, 467
- record_history(), for NODE_LOCAL context, 470
- RecordIncomingMessage task, 393–394
- RecordingDevice
 - for bidirectional stream, 398
 - for one-way stream, 380, 386–387
- RecordingStream, 380, 381–386
- record-keeping information
 - for dynamic services, 428, 429
 - for static services, 424
- RecordMessage, 413–414
- record_message(), 417
- record_temperature(), 466
 - for NODE_LOCAL context, 470
- recursive mutex, 16, 291–292. *See also*
 - ACE_Recursive_Thread_Mutex
- recv()
 - for ACE_SOCKET_Dgram, 210
 - for ACE_SOCKET_Dgram_Mcast, 213
 - return value for, handle_input() method and, 152
- recv_n(), buffer and, 127
- recvv(), buffer allocation with, 134–135
- Red Black Tree, 111
- redirect* method, for output destination selection, 70
- reference containers, pointers in, 93
- reference documentation, for ACE, 21
- register_action(), for callback registration, 238
- register_handler()
 - acceptor events monitored by, 145
 - input events and, 148
 - for signal event handler, 241
 - for signals, 157–158
- release()
 - for message blocks, 262
 - for mutex, 252–254, 258
 - for readers/writer lock, 292
 - semaphores and, 304
 - for TextListener, 418
 - vs. guard, 255
- ReleaseDevice task, 394
- release versions, 25
- remap(), for pool growth handling, 369–374
- remove(), for services, 434
- remove_handler()
 - in ACE_WFMO_Reactor implementation, 184
 - handle_close() and, 148–149
- REQUIRE_ORDER, for argument ordering, 82
- reset_device(), for PROC_LOCAL context, 468
- reset_interval(), for timer dispatcher, 441
- resolve()
 - for NODE_LOCAL context, 473
 - for PROC_LOCAL context, 462–463, 466
- resource sharing, coordination of, 231
- Result
 - for ACE_Asynch_Read_Stream, 195
 - in Proactor framework, about, 191
 - in Proactor framework, about, 191
- resume()
 - for services, 432, 434
 - thread management with, 276
 - for thread schedule, 271
- retrieve_callerID(), 392, 416
- RetrieveCallerID module, 411–412
- RETURN_IN_ORDER, for arguments, 82
- reuse
 - of addresses, 136, 145, 200
 - of code, templates for, 11–12
 - in frameworks, 7
 - in patterns, 7
- reuse_addr flag, 136, 145, 200
- Riverace Corporation, 22, 26
- root section, of configuration data, 84
- round-robin scheduling policy, 271
- runtime
 - behavior at, altering with command lines, 77
 - configuration at, 420
 - debug statements at, 37–38
 - frameworks at, 8

- initialization at, 14–18
- logging configuration at, 73–75
- service configuration at, 419
- S**
- SA_RESTART, 239
- SaveMetaData task, 395–396
- schedule ()
 - for timer dispatcher, 443
 - timer ID returned by, 440
- Scheduler
 - in Active Object pattern, 315–316, 319
 - thread of control in, 320
 - aggregation of, with agent implementation, 321
- schedule_timer (), return value of. *See* timer ID
- schedule_wakeup (), 180
- scheduling, of threads
 - real-time, 271
 - time-shared, 271
 - user-level vs. kernel-level, 268–269
- scheduling classes, priorities and, 271–274, 272*t*
- scheduling state, thread, initial, 270–271
- Schmidt, Douglas C., 3–4
- Secure Sockets Layer (SSL) handshake, 199
- security parameters, in process spawning, 225–226
- SEH (structured exception handling), in pool
 - remapping, 369–370
- select ()
 - for event handling, 142
 - vs. WaitForMultipleObjects () function, 153
- self (), thread ID from, 285
- self-adjusting binary trees, 111–115
- semaphores
 - ACE_Process_Mutex with, 234
 - acquiring, 302, 303
 - conditional variables vs., 303
 - definition of, 302
 - initialization of, 307
 - releasing, 302
