Index

abandoned Mutex, 124
accept
    alternative, 273
    statement, 269
accept() (Java), 315
acquire() (Java), 190
acquireUninterruptibly() (Java), 190
Active object, 320
Ada language, 273
Adaptive tracing, 411. *See also* Tracing, testing, and replay
all-du-paths, 392. *See also* Path, path-based coverage criteria
Alternating bit protocol, 348
Amdahl's law, 44
Asynchronous message passing, 259
at-most-once property, 28, 44. *See also* Atomic actions
AtomicBoolean (Java), 82
Atomic
    actions, 25, 43. *See also* at-most-once property
    operations, 24, 95
auto_ptr<> (C++), 18, 266
Auto-reset Event, 132
Bakery algorithm, 56 - 58
Barging, 113, 202
Barrier, 170
Bathroom problem, Unisex, 169,248,305
Bear-and-the-honeybees problem, 171, 254, 306
_beginhthreadex() (Win32), 6, 16
Binary semaphore, 90
    implementation, 93
    simulating, 183
binarySemaphore (Java), 192, 195
Black-box testing, 34, 408
Blocking receive, 259
Blocking send, 259
boundedBuffer (class)
    C++/Pthreads, 197
    C++/Win32, 135
    Java, 117, 193, 195, 236
    message-passing, 276
    monitor, 179, 181
Bounded buffer problem, 25, 96
    programmed using
        message-passing, 275, 328
        monitors, 181
        semaphores, 97
    reachability testing, and, 159, 235, 299
Bounded waiting requirement, 47
Breakpoint, 242
Buffer-blocking send, 259
Busy-waiting, 60-61
C run-time library, 6
C#, 42
Cache, 61
    consistency problem, 74
    multiprocessor, in a, 75
call(), 269. *See also* Rendezvous
CARC-sequence, 353-362. *See also* SYN-sequence
Cascaded wakeup, 104, 202
Causal ordering for tracing, 338
Causal synchronization, 298
Causality, 332
Channel, for message passing, 258
channel (Java class), 260
choose(), 274
Circular wait, 407
Client and server, 266
clone(), 266
Closed monitor call, 213
CloseHandle() (Win32), 9, 123, 125
Coincidental correctness, 398
Communication channel, *see* Channel
Communication event, 224
Communication-sequence, 233, 239. *See also* SYN-sequence
Concurrency graph, 393
    coverage, based on
        all-concurrency-paths, 394
        all-concurrency-states, 394
        all-edges-between-concurrency-states, 394
        all-possible-rendezvous, 395
        all-proper-concurrency-paths, 394
Concurrent Pascal language, 187
Concurrent program, definition of, 1
Concurrent reading exclusive writing (CREW) 70
Condition (Java class), 196
    operations on
        await(), 196
        signal(), 196
Condition synchronization, 38, 178
Condition variable, 178
    Java, in, 194, 196
    operations on, 178 - 180
    Pthreads, in, 196
    signaling disciplines, 199
conditionVariable (class)
    (C++), 214-215
    (Java), 207-208
Consumer thread, s*ee* Bounded buffer problem
Context switch, 3
Controlling executions
    distributed programs, 369
    message-passing programs, 290-294

monitor-based programs, 230-233
semaphore- and lock-based programs, 152-153
Correctness
checking, 74
definition, of, 395
Counting semaphore, 84
implementation, 93, 95
simulating, 182, 281
countingSemaphore
C++ class, 130
Java class, 111-113, 191
Coverage-based reachability testing, 447
Coverage criteria, *see* Path, path-based coverage criteria;
Concurrency graph, coverage, based on
Coverage-preserving reductions, 447
CREATE_SUSPENDED (Win32), 7
CreateMutex() (Win32), 122
CreateSemaphore() (Win32), 124
CreateThread() (Win32), 6
Critical section of code, 46
Critical section problem, 46
bakery algorithm, 56-58
busy-waiting solution, 60-61
Peterson's algorithm, 52-53, 72
semaphore solution, 88
ticket algorithm, 54
CRITICAL_SECTION (Win32), 119-120
CSC-sequence, 357-362. *See also* SYN-sequence
Data race, 38, 187
detection, 144
Deadlock, 62. *See also* Deadlock detection
avoiding during VP(), 129
