

Contents

- 1 Introduction** 1
 - 1.1 Control Flow 1
 - 1.2 Basic Control Structures 2
 - 1.3 Routine/Member Call/Return 2
 - 1.4 Recursion 4
 - 1.5 Routine Pointer 5
 - 1.6 Exception 6
 - 1.7 Coroutine 7
 - 1.8 Concurrency 7
 - 1.9 Summary 7
 - Reference 7
- 2 Advanced Control Flow** 9
 - 2.1 Basic Control Flow 9
 - 2.2 GOTO Statement 10
 - 2.3 Multi-Exit (Mid-Test) Loop 13
 - 2.3.1 Loop-Exit Criticisms 17
 - 2.3.2 Linear Search Example 17
 - 2.4 Static Multi-Level Exit 20
 - 2.5 Routine 23
 - 2.6 Recursion 26
 - 2.7 Functional Programming 33
 - 2.8 Routine Pointers 37
 - 2.9 Iterator 41
 - 2.10 Dynamic Multi-Level Exit 43
 - 2.11 Summary 47
 - 2.12 Questions 48
 - References 58
- 3 Exceptions** 61
 - 3.1 Traditional Approaches 62
 - 3.2 Exception Handling 67

3.3	EHM Objectives	69
3.4	Execution Environment	70
3.5	Terminology	71
3.6	Control-Flow Taxonomy	72
3.7	Exception Models	74
3.7.1	Static Propagation/Termination	75
3.7.2	Dynamic Propagation/Termination	77
3.7.3	Dynamic Propagation/Resumption	82
3.8	EHM Features	89
3.8.1	Handler Association	89
3.8.2	Derived Exception-Type	90
3.8.3	Catch-Any	92
3.8.4	Exception Parameters	93
3.8.5	Exception List	94
3.8.6	Bound Exceptions and Conditional Handling	99
3.9	Choosing an Exception Model	101
3.10	Derived Exception Implications	104
3.11	Propagation Mechanism	105
3.12	μ C++ EHM	107
3.13	Exception Type	107
3.13.1	Creation and Destruction	108
3.13.2	Inherited Members	108
3.14	Raising	109
3.15	Handler	110
3.15.1	Termination	110
3.15.2	Resumption	111
3.15.3	Termination/Resumption	112
3.16	Bound Exceptions	113
3.16.1	Matching	114
3.16.2	Termination	114
3.16.3	Resumption	115
3.17	Inheritance	115
3.18	Summary	116
3.19	Questions	117
	References	123
4	Coroutine	125
4.1	Fibonacci Series	125
4.1.1	Routine Solution	126
4.1.2	Class Solution	128
4.1.3	Coroutine Solution	129
4.2	Formatting	132
4.3	Coroutine Construction	135
4.4	Correct Coroutine Usage	135
4.5	Iterator	136

4.6	Parsing	137
4.7	Coroutine Details	145
4.7.1	Coroutine Creation and Destruction	145
4.7.2	Inherited Members	146
4.7.3	Coroutine Control and Communication	149
4.8	Semi- and Full Coroutine	150
4.8.1	Semi-Coroutine	151
4.8.2	Full Coroutine	152
4.8.3	Ping/Pong	154
4.9	Producer-Consumer Problem	156
4.9.1	Semi-coroutine Solution	157
4.9.2	Full Coroutine Solution	159
4.10	Nonlocal Propagation	162
4.11	Summary	164
4.12	Questions	165
	References	190
5	Concurrency	191
5.1	Why Write Concurrent Programs	192
5.2	Why Concurrency Is Complex?	192
5.3	Concurrent Hardware	195
5.4	Execution States	197
5.5	Threading Model	200
5.6	Concurrent Systems	201
5.7	Speedup	202
5.8	Basic Concurrent Programming	205
5.8.1	Thread Creation/Termination	205
5.8.2	Thread Graph	206
5.8.3	COBEGIN/COEND	207
5.8.4	START/WAIT	208
5.8.5	Termination Synchronization	210
5.9	μ C++ Threads	210
5.10	Task Details	214
5.10.1	Task Creation and Destruction	214
5.10.2	Inherited Members	215
5.11	Concurrent Propagation	216
5.11.1	Enabling/Disabling Propagation	217
5.12	Divide-and-Conquer	218
5.13	Synchronization and Communication During Execution	219
5.14	Summary	222
5.15	Questions	222
	References	232
6	Atomicity	233
6.1	Critical Section	234
6.2	Mutual Exclusion	235

