JEAN BACON & TIM HARRIS

OPERATING SYSTEMS

CONCURRENT AND DISTRIBUTED SOFTWARE DESIGN

ADDISON-WESLEY

2-005-588-1

المكتبة المركزية

Operating Systems

Concurrent and Distributed Software Design

Jean Bacon and Tim Harris
University of Cambridge

ADDISON-WESLEY
An imprint of Pearson Education

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney • Singapore • Hong Kong
Tokyo • Seoul • Taipei • New Delhi • Cape Town • Madrid • Mexico City • Amsterdam • Munich • Paris • Milan

Brief contents

Preface		xxiii
1	System design requirements	1
PART I SYSTEM DESIGN: TECHNOLOGY AND PRINCIPLES		35
2	System structure and dynamic execution	37
3	The hardware interface, I/O and communications	73
4	Support for processes	109
5	Memory management	163
6	File management	193
7	Fundamentals of distributed systems	229
8	Security	247
PART II CONCURRENCY CONTROL IN MAIN MEMORY		273
9	System structure	277
10	Low-level synchronization: Implementation	297
11	Low-level synchronization: Algorithms	321
12	IPC with shared memory	357
13	IPC and system structure	391
14	IPC without shared memory	405
15	Crash resilience and persistent data	429
16	Distributed IPC	443
PART III TRANSACTIONS		485
17	Composite operations	487
18	Resource allocation and deadlock	499

19	Transactions	519
20	Concurrency control	543
21	Recovery	569
22	Distributed transactions	583
23	Distributed computations	599
PART IV CASE STUDIES		**615**
24	Classical UNIX	617
25	LINUX, Solaris and contemporary UNIX	651
26	Extensible systems	679
27	Windows 2000	707
28	The World Wide Web	741
29	Middleware	757
30	Transaction processing monitors and systems	789
Appendix: Evolution of computer systems		809
Bibliography		831
Glossary		847
Author index		853
Subject index		857

Contents

Preface xxiii

1 System design requirements 1

1.1 Inherently concurrent systems 2

1.1.1 Real-time and embedded systems 3

1.1.2 Operating systems and distributed operating systems 7

1.1.3 Window-based interfaces 11

1.1.4 Database management and transaction processing systems 12

1.1.5 Middleware 14

1.1.6 Distributed systems: The World Wide Web 15

1.2 Supporting potentially concurrent applications 15

1.2.1 Replicated code, partitioned data 16

1.2.2 Pipelined processing 17

1.2.3 Tree-structured algorithms 17

1.2.4 Shared data 18

1.2.5 Application areas 18

1.2.6 Requirements for supporting concurrent applications 19

1.3 Architectures for software systems 19

1.3.1 System classification 20

1.3.2 Conventional uniprocessors 21

1.3.3 Shared-memory multiprocessors 22

1.3.4 Multicomputer multiprocessors 23

1.3.5 Dataflow (data-driven) architectures 23

1.3.6 Architectures for functional languages 24

1.3.7 Network-based systems 25

1.3.8 Summary of hardware bases for concurrent systems 28

1.4 Defining a concurrent system 29

1.5 Systems implementation requirements 30

1.6 Security, protection and fault tolerance in system design 31

Exercises 32

PART I SYSTEM DESIGN: TECHNOLOGY AND PRINCIPLES **35**

2 System structure and dynamic execution **37**

2.1 System structure: Conceptual views 38
2.1.1 Types and abstract data types 38
2.1.2 Interfaces 40
2.1.3 State 40
2.2 System structure: Implementational views 41
2.2.1 Object-oriented programming 41
2.2.2 Interfaces 42
2.2.3 Implementation 43
2.2.4 Object interactions 45
2.2.5 The life-cycle of an object 47
2.2.6 Specialization and generalization 49
2.2.7 Composition 50
2.2.8 Object collaboration 50
2.3 The process concept 51
2.3.1 Process by analogy 52
2.3.2 Multi-threaded processes 54
2.4 Operating system functions 55
2.5 Operating system invocation and protection 57
2.5.1 Protection of the operating system 58
2.6 Operating system structure 59
2.6.1 Layering 60
2.6.2 Microkernels 60
2.7 Object structuring within and supported by operating systems 64
2.7.1 Object naming 64
2.7.2 Object protection, invocation and sharing 65
2.7.3 Unified object mechanisms 66
2.8 Distributed object systems, platforms and middleware 66
2.9 Security and protection 67
2.9.1 An object model for access control 68
2.10 Summary 69
Study questions and exercises 70