avoiding using locks, 91
detection algorithm, 402
dining philosophers program, in, 98
formal definition, 401
Deadlock avoidance, 154
Deadlock detection, 154
instructions for using, 161, 238, 300, 301
Java Hotspot VM, in, 156
Deadlock prevention, 154
Deadness error, 400
Debuggers, 242-243
Debugging, 30, 382. *See also* Tracing, testing, and replay
definition-use (du) pairs, 392, 414. *See also* Path, bath-based coverage criteria
Déjà vu, 222, 331
Delay alternative, 273
delayAlternative (Java class), 273
delete (C++) 18
DeleteCriticalSection() (Win32), 120
Detached thread, 12
Deterministic testing, 296, 410
advantages over nondeterministic testing, 410
combined with nondeterministic testing, 414
distributed programs, applied to, 362-369
issues, 411-414
message-passing programs, applied to, 290-296
monitor-based programs, applied to, 239
Dining philosophers problem, 98
deadlock detection, and, 156
programmed using
monitors, 183-187
semaphores, 98-101
reachability testing, and, 159, 235
Distributed mutual exclusion, 341
Distributed program
definition of, 3, 312
testing and debugging
CARC-sequence, 353-362
controller, 369
instructions for running programs, 369-371
node based sequence, 360-362
object-based SYN-sequence, 353-358
simple CARC-sequence, 362
thread-based sequence, 358-360
totally ordered sequence, 359-360
Distributed programming, 19, 312
Distributed readers and writers, 346
down() operation, 84. *See also* Semaphore
Duality of monitors and message passing, 278-281
EAGAIN (Pthreads), 139
EBUSY (Pthreads), 136
else alternative, 273
elseAlternative (Java class), 274
empty(), 180
EnterCriticalSection() (Win32), 119
Entry queue, 178
Entry, 268. *See also* Rendezvous
entry (Java class) 268
Entry-based execution, 218
Eraser, 146
errno (C++), 139
error.h (C++), 10
Event (Win32), 119, 132
Event ordering problem, 96
during tracing, 330
Failure, 30, 381
types of, 398
Fault, 30, 382
example of, 397
Fairness, 3
Parameter, 115-116
False alarms, 145
FCFS semaphore, 113, 115

Feasibility, 32, 412. *See also* Deterministic testing
    CARC-sequences, of, 362-369
    Communication-sequences, of, 233, 239
    M-sequences, of, 222
    SR-sequences, of, 290
Feasible SYN-sequence, 32, 222, 396
FIFO synchronization, 298
Flowgraph, 392
Frontier of a race set, 430
getAndIncrement() (Java), 55, 82
getAndSet() (Java), 55, 82
GetExitCodeThread() (Win32), 7
getInputStream() (Java), 315
getOutputStream() (Java), 315
Global real-time clock, 332
Guard conditon, 274
HANDLE (Win32),6- 7
Happened-before relation, 333
Hold-and-wait, 98. *See also* Circular wait
I/O port, 313
illegalMonitorStateException (Java), 189
Implementation of
    monitors, 206-213
    semaphores, 92-96
    rendezvous, 269-271
Implementation relation, 396-397
Indefinite postponement, 400
Indivisible action, *see* Atomic, action
InetAddress (Java class), 314
Infinite wait, 400
InitializeCriticalSection() (Win32), 120
Instruction count, 413
IN-SYN test, 296, 410
    selecting, 411
Integer timestamp, 334
Interleaving, 27, 65
Interlocked functions (Win32), 55
InterlockedExchange (Win32), 58
InterlockedExchangeAdd() (Win32), 55
Internet Protocol (IP), 314
interrupted() (Java), 190
InterruptedException (Java), 112, 189-190
    class *Condition*, and, 199
interrupts, *see also InterruptedException*
    critical section problem, and, 47
    implementing P and V, 94, 199
Interval timer, 47
Invalid sequence, 222, 396
IOException (Java), 314
isInterrupted() (Java), 190
Java Hot Spot VM, 156
Java PathFinder, 146
java.concurrent.locks, 116, 196
java.net, 314
java.util.concurrent.atomic, 55, 82
join(), 14, 133
Kernel object, 124
Kernel space, 1
Lead race, 429
LeaveCriticalSection() (Win32), 119
length(), 180
Life cycle, 34
Limited white-box testing, 408