 - for thread synchronization, 302–307
- send methods, for ACE SOCK_Stream, 127
- send_n (), buffer and, 127
- sendv (), 133
- sensors, state data from, 163–166
- sequence containers. *See* array; doubly linked list; queue(s); set(s); stack container
- server. *See also* I/O sources
 - communication with, 126
 - connection to (*See* connection(s))
 - constructing, 135–140
 - message to, processing with threads, 250
 - querying, 125, 129, 133–134
 - Reactor-based, 145
 - send and receive in, 137–138
 - socket connection to, 126
- service
 - configuration of
 - methods for, 434
 - reprocessing, 431–432
 - without configuration files, 434
 - XML for, 432–433
 - dynamic configuration of
 - overview of, 420
 - at runtime, 419
 - reconfiguring, during execution, 431–432
 - removal of, 431
 - singletons and, 434–435
 - specifications of, in stream configuration file, 430
- service, dynamic
 - configuration of, 426–430
 - declaration of, 428
 - loading, 429
 - runtime substitution of, 426
 - runtime substitution of, 426
 - writing, 427
- service, static
 - cleanup of, 423–424
 - configuration of, 420–426
 - ignoring, 426
 - initialization of, 423, 425
 - instantiation of, 421
 - in service configurator repository, 424
- Service Configurator framework
 - direct action on, 434
 - for logging, 65–66
 - for logging strategy, 74
 - options in, 426, 427*t*
 - overview of, 420
 - repository in, 424
 - XML event handlers for, 433
- service handler. *See also* ClientService handler
 - in ACE_Acceptor, 170
 - allocation of, 155

- ACE_Svc_Handler and, 182
- creation of, 147
- declaration of, 149–150
- deriving at compile time, 203–204
- handle_input() method for, 150–152
- messages enqueued by, 264
- messages received by, 263
- in Proactor framework, 198
- queueing in, 152–153
- queuing in, 152–153
- registration of, with reactor, 148
- separation of, 144
- set(s), 101–104. *See also* bounded set; unbounded set
 - equality operator in, 101
 - pointers in, 101, 103
 - for signal registering, 158
 - signals in, 238, 245
- set() methods, 129–130
- set_process_attribute(), for process ID, 226
- severities
 - enabling and disabling, 44–47
 - mapping to Event Log severities, 58, 59*t*
 - mapping to Event Log severities, 58, 59*t*
 - parameter for, 38, 39*t*
 - at process level, 45–47
 - in runtime logging configuration, 74–75
 - at thread level, 45–47
- severity mask, 45–47
- shared libraries
 - building, from multiple source files, 32–33
 - naming, 430
 - services loaded from, 426
 - services resident in, 427
- shared memory
 - allocation in, 115
 - hash map in, 362–363
 - for interprocess communication, 349
 - Unbound_Queue for, 370–371
- shared memory allocator
 - creation of, 365
 - instantiation of, 353
 - persistence with, 352–356
 - remapping, 359
- shared memory pool
 - base address for, 356–357, 369
 - growth of, 367, 369–374
 - pointers to, 354, 356–357
- shared memory stream, for intrahost communication, 214
- shared mutex, 231–232
- shared resource
 - coordinating, 232
 - queue as, 260
- sharing mode, for memory-mapped files, 376
- short command line options, 78
- short reads, send_n() and, 127
- short writes, recv_n() and, 127
- shutdown. *See also* MB_HANGUP message type
 - reactor, I/O event handlers at, 155
 - with semaphores, 306
 - worker thread pool at, 332
- shutdown_barrier, 307–308
- si_address, 243
- si_code, 243
- sigaction()
 - for action association, 236
 - signal interruption and, 239
- sig_add() routine, 158
- SIGBUS, si_address and, 243
- sig_del() routine, 158