3 The hardware interface, I/O and communications **73**

3.1 Overview 73
3.2 Device interfacing 74
3.2.1 Processor and device speeds 76
3.2.2 CISC and RISC computers 77
3.2.3 A simple device interface 78
3.2.4 Polling and interrupts 79
3.2.5 Interrupt handling: Priorities 80
3.2.6 Interrupt vectors 81
3.2.7 The RISC approach to interrupt handling 81
3.2.8 Direct memory access (DMA) devices 84
3.2.9 Memory-mapped I/O 85
3.2.10 Timers 85

3.3 Exceptions 86
3.3.1 Exceptions caused by a running program 86
3.3.2 System (privileged) mode and user (unprivileged) mode 87
3.3.3 The system call mechanism 88
3.3.4 Summary of the use of the exception mechanism 88
3.3.5 User-level exception handling 90
3.4 Multiprocessors 90
3.5 User-level input and output 92
3.5.1 Buffers and synchronization 93
3.5.2 Synchronous and asynchronous input and output 94
3.6 Communications management 94
3.7 Communications networks, interfaces and drivers 96
3.7.1 Ethernet 97
3.7.2 Ring-based LANs 98
3.7.3 Examples of network interfaces 99
3.8 Communications software 99
3.8.1 The ISO reference model for Open Systems Interconnection 100
3.8.2 Connection-oriented and connectionless communication 103
3.9 Communications handling within and above an operating system 104
3.10 Summary 105
Exercises 106

4 Support for processes 109

4.1 Use of processes in systems 110
4.2 Processes and processors 111
4.3 Process state 113
4.3.1 Saving process state 114
4.3.2 Context switching 114
4.4 Synchronizing with the hardware: Events and the WAIT operation 115
4.4.1 Race conditions 116
4.4.2 Event and process objects 117
4.5 The process data structure 118
4.6 Scheduling: General approaches 118
4.6.1 Unary, binary and general scheduling 119
4.6.2 Process behaviour and priority 119
4.7 Scheduling for shared-memory multiprocessors 122
4.8 Process scheduling to meet real-time requirements 123
4.8.1 System structure and response to events 126
4.9 Process abstraction and implementation 126
4.10 Operating system structure and placement of processes 128
4.11 Multi-threaded process implementation 129
4.12 Processes in languages, runtime systems and operating systems 131
4.13 Process state in language systems and operating systems 132
4.13.1 Activation records and the runtime stack 134
4.13.2 The heap and garbage collection 134

4.14 Sequential programs with system calls 136
4.15 Evolution of concurrency in programming languages 137
4.15.1 Examples 137
4.15.2 Concurrency from a sequential language 139
4.15.3 Coroutines 140
4.15.4 Processes 144
4.15.5 Issues arising 146
4.16 Implementation of processes in language systems 147
4.16.1 Specification, creation and termination 147
4.16.2 Parental control of processes 149
4.16.3 Exception handling in programming languages 150
4.16.4 Storage allocation for language-level processes 150
4.17 Thread package architectures 151
4.18 Java threads and exceptions 154
4.19 Summary 157
Study questions and exercises 158

5 Memory management 163

5.1 Memory management 164
5.2 The memory hierarchy 164
5.3 The address space of a process 165
5.3.1 Address binding 166
5.3.2 Static binding 167
5.3.3 Dynamic binding 167
5.3.4 Hardware-assisted relocation and protection 168
5.4 Segmented virtual memory 170
5.4.1 Segmented virtual addresses 172
5.4.2 Address translation 173
5.4.3 Segmentation: Summary 173
5.5 Paged virtual memory 174
5.5.1 Address translation 175
5.5.2 Copy-on-write paging 176
5.5.3 Paging: Summary 177
5.6 Combining segmentation and paging 178
5.6.1 Hardware-supported segments, hardware-supported pages 178
5.6.2 Software-supported segments, hardware-supported pages 179
5.7 Operating system data structures 179
5.7.1 Multi-level page tables 181
5.7.2 Guarded page tables 182
5.7.3 Inverted page tables 182
5.7.4 Managing pages in main memory 183
5.7.5 Managing pages on backing store 183
5.7.6 Page replacement 184
5.8 An example of a memory management unit (MMU) 185
5.9 Memory management in system design 188
5.10 Summary 189
Exercises 190

