

Table of Contents

Preface	v
Chapter 1: The Concurrent Computing Landscape .	1
1.1 The Essence of Concurrent Programming	2
1.2 Hardware Architectures	4
1.2.1 Processors and Caches	4
1.2.2 Shared-Memory Multiprocessors	6
1.2.3 Distributed-Memory Multicomputers and Networks	8
1.3 Applications and Programming Styles	10
1.4 Iterative Parallelism: Matrix Multiplication	13
1.5 Recursive Parallelism: Adaptive Quadrature	17
1.6 Producers and Consumers: Unix Pipes	19
1.7 Clients and Servers: File Systems	21
1.8 Peers: Distributed Matrix Multiplication	23
1.9 Summary of Programming Notation	26
1.9.1 Declarations	26
1.9.2 Sequential Statements	27
1.9.3 Concurrent Statements, Processes, and Procedures	29
1.9.4 Comments	31
Historical Notes	31
References	33
Exercises	34

Part 1: Shared-Variable Programming	39
Chapter 2: Processes and Synchronization	41
2.1 States, Actions, Histories, and Properties	42
2.2 Parallelization: Finding Patterns in a File	44
2.3 Synchronization: The Maximum of an Array	48
2.4 Atomic Actions and Await Statements	51
2.4.1 Fine-Grained Atomicity	51
2.4.2 Specifying Synchronization: The Await Statement	54
2.5 Producer/Consumer Synchronization	56
2.6 A Synopsis of Axiomatic Semantics	57
2.6.1 Formal Logical Systems	58
2.6.2 A Programming Logic	59
2.6.3 Semantics of Concurrent Execution	62
2.7 Techniques for Avoiding Interference	65
2.7.1 Disjoint Variables	65
2.7.2 Weakened Assertions	66
2.7.3 Global Invariants	68
2.7.4 Synchronization	69
2.7.5 An Example: The Array Copy Problem Revisited	70
2.8 Safety and Liveness Properties	72
2.8.1 Proving Safety Properties	73
2.8.2 Scheduling Policies and Fairness	74
Historical Notes	77
References	80
Exercises	81
Chapter 3: Locks and Barriers	93
3.1 The Critical Section Problem	94
3.2 Critical Sections: Spin Locks	97
3.2.1 Test and Set	98
3.2.2 Test and Test and Set	100
3.2.3 Implementing Await Statements	101
3.3 Critical Sections: Fair Solutions	104
3.3.1 The Tie-Breaker Algorithm	104
3.3.2 The Ticket Algorithm	108
3.3.3 The Bakery Algorithm	111
3.4 Barrier Synchronization	115
3.4.1 Shared Counter	116
3.4.2 Flags and Coordinators	117

3.4.3	Symmetric Barriers	120
3.5	Data Parallel Algorithms	124
3.5.1	Parallel Prefix Computations	124
3.5.2	Operations on Linked Lists	127
3.5.3	Grid Computations: Jacobi Iteration	129
3.5.4	Synchronous Multiprocessors	131
3.6	Parallel Computing with a Bag of Tasks	132
3.6.1	Matrix Multiplication	133
3.6.2	Adaptive Quadrature	134
	Historical Notes	135
	References	139
	Exercises	141

Chapter 4: Semaphores

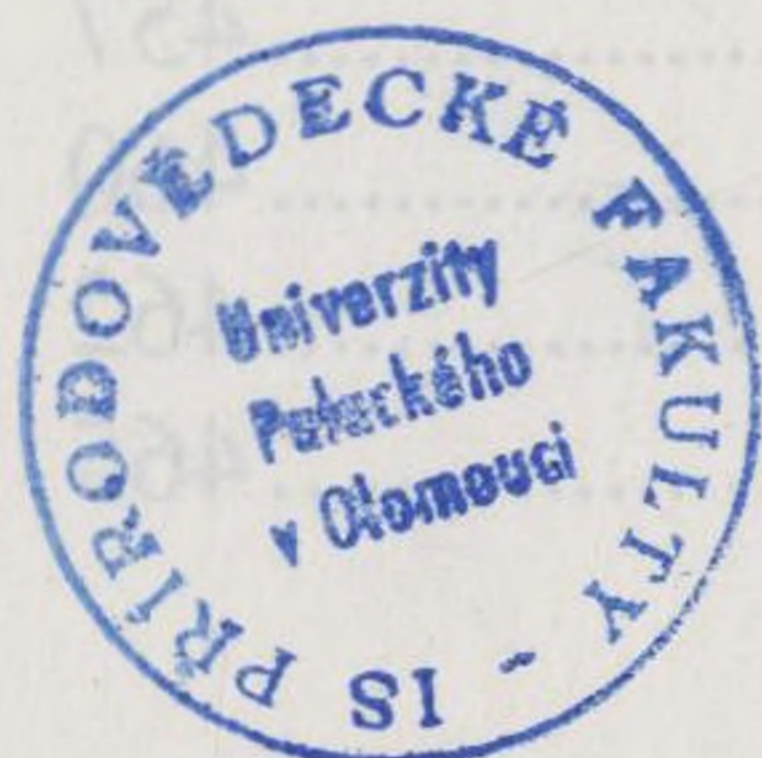
4.1	Syntax and Semantics	154
4.2	Basic Problems and Techniques	156
4.2.1	Critical Sections: Mutual Exclusion	156
4.2.2	Barriers: Signaling Events	156
4.2.3	Producers and Consumers: Split Binary Semaphores	158
4.2.4	Bounded Buffers: Resource Counting	160
4.3	The Dining Philosophers	164
4.4	Readers and Writers	166
4.4.1	Readers/Writers as an Exclusion Problem	167
4.4.2	Readers/Writers Using Condition Synchronization	169
4.4.3	The Technique of Passing the Baton	171
4.4.4	Alternative Scheduling Policies	175
4.5	Resource Allocation and Scheduling	178
4.5.1	Problem Definition and General Solution Pattern	178
4.5.2	Shortest-Job-Next Allocation	180
4.6	Case Study: Pthreads	184
4.6.1	Thread Creation	185
4.6.2	Semaphores	186
4.6.3	Example: A Simple Producer and Consumer	186
	Historical Notes	188
	References	190
	Exercises	191