6.2.1	Mutual Exclusion Game	235
6.2.2	Self-Checking Critical Section	237
6.3	Software Solution	238
6.3.1	Lock	238
6.3.2	Alternation	240
6.3.3	Declare Intent	241
6.3.4	Retract Intent	243
6.3.5	Prioritized Retract Intent	244
6.3.6	Fair Retract Intent	246
6.3.6.1	Alternation	246
6.3.6.2	Racing	248
6.3.7	Read/Write-Safe	251
6.3.8	<i>N</i> -Thread Mutual Exclusion	254
6.3.8.1	Prioritized Retract Intent	254
6.3.8.2	Knuth/De Bruijn	257
6.3.8.3	Eisenberg and McGuire	262
6.3.8.4	Bakery Algorithm	265
6.3.8.5	Tournament	269
6.3.8.6	Arbiter	270
6.4	Hardware Solutions	272
6.4.1	MIPS R4000	273
6.4.2	Test and Set	273
6.4.3	Fetch and Assign	277
6.4.4	Fetch and Increment	278
6.4.5	Compare and Assign	281
6.4.6	Mellor-Crummey and Scott	283
6.4.7	Nonreentrant Problem	287
6.5	Atomic (Lock-Free) Data-Structure	287
6.5.1	Stack	288
6.5.2	Generalizing Lock-Free	295
6.5.3	Lock Versus Lock-Free	297
6.6	Exotic Atomic Instructions	298
6.7	Uniprocessor Tricks	302
6.8	Summary	302
6.9	Questions	303
	References	310
7	Locks	313
7.1	Lock Taxonomy	315
7.2	Spin Lock	316
7.2.1	Spin Lock Details	317
7.2.2	Synchronization	318
7.2.3	Mutual Exclusion	318
7.3	Blocking Locks	319
7.3.1	Mutex Lock	320

7.3.1.1	Owner Lock Details	325
7.3.1.2	Mutual Exclusion	325
7.3.1.3	Lock-Release Pattern	326
7.3.1.4	Stream Lock	327
7.3.2	Synchronization Lock	328
7.3.2.1	Condition Lock Details	331
7.3.2.2	Synchronization	332
7.3.3	Barrier	333
7.3.4	Binary Semaphore.....	339
7.3.4.1	Synchronization	339
7.3.4.2	Mutual Exclusion	340
7.3.4.3	Implementation	341
7.3.5	Counting (General) Semaphore.....	341
7.3.5.1	Synchronization	342
7.3.5.2	Mutual Exclusion	342
7.3.5.3	Implementation	343
7.3.6	Semaphore Details	344
7.4	Lock and COBEGIN.....	345
7.5	Producer-Consumer with Buffer	348
7.5.1	Unbounded Buffer	348
7.5.2	Bounded Buffer	350
7.6	Readers and Writer	352
7.6.1	Split Binary Semaphores and Baton Passing	356
7.6.2	Solution 1	359
7.6.3	Solution 2	363
7.6.4	Solution 3	363
7.6.5	Solutions 4 and 5	365
7.6.6	Solution 6	370
7.6.7	Solution 7	373
7.7	Summary	373
7.8	Questions	375
	References.....	394
8	Concurrency Errors	395
8.1	Race Error	395
8.2	No Progress	397
8.2.1	Livelock.....	397
8.2.2	Starvation	399
8.2.3	Deadlock.....	399
	8.2.3.1 Synchronization Deadlock	400
	8.2.3.2 Mutual Exclusion Deadlock	400
8.3	Deadlock Prevention	403
	8.3.1 Synchronization Deadlock Prevention	403
	8.3.2 Mutual Exclusion Deadlock Prevention.....	404

8.4	Deadlock Avoidance	407
8.4.1	Synchronization Deadlock Avoidance	409
8.4.2	Mutual-Exclusion Deadlock Avoidance.....	409
8.4.2.1	Banker's Algorithm	410
8.4.2.2	Allocation Graph	413
8.5	Deadlock Detection and Recovery	417
8.6	Summary	418
8.7	Questions	419
	References.....	422
9	High-Level Concurrency Constructs	425
9.1	Critical Region	426
9.2	Conditional Critical Region	428
9.2.1	Implementation.....	432
9.3	Monitor	434
9.3.1	Mutex Calling Mutex	437
9.4	Scheduling	438
9.4.1	External Scheduling.....	439
9.4.2	Internal Scheduling.....	443
9.5	External and Internal Scheduling	447
9.5.1	Combining Scheduling Techniques	447
9.5.2	Scheduling Selection	449
9.5.2.1	Member Parameter Scheduling	450
9.5.2.2	Delay Scheduling	453
9.6	Monitor Details	455
9.6.1	Monitor Creation and Destruction.....	456
9.6.2	Accept Statement.....	457
9.6.3	Condition Variables and Wait/Signal Statements	459
9.7	Readers and Writer Problem	460
9.7.1	Solution 3	463
9.7.2	Solutions 4 and 5	464
9.7.3	Solution 6	470
9.8	Condition, Wait, Signal vs. Counting Semaphore, P, V	470
9.9	Monitor Errors	471
9.10	Coroutine-Monitor	472
9.10.1	Coroutine-Monitor Creation and Destruction	472
9.10.2	Coroutine-Monitor Control and Communication	473
9.11	Monitor Taxonomy	474
9.11.1	Explicit and Implicit Signal Monitor	474
9.11.2	Implicit Monitor Scheduling.....	475
9.11.3	Monitor Classification	477
9.11.3.1	Explicit-Signal Monitors	477
9.11.3.2	Immediate-Return Monitor	478
9.11.3.3	Automatic-Signal Monitors	480
9.11.3.4	Simplified Classification	481