Link, 260
link (Java), 261-262
Linked list, 29
Livelock, 62
    detection algorithm, 404
    formal definition, 403
load-add-store, 26-28
Local real-time clock, 331
Lock, 90. *See also* Mutex Lock
lockableObject (C++ class) 121
Lockset algorithm, 144-146
LockUnlock-sequence, 147. *See also* SYN-sequence
Logical address, 1
Lost notification, 189
LPVOID (Win32), 7
Machine instructions, 26
Mailbox, 260
mailbox (Java)
    synchronous, 261
    asynchronous, 262
main thread, 2
    Java, in, 4
    Pthreads, in, 10
    Win32, in, 7
manual-reset *Event* (Win32), 132
Mapping between specification and implementation, 397
MAX_PRIORITY (Java), 5
Memory barrier, 54, 59
Memory cache, *see* Cache
Memory consistency, 74-77, 141-143
Memory management, 2
Memory mapped I/O, 76
Memory model, 77
message_ptr<> (C++ class), 263
messageParts (Java class), 320
Message passing, 258, 312. *See also* Asynchronous message passing; Synchronous message passing
    duality with monitors, 278-281
    solution to concurrent programming problems
        readers and writers, 275-278
        resource allocation, 278-281
        simulating counting semaphores, 281
MIN_PRIORITY (Java), 5
Monitor, 177

condition variables, 178-182
duality with message passing, 278-281
implementation of, 206
Java, in, 187
Pthreads, in, 196
signaling disciplines, 199-206
solution to concurrent programming problems
dining philosophers, 183
readers and writers, 187
resource allocation, 278-280
simulating binary semaphore, 183
simulating counting semaphore, 182
Monitor toolbox
C++/Win32/Pthreads, 211-213
Java, 209-211
M-sequence, 222-227. *See also* SYN-sequence
Multiprocessing, 2
Multithreading, 2
advantages, of, 3
Mutation-based testing, 415-419
Mutex, 88
Abandoned, 124
Pthreads, 136-137
Win32, 119, 122-124
Mutex lock, 90. *See also* Mutex
mutexLock
Java class, 113-115
C++ class, 125-129
mutexLocker<> (C++ class) 121-122
Mutual exclusion, 38. *See also* Critical section problem
advantages for replay, 149
Nested monitor call, 213-217
Node-based sequence, 360-362
Non-atomic operations 26-29
Non-blocking send, 259
Nondeterminism, Nondeterminsitic, 22, 23-25, 65
concurrency, and, 25
during testing and debugging, 25, 382
execution behavior, 24, 65
interleavings, 15, 23
problems for testing and debugging, 382
sources of, 23-35
Nondeterministic testing, 143-144, 409-410
combining with deterministic testing, 414
detecting deadlocks, 157
Non-signaled state (Win32), 132
notify vs. notifyAll, 191
notify() (Java),112, 189
Object (Java class), 189
Object, active, 320
Object-based SYN-sequence, 65, 387-388
Observability problem, 330-331, 413
Open monitor call, 213
OpenMutex() (Win32), 124
OpenSemaphore() (Win32), 124
Operating system, 1
Optimizations, compiler and hardware, 75-76
Order number, 283-284
Ordering events in distributed Systems, 330-339
OutputDebugString() (Win32), 243
P operation, 84. *See also* Semaphore
Parallel program, 432
Partial order, 70, 148
object-based, 387-388
thread-based, 386-387
total order, vs., 288, 386-388
Passing-the-batton, 104
Path
-based testing, 391
-based coverage criteria
all-paths, 391
branch coverage, 391
condition coverage, 391
decision/condition coverage, 391
decision coverage, 391, 414
multiple condition coverage, 392
statement coverage, 391, 414
definition, 388
domain, 390
feasibility, 389

relationship to SYN-sequence, 390
Patterns, semaphore. *See* Semaphore patterns
Peterson's algorithm, 52, 72
Port, 260. *See also* Channel
port (Java), 261
POSIX_Semaphore (C++ class) 142
POSIX1.b, 134
POSIX1.c, *see* Pthreads
Preemption, 3
Java, and, 5
Win32, and, 9
Prefix-based testing, 414, 420
PrintError(), 8
Priority, 5, 9
Probe effect, 33, 412
nondeterministic testing, and, 144