Chapter 5: Monitors

5.1	Syntax and Semantics	204
5.1.1	Mutual Exclusion	206

5.1.2	Condition Variables	207
5.1.3	Signaling Disciplines	208
5.1.4	Additional Operations on Condition Variables	212
5.2	Synchronization Techniques	213
5.2.1	Bounded Buffers: Basic Condition Synchronization	213
5.2.2	Readers and Writers: Broadcast Signal	215
5.2.3	Shortest-Job-Next Allocation: Priority Wait	217
5.2.4	Interval Timer: Covering Conditions	218
5.2.5	The Sleeping Barber: Rendezvous	221
5.3	Disk Scheduling: Program Structures	224
5.3.1	Using a Separate Monitor	228
5.3.2	Using an Intermediary	230
5.3.3	Using a Nested Monitor	235
5.4	Case Study: Java	237
5.4.1	The Threads Class	238
5.4.2	Synchronized Methods	239
5.4.3	Parallel Readers/Writers	241
5.4.4	Exclusive Readers/Writers	243
5.4.5	True Readers/Writers	245
5.5	Case Study: Pthreads	246
5.5.1	Locks and Condition Variables	246
5.5.2	Example: Summing the Elements of a Matrix	248
	Historical Notes	250
	References	253
	Exercises	255
	Chapter 6: Implementations	265
6.1	A Single-Processor Kernel	266
6.2	A Multiprocessor Kernel	270
6.3	Implementing Semaphores in a Kernel	276
6.4	Implementing Monitors in a Kernel	279
6.5	Implementing Monitors Using Semaphores	283
	Historical Notes	284
	References	286
	Exercises	287
	Part 2: Distributed Programming	291
	Chapter 7: Message Passing	295
7.1	Asynchronous Message Passing	296

7.2	Filters: A Sorting Network	298
7.3	Clients and Servers	302
7.3.1	Active Monitors	302
7.3.2	A Self-Scheduling Disk Server	308
7.3.3	File Servers: Conversational Continuity	311
7.4	Interacting Peers: Exchanging Values	314
7.5	Synchronous Message Passing	318
7.6	Case Study: CSP	320
7.6.1	Communication Statements	321
7.6.2	Guarded Communication	323
7.6.3	Example: The Sieve of Eratosthenes	326
7.6.4	Occam and Modern CSP	328
7.7	Case Study: Linda	334
7.7.1	Tuple Space and Process Interaction	334
7.7.2	Example: Prime Numbers with a Bag of Tasks	337
7.8	Case Study: MPI	340
7.8.1	Basic Functions	341
7.8.2	Global Communication and Synchronization	343
7.9	Case Study: Java	344
7.9.1	Networks and Sockets	344
7.9.2	Example: A Remote File Reader	345
	Historical Notes	348
	References	351
	Exercises	353
Chapter 8: RPC and Rendezvous		361
8.1	Remote Procedure Call	362
8.1.1	Synchronization in Modules	364
8.1.2	A Time Server	365
8.1.3	Caches in a Distributed File System	367
8.1.4	A Sorting Network of Merge Filters	370
8.1.5	Interacting Peers: Exchanging Values	371
8.2	Rendezvous	373
8.2.1	Input Statements	374
8.2.2	Client/Server Examples	376
8.2.3	A Sorting Network of Merge Filters	379
8.2.4	Interacting Peers: Exchanging Values	381
8.3	A Multiple Primitives Notation	382
8.3.1	Invoking and Servicing Operations	382
8.3.2	Examples	384