9.12	Monitor Equivalence	482
9.12.1	Non-FIFO Simulations	482
9.12.1.1	Problems	485
9.12.2	FIFO Simulation	487
9.13	Monitor Comparison	490
9.13.1	Priority/No-Priority	490
9.13.2	Blocking/Non-blocking	491
9.13.3	Quasi-blocking	493
9.13.4	Extended Immediate Return	493
9.13.5	Automatic Signal	493
9.14	Summary	494
9.15	Questions	494
	References	521
10	Active Objects	523
10.1	Execution Properties	523
10.2	Indirect/Direct Communication	526
10.2.1	Indirect Communication	526
10.2.2	Direct Communication	529
10.3	Task	531
10.4	Scheduling	531
10.4.1	External Scheduling	531
10.4.2	Internal Scheduling	537
10.5	External and Internal Scheduling	539
10.6	Accepting the Destructor	539
10.7	When a Task Becomes a Monitor	544
10.8	Task Details	545
10.8.1	Accept Statement	545
10.9	When to Create a Task	547
10.10	Producer-Consumer Problem	548
10.11	Tasks and Coroutines	550
10.12	Inheritance Anomaly Problem	551
10.13	Summary	552
10.14	Questions	553
	References	561
11	Enhancing Concurrency	563
11.1	Server Side	565
11.1.1	Buffers	566
11.1.2	Administrator	567
11.2	Client Side	570
11.2.1	Returning Values	571
11.2.2	Tickets	571
11.2.3	Call-Back Routine	572
11.2.4	Future	572

11.3	μ C++ Future	573
11.3.1	Client Operations	574
11.3.2	Server Operations	574
11.3.3	Explicit Storage Management	575
11.3.4	Example	576
11.3.5	Implicit Storage Management	577
11.3.6	Example	578
11.4	Future Access	578
11.4.1	Select Statement	579
11.4.2	Wait Queue	581
11.5	Future Server	583
11.6	Executors	583
11.7	Summary	585
11.8	Questions	586
	References	613
12	Optimization	615
12.1	Basic Optimizations	616
12.2	Sequential Optimizations	617
12.2.1	Reordering	617
12.2.2	Eliding	619
12.2.3	Replication	619
12.3	Memory Hierarchy	620
12.3.1	Cache Review	621
12.3.2	Cache Coherence	622
12.4	Concurrent Optimizations	626
12.4.1	Disjoint Reordering	627
12.4.2	Eliding	628
12.4.3	Replication	628
12.5	Memory Models	630
12.6	Preventing Optimization Problems	631
12.7	Summary	634
12.8	Questions	635
	References	635
13	Control Flow Paradigms	637
13.1	Coroutines	637
13.1.1	Generators	641
13.2	Thread and Lock Library	644
13.2.1	Pthreads	644
13.2.1.1	Thread Creation and Termination	644
13.2.1.2	Thread Synchronization and Mutual Exclusion	647
13.2.2	Object-Oriented Thread Library: C++	657
13.2.2.1	Thread Creation and Termination	657

	13.2.2.2 Thread Synchronization and Mutual Exclusion	663
13.3	Threads and Message Passing	665
	13.3.1 Send	667
	13.3.1.1 Blocking Send	667
	13.3.1.2 Nonblocking Send	667
	13.3.1.3 Multicast/Broadcast Send	668
	13.3.2 Receive	668
	13.3.2.1 Receive Any	669
	13.3.2.2 Receive Message Specific	669
	13.3.2.3 Receive Thread Specific	669
	13.3.2.4 Receive Reply	670
	13.3.2.5 Receive Combinations	671
	13.3.3 Message Format	671
	13.3.4 Typed Messages	672
13.4	Concurrent Languages	673
	13.4.1 Ada 95	673
	13.4.1.1 Thread Creation and Termination	673
	13.4.1.2 Thread Synchronization and Mutual Exclusion	676
	13.4.2 SR/Concurrent C++	690
	13.4.3 Java	692
	13.4.3.1 Thread Creation and Termination	692
	13.4.3.2 Thread Synchronization and Mutual Exclusion	694
	13.4.4 java.util.concurrent	700
	13.4.5 Go	703
	13.4.6 C++11 Concurrency	705
13.5	Concurrent Models	708
	13.5.1 Actor Model	709
	13.5.2 Linda Model	709
	13.5.3 OpenMP	713
	13.5.4 MPI	716
13.6	Summary	722
13.7	Questions	723
	References	725
14	μ C++ Grammar	727
	Reference	729
	Index	731

<http://www.springer.com/978-3-319-25701-3>

Understanding Control Flow

Concurrent Programming Using $\mu\text{C}++$

Buhr, P.A.

2016, XXI, 741 p. 100 illus. in color., Hardcover

ISBN: 978-3-319-25701-3