- SIGFPE, display details of, 243
- SIGHUP, for service reconfiguration, 431
- SIGILL, si_address and, 231
- siginfo_t, 240, 241–245
- SIGINT signal, catching, 155–157
- SIGKILL, masking and, 246
- signal(s), 156–158
 - about, 235
 - accept() method interrupted by, 136–137
 - action associated with, 236, 239
 - asynchronous, in multithreaded programs, 282
 - callback registration for, 236–237
 - display details of, 242
 - interruption by, 239
 - multiple callbacks for, 237, 245
 - in multithreaded programs, 282
 - passed to condition variable, 258
 - on POSIX systems, response to, 156
 - in process event handling, 230
 - in process termination, 228
 - for service reconfiguration, 431
 - synchronous, in multithreaded programs, 282
 - system calls and, 239
 - in threaded applications, 279–283
- signal context, control in, 247

- signal event handler
 - execution of, signal disabled during, 238–239
 - registration of, 241
 - for signals, 239–245
 - multiple, 157–158
 - single, 156–157
- signal handler
 - associating, 235, 279
 - callback for, registration of, 238
 - code for, 237
 - creation of, 240–241
 - in multithreaded programs, 282
 - in pool remapping, 369–370
 - registration of, 238, 281
 - stacking, 245
 - testing, 245
- signal mask, for threads, 279
- signal number
 - character strings mapped to, 242
 - passed to `handle_signal()`, 240
- signal state, control in, 158–160, 246
- signal timer dispatchers, 449–450
- signal type, signal handler for, 279
- SignalHandler
 - timer cancelled with, 168–169
 - timer reset with, 167
- SIGSEGV
 - display details of, 243
 - pool remapping and, 370
 - `si_address` and, 243
- `sigset_t` argument, 236
- SIGSTOP, masking and, 246
- Singleton
 - about, 15–16
 - declaring, 71
 - services and, 434–435
- Singleton method, for cleanup, 16
- Singleton template, in `LogManager`, 71
- slave process
 - code for, 223–224
 - command line arguments for, 223
 - handles for, 223, 224
 - mutex shared with, 231–232
 - options for, 220–222
- sleep, in signal handlers, 237–238
- `sockaddr_in` structure, in Sockets client, 124–125
- `socket()` function, file descriptor from, 125
- socket handle. *See* `handle(s)`
- socket pointer, in command module, 400, 402–403, 409, 410
- Sockets programming
 - client program in, 127–128
 - disadvantages of, 124
- software development, complexity and cost of, 7
- source code, multiplatform, difficulty of writing, 5
- source files, multiple, building from, 32
- `spawn()`, 46
 - `fork()` compared to, 220
 - for multiple processes, 227
 - for processes, 221
 - `system()` compared to, 220
- `spawn_n()`, 46
 - for multiple processes, 221
- special conditions, in one-way stream, 381, 391
- `sprintf()`, for `PROC_LOCAL` context, 462
- SSL (Secure Sockets Layer) handshake, 199
- stack container, 94–97. *See also* bounded stack;
 - fixed stack; unbounded stack
 - dynamic, 94
 - insertions on, 98
 - iterator in, 94
 - static, 94
- stack memory
 - exit handler on, 278
 - guard called on, 255
 - queue on, 98
- standard error stream (STDERR), output to, 55–56
- standards, portability and, 9
- standard template library (STL), support for, 87
- `start()`, for start-up hooks, 283–284
- `startup_barrier`, 307–308
- start-up hooks, for threads, 283–284
- state change, intertask communication on, 257
- state data
 - consistency of, 251
 - consistency of, 251
 - passing, to event handler, 163–166
 - in thread-specific storage, 309
- static service
 - configuration of, 420–426
 - ignoring, 426
 - initialization of, 423, 425