Process state information, 2
Process
definition, 1
scheduling, 2-3
state information, 2
Producer consumer problem, *see* Bounded buffer problem
Producer thread, *see* Bounded buffer problem
Program replay, 31, 411. *See also* Tracing, testing, and replay
Program testing, 31, 411. *See also* Tracing, testing, and replay

Program tracing, 31, 411. *See also* Tracing, testing, and replay
Progress requirement, 47
Progress statement, 403
Pthreads library
    condition variable, 196
    mutex lock, 136-137
    semaphore, 137-141
    threads, 9-14
pthread_attr_destroy() (Pthreads), 11
pthread_attr_init() (Pthreads), 10
pthread_attr_setscope() (Pthreads), 10
pthread_attr_t() (Pthreads), 9
pthread_cond_broadcast() (Pthreads), 196
pthread_cond_signal() (Pthreads), 196
pthread_cond_wait() (Pthreads), 196
pthread_create() (Pthreads), 9
PTHREAD_CREATE_DETACHED (Pthreads), 12
PTHREAD_CREATE_JOINABLE (Pthreads), 12
pthread_detach() (Pthreads), 12
pthread_equal() (Pthreads), 10
pthread_exit() (Pthreads), 13
pthread.h (Pthreads), 10
pthread_join() (Pthreads), 10
pthread_mutexattr_settype() (Pthreads), 137
pthread_mutex_destroy() (Pthreads), 137
pthread_mutex_init() (Pthreads), 137
PTHREAD_MUTEX_INITIALIZER (Pthreads), 137
pthread_mutex_lock() (Pthreads), 136
PTHREAD_MUTEX_RECURSIVE (Pthreads), 137
pthread_mutex_unlock() (Pthreads), 136
PTHREAD_SCOPE_PROCESS (Pthreads), 10
PTHREAD_SCOPE_SYSTEM (Pthreads), 10
pthread_self() (Pthreads), 10
pthread_setdetachstate() (Pthreads), 13
pthread_t (Pthreads), 10
Pthreads threads
    attributes, 10
    creating, 9
    detaching, 12
PulseEvent() (Win32), 132
PV-sequence, 146-147. *See also* SYN-sequence
Quantum, 3
    changing, 144, 409
Race analysis, 421. *See also* Reachability testing
    message-passing programs, of, 297
    monitor-based programs, of, 233
    semaphore-based programs, of, 157
Race set, 430. *See also* Reachability testing
Race table, 440. *See also* Reachability testing
Race variant, 421. *See also* Reachability testing
    computing, 439
    message-passing programs, of, 297
    monitor-based programs, of, 233
    semaphore- and lock-based programs, of, 157
Random delay, 300, 301, 409
    creating in
        message-passing programs, 300-301
        monitor-based programs, 238-239
        semaphore- and lock-based programs, 161-162
    during nondeterministic testing, 409
Random testing, 414
Reachability graph, 393, 400
Reachability testing, 74
    algorithm, 444-447
    event descriptors, 424-429
    message passing programs, 297-299
    monitor-based programs, 233-235
    open list, 425
    process, 420-424
    race analysis, 421
        message-passing programs, of, 297
        monitor-based programs, of, 233
        semaphore-based programs, of, 157
    race set 430
    race table, 440
    race variant, 421
        computing, 439
        message-passing programs, of, 297
        monitor-based programs, of, 233
        semaphore- and lock-based programs, of, 157
    symmetry, use of, 235
    SYN-sequences for, 424-429
Read and write events, 66-67
Readers and writers problem, 101
    distributed, 346-347
    programmed with
        message-passing, 275-278
        monitors, 187-188
        semaphores, 102-109
    reachability testing, and, 159, 299
    scheduling policies, 101-102
ReadWrite-sequence, 65-67, 146-148. *See also* SYN-sequence
Real-time clock, 331-332
Real-time, 33, 413
Recollected events, 445-446. *See also* Reachability testing
Recursive mutex, 137
Reduction of nondeterminsim, 397
ReentrantLock (Java class), 116, 196
Reentry queue, 199
Regression testing, 30, 222
reinterpret_cast<> (C++), 14
release() (Java), 190
ReleaseMutex() (Win32), 122
ReleaseSemaphore() (Win32), 124

Remote procedure call, *see* Rendezvous