8.4	Readers/Writers Revisited	386
8.4.1	Encapsulated Access	387
8.4.2	Replicated Files	389
8.5	Case Study: Java	393
8.5.1	Remote Method Invocation	393
8.5.2	Example: A Remote Database	395
8.6	Case Study: Ada	397
8.6.1	Tasks	398
8.6.2	Rendezvous	399
8.6.3	Protected Types	401
8.6.4	Example: The Dining Philosophers	403
8.7	Case Study: SR	406
8.7.1	Resources and Globals	406
8.7.2	Communication and Synchronization	408
8.7.3	Example: Critical Section Simulation	409
	Historical Notes	411
	References	415
	Exercises	416
Chapter 9: Paradigms for Process Interaction		423
9.1	Manager/Workers (Distributed Bag of Tasks)	424
9.1.1	Sparse Matrix Multiplication	424
9.1.2	Adaptive Quadrature Revisited	428
9.2	Heartbeat Algorithms	430
9.2.1	Image Processing: Region Labeling	432
9.2.2	Cellular Automata: The Game of Life	435
9.3	Pipeline Algorithms	437
9.3.1	A Distributed Matrix Multiplication Pipeline	438
9.3.2	Matrix Multiplication by Blocks	441
9.4	Probe/Echo Algorithms	444
9.4.1	Broadcast in a Network	444
9.4.2	Computing the Topology of a Network	448
9.5	Broadcast Algorithms	451
9.5.1	Logical Clocks and Event Ordering	452
9.5.2	Distributed Semaphores	454
9.6	Token-Passing Algorithms	457
9.6.1	Distributed Mutual Exclusion	457
9.6.2	Termination Detection in a Ring	460
9.6.3	Termination Detection in a Graph	462
9.7	Replicated Servers	465

9.7.1	Distributed Dining Philosophers	466
9.7.2	Decentralized Dining Philosophers	467
	Historical Notes	471
	References	474
	Exercises	477

Chapter 10: Implementations 487

10.1	Asynchronous Message Passing	488
10.1.1	Shared-Memory Kernel	488
10.1.2	Distributed Kernel	491
10.2	Synchronous Message Passing	496
10.2.1	Direct Communication Using Asynchronous Messages	497
10.2.2	Guarded Communication Using a Clearinghouse	498
10.3	RPC and Rendezvous	504
10.3.1	RPC in a Kernel	504
10.3.2	Rendezvous Using Asynchronous Message Passing	507
10.3.3	Multiple Primitives in a Kernel	509
10.4	Distributed Shared Memory	515
10.4.1	Implementation Overview	516
10.4.2	Page Consistency Protocols	518
	Historical Notes	520
	References	521
	Exercises	522

Part 3: Parallel Programming 527

Chapter 11: Scientific Computing 533

11.1	Grid Computations	534
11.1.1	Laplace's Equation	534
11.1.2	Sequential Jacobi Iteration	535
11.1.3	Jacobi Iteration Using Shared Variables	540
11.1.4	Jacobi Iteration Using Message Passing	541
11.1.5	Red/Black Successive Over-Relaxation (SOR)	546
11.1.6	Multigrid Methods	549
11.2	Particle Computations	553
11.2.1	The Gravitational N-Body Problem	554
11.2.2	Shared-Variable Program	555
11.2.3	Message-Passing Programs	559

Later chapters cover applications and programming techniques in detail. The chapters are organized into three parts—shared-variable programming,

11.2.4	Approximate Methods	569
11.3	Matrix Computations	573
11.3.1	Gaussian Elimination	573
11.3.2	LU Decomposition	575
11.3.3	Shared-Variable Program	576
11.3.4	Message-Passing Program	581
	Historical Notes	583
	References	584
	Exercises	585

Chapter 12: Languages, Compilers, Libraries, and Tools 591

12.1	Parallel Programming Libraries	592
12.1.1	Case Study: Pthreads	593
12.1.2	Case Study: MPI	593
12.1.3	Case Study: OpenMP	595
12.2	Parallelizing Compilers	603
12.2.1	Dependence Analysis	604
12.2.2	Program Transformations	607
12.3	Languages and Models	614
12.3.1	Imperative Languages	616
12.3.2	Coordination Languages	619
12.3.3	Data Parallel Languages	620
12.3.4	Functional Languages	623
12.3.5	Abstract Models	626
12.3.6	Case Study: High-Performance Fortran (HPF)	629
12.4	Parallel Programming Tools	633
12.4.1	Performance Measurement and Visualization	633
12.4.2	Metacomputers and Metacomputing	634
12.4.3	Case Study: The Globus Toolkit	636
	Historical Notes	638
	References	642
	Exercises	644

Glossary 647

Index 657