 - instantiation of, 421
 - in service configurator repository, 424
- static stack, definition of, 94

- status server, querying, 133–134
- STDERR (standard error stream), output to, 55–56
- STL (standard template library), support for, 87
- Strategy pattern
 - in `ACE_Reactor_Notification_Strategy`, 178
 - wrapper facades and, 123
- `strdup()`, 464
- stream. *See also* `ACE_SOCKET_Stream`; `ACE_Stream`
 - bidirectional (*See* bidirectional stream; command stream)
 - configuration of
 - from file, 430
 - at runtime, 420
 - with XML, 432–433
 - definition of, 123
 - one-way (*See* one-way stream)
 - in Streams framework, 377, 379f
 - vs. datagram, 208
 - Streams framework, 377–378
 - string, argument vector conversion to, 85
 - structured exception handling (SEH), in pool
 - remapping, 369–370
 - subtypes, in object containers, 89
 - `suspend()`
 - for services, 432, 434
 - thread management with, 276
 - `svc()`
 - for command task, 404, 406–407, 410–411
 - for leader/follower thread pool, 339–340, 342
 - for stream tasks, 389–390
 - thread started in, 250
 - `switch()`, 243
 - symbols, importing and exporting, in DLLs, 33–34
 - `sync()`, for `ACE_Malloc`, 352
 - synchronization classes, 231
 - synchronization complexity, in multithreaded I/O, 188
 - synchronization primitives, for threads, 302f
 - synchronous layer, of half-sync/half-async thread pool, 326–327
 - synchronous signals, handling, in multithreaded programs, 282
 - sync interface, for `ACE_Malloc`, 352
 - system logger, output to, 56–58
 - `system()` vs. `spawn()`, 220
 - System V shared memory, 119
 - System V STREAMS framework, 377

T

 - tail module
 - in command stream, 399
 - in one-way stream, 381
 - tasks, in Streams framework. *See also* downstream tasks; upstream tasks
 - base class for, 387–392
 - for command stream, 399–400, 402
 - message queue of, 385
 - methods in, 387–392
 - in modules, 383
 - in one-way stream, 386–397
 - in `open()`, 382–383
 - overview of, 389–390
 - shutting down, 387, 388
 - threads for, releasing, 394
 - Tcl/Tk reactor, 186
 - TCP connection, vs. UDP, 207–208
 - technical support services, 22
 - temperature graphing application, 471–476, 478
 - `Graphable_Element` in, 473–475
 - `Graphable_Element_List` in, 474, 475–476
 - `Graph` in, 474
 - temperature monitor application, 465–468
 - template(s),
 - compiler application of, 176
 - in compilers, difference among, 11–14
 - instantiation of, in compilers, 11–12, 71
 - template arguments classes, types defined in, 13–14
 - template-based containers, 88–89
 - template specialization
 - about, 89
 - for `ACE_Less_Than` functor, 114–115
 - for hashing function, 110–111
 - for key type comparability, 107–108
 - `terminate()`, for slave processes, 228
 - termination, of Object Manager, 17–18
 - `testcancel()`, 285
 - testing
 - in doubly linked list, 92–93
 - of signal handler, 245
 - tests directory, 27
 - `TextListener`, command stream used by, 414–418

- TextListenerAcceptor, 410
- THANKS file, 27
- THR_BOUND flag, user-level thread bound by, 269
- THR_DETACHED flag, 269
- thread(s). *See also* thread of control
 - in ACE_Select_Reactor, 183
 - cancellation of, 284–288
 - for command task, creation of, 405
 - communication among, 257–266
 - cooperation between, 313
 - creation of, 250, 272
 - data added to, 283
 - detached, 269–270
 - event handling with, demultiplexer for, 142
 - execution of, ordering, 302
 - exit functions for, number of, 277
 - joinable, 269–270
 - kernel-level vs. user-level, 268–269
 - logging severities of, 45–47