Rendezvous, 266-269. *See also* Entry
Replay, *see* Tracing, testing, and replay
ResetEvent() (Win32), 132
Resource allocation problem, 86
    programmed with
        message passing, 278
        semaphores, 87
        SU monitor, 278
Resource pooling, 127
ResumeThread() (Win32), 7, 18
Round-robin scheduling, 9
run() (Java), 4
Runnable interface
    C++/Pthreads, 19
    C++/Win32, 14-15
    Java, 4
Run-time monitoring, 449
Run-time verification, 449
Scheduling policy, 2-3
select statement, 273-275
selectableEntry (Java class), 273, 290
selectablePort (Java class), 278
selective wait, 272-278
selectiveWait (Java class), 273-275
sem_destroy() (POSIX), 139
sem_post() (POSIX), 137
sem_t (POSIX), 139
sem_trywait() (POSIX), 139
SEM_VALUE_MAX (POSIX), 137
sem_wait() (POSIX), 137
semaphore (Java), 111
Semaphore, 84
    binary, 90-92
    counting, 84-86
    implementation of, 92-96
    invariant, 85, 93
    lock, vs., 92
    patterns
        condition queue, 89, 102
        enter-and-test, 89
        exit-before-wait, 89, 96, 104, 107
        mutex, 84
        passing-the-batton, 104
    solution to concurrent programming
            problems
        event ordering, 96
        bounded buffer, 96-98
        dining philosophers, 98-101
        readers and writers, 101-108
        simulating counting semaphores, 108-111
    POSIX, in, 137
    Win32, in, 119, 124
Semaphore (Java), 115
Semaphore pool, 127
semaphore.h (POSIX), 139
Sequence/Variant graph (SV-graph), 442
Sequential consistency, 74
Serializable (Java), 317
ServerSocket (Java), 315
setCompleted(), 18, 133
SetEvent() (Win32), 132
setPriority() (Java), 5
setSoTimeout() (Java), 315
Shared memory consistency, *see* Memory consistency
sharedVariable (C++), 34-37
    lockset algorithm, and, 145
    tracing and replay, for, 70-71
signal(), 179
    signal-and-continue, 180-182
    signaling disciplines, 199-206
signalAll() (Java), 180
Signal-and-continue (SC), 179, 180-182
    C++/Win32/Pthreads toolbox, 213
    comparing with SU signals, 204
    implementing with semaphores, 206-207
    Java toolbox, 210-211
Signal-and-exit (SE), 202-203, 212
Signal-and-urgent-wait (SU), 199-202, 207
    C++/Win32/Pthreads toolbox, 213
    comparing with SU signals, 204
    Java toolbox, 210-211
Signal-and-wait (SW), 253
Signaled state (Win32), 132
SignalObjectAndWait() (Win32), 134
Simple SYN-sequence, 148, 383-386
    simple CARC-sequence, 362
    simple LockUnlock-sequence, 146-150
    simple M-sequence, 217-219
    simple PV-sequence, 146-150
    simple ReadWrite-sequence, 65-68, 146-149
    simple SR-sequence, 288-290
Simulating a binary semaphore
    programmed with
        monitor, 183-184
Simulating a counting semaphore
    programmed with
        binary semaphores, 108-110
        message passing, 281
Sleep statement, 5, 144
sleep(), 60
Sleeping barber problem, 246
Smart pointer, 263
Sockets, 312
Socket (Java), 314-317
SocketTimeoutException (Java), 315
Speedup, 3. *See also* Amdahl's law
Spurious wakeup, 113, 189, 196
SR-sequence, 282. *See also* Tracing, testing, and replay; SYN-sequence
    determining feasibility, of, 290

object-based, 282-284, 387-388
simple SR-sequence, 288-290, 384-386
thread-based, 284-287, 387-388
totally ordered, 287-288
Stack size, 7
start(), 4, 18
startThread(), 16-18, 132
startThreadRunnable(), 16-18, 132
Starvation, 63-64
detection algorithm, 407
dining philosophers program, in, 98, 184
formal definition, 405
ignoring, 184
State transformation, 26
Static function, 18-19
stderr (C++),139
Strong component, 401
Strong semaphore, 92
Structural coverage criteria, s*ee* Path, Path-based coverage criteria
Subsumes relation, 392
`synchronized` (Java), 112, 187
synchronized block, 194
Synchronous message passing, 259