 - management of, 276–279
 - multiple
 - in ACE_WFMO_Reactor implementation, 184
 - handlers in, registering and unregistering, 159
 - number of, in barrier, 307
 - owner, recording, 258
 - priority in creation of, 272
 - in proactive I/O, 189
 - readers/writer lock on, 292–293
 - scheduling, 268–269, 271
 - scheduling classes for, 271–274
 - scheduling state of, initial, 270–271
 - shutdown of, barrier and, 307
 - signaling, 279–281
 - signal mask for, 279
 - start-up hooks for, 283–284
 - start-up of, barrier and, 307
 - for stream tasks, 387, 394
 - termination of, with cancellation, 284
 - types of, 267–271
- thread of control, 313, 320
- thread creation flags, for thread attributes, 267–268, 268*t*
- thread_hook(), 284
- thread ID
 - of leader thread, 338
 - mutex and, 291
 - obtaining, 285
 - in thread-specific storage, 310
- threading policy, in multithreaded I/O, 188
- thread manager
 - exit handler registered with, 277
 - multiple, 279
 - pointer to, 277
 - signals sent with, 279
 - as singleton, 279
- thread-per-connection model, 142
 - for multithreaded I/O, 188
 - thread-specific storage with, 310
- thread-per-request model, vs. thread pool model, 326
- thread pool(s), 274–275
 - about, 325–326
 - in ACE_TP_Reactor implementation, 185
- thread pool model, for multithreaded I/O, 188
- thread-pool reactor, implementation of, 183
- thread priority macros, 272*t*
- thread safety
 - basics of, 251–257
 - in map manager, 108
 - mutexes for, 252–254
- thread-specific storage (TSS), 309–311, 327
- thread synchronization, 301–309
 - ACE_Mutex for, 252
 - ACE_Process_Mutex for, 231–234
 - in half-sync/half-async thread pool, 327
 - semaphores for, 302–307
- threat, callback in, 61
- THR_JOINABLE flag, 269
- thr_mgr(), 277
- THR_NEW_LWP flag, user-level thread bound by, 269
- THR_SCHED_FIFO flag, 272
- THR_SCHED_RR flag, 272
- thr_self(), 285
- THR_SUSPENDED flag, 270
- timed block, on condition variable, 258
- timeout
 - with ACE_SOCKET_Stream, 132
 - on connection request, 130–131
 - on connection requests, 136
 - in multiple process management, 229
 - thread for handling, 345–346
 - in timer queue, 438
- timeout(), for upcall handler, 451, 453
- timer(s), 160–169
 - about, 437–438

- block on, 443
 - cancellation of, 168–169, 441
 - expiration of, event handlers for, 439, 453
 - handling, with reactor, 162–163
 - hardware, 437
 - interval
 - resetting, 441
 - timer queue and, 438
 - in Proactor framework, 202
 - process-based, 162–164
 - resetting, 166–167
 - scheduling, 443
 - timer dispatcher, 441*f*, 442*f*
 - parts of, 439–440
 - prebuilt, 440, 447–450
 - timer queue in, 455
 - timer driver, definition of, 440
 - timer event handler
 - for active timer dispatcher, 448
 - for cancellation, 444
 - managing, 450–455
 - registration of, 163
 - specification of, 450
 - for timeout, 444–445
 - for timer expiration, 162, 439
 - timer event listener, 440–441
 - timer ID, 167–169, 440
 - timer queue
 - about, 438
 - characteristics of, 439
 - class hierarchy for, 438*f*
 - memory allocation in, 439
 - template types in, 450, 451*f*
 - in timer dispatcher, 455
 - timer queue event handler, 451–452
 - timer singleton, 441
 - `timerTask()` function, 160–162
 - time-shared schedulers, 271
 - token, 254, 299
 - Token framework, 297–301
 - token manager, 297
 - Trace, 51–55
 - `TRACE_RETURN` macro, 53–55
 - `TRACE_RETURN_VOID` macro, 53–54
 - tracing. *See also* `ACE_TRACE` macro
 - about, 39–42, 44*t*
 - of functions, macros for, 53–55