SYN-sequence, s*ee also* Tracing, testing, and replay
complete vs. simple, 383-386
definition of, 65, 383
implementation-based definition, 383
language-based definition, 383
total vs. partial order, 386-388
types of
CARC-sequence, 357-362
Communication-sequence, 233
CSC-sequence, 357-362
M-sequence, 222-227
simple CARC-sequence, 362
simple LockUnlock-sequence, 146-150
simple M-sequence, 217-219
simple PV-sequence, 146-150
simple ReadWrite-sequence, 65-68, 146-149
simple SR-sequence, 288-290
SR-sequence, 282-288
TCPMailbox (Java class), 318-326
TCPSelectableSynchronousMailbox (Java class), 328-329
TCPSender (Java class), 318-326
TCP sockets, 312-314
Java, in, 314-317
TCPSynchronousSender (Java class), 326-328
TCPUnreliableMailbox (Java class), 351
TCPUnreliableSelectableMailbox (Java class), 351
TDThread (class) 34-37
Test oracle, 412
Testability, 149
Testing, s*ee also* Deterministic testing; Non-deterministic testing;
Tracing, testing, and replay
definition of, 30
problems and issues, 30
tools, 33
this pointer, 19
Thread
C++/Pthreads, 19
C++/Win32, 14
Java, 4-5
Thread attribute, 10
Thread-based SYN-sequence, 65, 386
Thread.currentThread() (Java), 115
Thread, definition of, 2
Thread IDs, 36-37
Thread.interrupt() (Java), 112
Thread schedule, 221-222
Thread.sleep() (Java), 5
Ticket algorithm, 54
Tie-breaker algorithm, see Peterson's algorithm
Time slice, 2
Timed wait, 189
Timestamp
causality, and, 332-224
event ordering, and, 330
integer timestamp, 334-335
real-time clock, 331-332
shared variables, for, 339
vector timestamp, 335-339
Timestamps for reachability testing
object centric, 437-439
thread-centric, 433-437
total order, 70-73, 148
partial order, vs., 288, 386-388
Tracing, testing, and replay, s*ee also* Deterministic testing; Nondeterministic testing; Reachability testing; SYN-sequence; feasibility
adaptive tracing, 411
approaches to testing, 408-419
class *sharedVariable*<>, 37-38
class *TDThread*, 34-37
deadlock detection, 154-157
distributed programs, s*ee* Distributed program, testing and debugging
lock-based programs, 143-154
instructions for running programs, 160-163
memory consistency, and, 74-77
message-passing programs, 281-296
instructions for running programs, 299-304
monitor-based programs, 217-233

instructions for running programs, 235-243
semaphore-based programs, 143-154
instructions for running programs, 160-163
shared variables, 64-71
Transmission Control Protocol (TCP), 313-317
Transport protocol, 313
try – finally, 199
tryAcquire() (Java), 115
tryLock() (Java), 116
Unbounded waiting, 59. *See also* Bounded waiting requirement
Unisex bathroom problem, 169, 248, 305
UnknownHostException (Java), 314
up() operation, 84. *See also* Semaphore
Urgent-signal-and-continue (USC), 204. *See also* signal()
User Data Protocol (UDP), 313
V operation, 84. *See also* Semaphore
Valid SYN-sequence, 222, 396
validity, 32, 412. *See also* Valid SYN-sequence
Vector timestamp, 335-339. *See also* Timestamp
Version number, 66
Visual C++, 243
`volatile`, 53-54, 76
VP operation, 94-96. *See also* Semaphore
implementation of, 127
wait() and *notify()* (Java),111-112, 189
wait()
monitor operation, 178-182
Java operation, s*ee wait()* and *notify()*
WAIT_ABANDONED (Win32), 124
Wait, circular, 407
WAIT_FAILED (Win32), 123, 125
WAIT_OBJECT_0 (Win32), 123, 125
WAIT_TIMEOUT (Win32), 123, 125
Wait-for graph, 154
WaitForMultipleObjects() (Win32), 7, 134
WaitForSingleObject() (Win32), 122, 124
Watchdog thread, 231
Weak semaphore, 92
White-box testing, 34, 408
WINAPI (Win32), 8
Win32 threads
creating, 6
priority, 9
scheduling, 9
win32Critical_Section (C++/Win32 class), 119-121
win32Mutex (C++/Win32 class), 124
win32Semaphore (C++/Win32 class), 125-127
XADD, 55, 77