 - try and back-off strategy, for deadlock prevention, 301
 - TSS (thread-specific storage), 309–311, 327
 - type(s)
 - in naming context, 457
 - synchronization around, 293
 - in template arguments classes, use of, 13–14
 - type awareness, allocators and, 116
 - type definition
 - for doubly linked list, 91–92
 - nested, in maps, 106
 - type information, in configuration information, 85
- U**
- `ucontext_t`, 245–246
 - passed to `handle_signal()`, 240
 - UCS (Universal Multiple Octet Coded Character Set), 19
 - UDP/IP
 - for interhost communication, 207–213
 - vs. TCP, 207–208
 - UDP sockets
 - closing, 210
 - vs. TCP, 126
 - `unbind()`
 - allocator argument in, 369
 - for element deletion, 114
 - variables reset with, 467
 - unbounded set, 100, 102–103
 - unbounded stack, 94–97. *See also* `ACE_Unbounded_Stack`
 - `Unbound_Queue`, for shared memory, 370–371
 - unicast connection, in UDP, 208, 209–211
 - Unicode, 19
 - Universal Multiple Octet Coded Character Set (UCS), 19
 - UNIX/Linux syslog
 - in mixed environment, 65
 - output to, 58
 - upcall handler, 451–455, 452*f*
 - upcall manager, 451
 - `update_device()`
 - with multiple threads, 298
 - mutex acquisition in, 252–254
 - `update_graph()`, for `NODE_LOCAL` context, 472–473
 - upstream tasks
 - in command stream, 408
 - definition of, 378

- putq() method on, 411
 - for RecordMessage, 411–414
 - for RetrieveCallerID, 411–412
- user ID
 - in signal handling, 242
 - for slave process, setting, 223, 225–226
- user-level threads
 - binding, 269
 - vs. kernel-level threads, 268–269
- V**
- __VA_ARGS__, 49, 50–51
- validate_connection(), for
 - ACE_Asynch_Acceptor, 198–199
- validate_new_connection argument, 200
- value()
 - for ACE_Atomic_Op, 294
 - for name binding, 466
- value containers
 - bounded stack as, 95
 - vs. reference containers, 93
- VERSION file, 26
- Visual C++. *See* Microsoft Visual C++
- W**
- wait()
 - on barrier, 307
 - on condition variable, 258
 - on follower thread, 341
 - for process termination, 228
 - slave process and, 221–222
 - for thread completion, 251
 - for thread joining, 269–270
 - thread management with, 276
 - with thread manager, 278
 - on timer, 443
- wait_for_activity()
 - for RecordingDevice, 380
 - for TextListenerAcceptor, 414–415
- wait_for_event(), for timer dispatcher, 441
- WaitForMultipleObjects()
 - in ACE_WFMO_Reactor implementation, 183
 - for event handling, 142
 - select() function vs., 153
- while loop, client connections and, 138–139
- wide character sets
 - macros for, 20*r*
 - vs. narrow, 19–21
- Windows
 - asynchronous I/O in, 188
 - Event Log of, 58, 65
 - proactor implementation on, 201–202, 204–205
 - reactor implementation on, 183
 - registry of, configuration information in, 83, 84–85
 - service DLL on, 430
 - signals in, 235
 - Sockets portability to, 125
- WinMain() function, 18
- worker thread pool
 - in ACE_Unbounded_Queue, 329
 - at shutdown, 332
- wrapper
 - for shared memory primitives, 375–376
 - for signal handling, 236–239
- wrapper facade layer
 - about, 6
 - interprocess communication (IPC) in, 123
- wrapper facade patterns, in OS adaptation layer, 9
- write(), noncontiguous buffers and, 133
- write operations
 - asynchronous, 195, 196
 - on memory-mapped files, 376
- write pointer
 - in asynchronous read operations, 194
 - automatic update of, 196
- writer object, initialization of, in Proactor framework, 193
- writer task, in Streams framework, 389
- writenv(), 132–134
- wr_ptr(), 262, 264
- X**
- X Toolkit reactor, 186
- X Windows, reactor extensions for, 185–186