6 **File management** 193

6.1 File management 193
6.2 An overview of filing system functions 194
6.3 File and directory structure 195
6.3.1 Pathnames and working directories 196
6.3.2 File sharing: Access rights and links 197
6.3.3 Existence control 197
6.4 The filing system interface 199
6.4.1 The directory service as type manager 200
6.4.2 The directory service interface 200
6.4.3 The file service interface 201
6.5 The filing system implementation 201
6.5.1 Hard links and symbolic links 204
6.5.2 Locating a file on disk 206
6.5.3 Storing new media types 208
6.6 Modern file system design 209
6.6.1 Logical volume management 210
6.6.2 Striping and mirroring 211
6.6.3 Journaling and logging 212
6.7 Network-based file servers 213
6.7.1 Open and closed storage architectures 214
6.7.2 The storage service interface 215
6.7.3 Location of function 216
6.7.4 Stateless servers: NFS 217
6.7.5 Write-mostly functionality and caching 219
6.7.6 File identifiers and protection at the storage service level 221
6.8 Integrating virtual memory and storage 222
6.8.1 File mapping 222
6.8.2 Object mapping 224
6.9 Summary 225
Exercises 225

7 **Fundamentals of distributed systems** 229

7.1 Introduction 229
7.2 Evolution of distributed systems for the workplace 230
7.3 Personal computing 232
7.4 Model and software architecture 233
7.5 Special characteristics of distributed systems 233
7.6 Time in distributed systems 234
7.6.1 Physical earth time 234
7.6.2 Use of time by distributed processes 235
7.6.3 Logical time: Event ordering 236
7.6.4 Algorithms for clock synchronization 237
7.7 Naming 239
7.7.1 Creating unique names 239
7.7.2 Pure and impure names 239
7.7.3 An example: The internet Domain Name Service (DNS) 241
7.7.4 Naming, name services and binding 242
7.7.5 Attributes stored by name services 243

7.8 Mobile users, computers and objects 244
7.8.1 Mobile computers 244
7.8.2 Pervasive computing 245
7.9 Summary 245
Exercises 246

8 Security 247

8.1 Scenarios and security requirements 247
8.1.1 Client–server interactions 248
8.1.2 Importing source code and object code 249
8.1.3 Connection to home 249
8.2 Threats and risk analysis 250
8.3 Approaches to encryption 251
8.3.1 Secret key encryption 252
8.3.2 Public key encryption 252
8.3.3 One-way functions 254
8.4 Algorithms 254
8.4.1 Substitution ciphers 255
8.4.2 Transposition ciphers 256
8.4.3 Rotor machines (e.g. Enigma) 256
8.4.4 Encryption by computer 257
8.5 Protocols 259
8.5.1 Secure communication 260
8.5.2 Session keys in hybrid protocols 261
8.5.3 Authentication 262
8.5.4 Replay attacks and 'nonces' 264
8.6 Examples 265
8.6.1 X.509 certification 265
8.6.2 Simple Public Key Infrastructure (SPKI) 266
8.6.3 Authentication and key exchange 266
8.6.4 Pretty Good Privacy (PGP) 268
8.6.5 Secure Socket Layer (SSL) 269
8.7 Summary 270
Exercises 271

PART II CONCURRENCY CONTROL IN MAIN MEMORY 273

9 System structure 277

9.1 Processes sharing an address space 278
9.1.1 Placement of processes within the subsystem 279
9.2 Processes in separate address spaces 280
9.3 Sharing the operating system 281
9.4 Summary of process placement in the two models 283
9.5 Requirements for process interaction 285
9.6 Types of process interaction 287

9.7 A process interaction 289

9.7.1 Problems when processes share data in memory 289

9.7.2 Problems when processes do not share data in memory 291

9.7.3 Granularity of concurrency 291

9.8 Definition of single concurrent actions 292

Study questions and exercises 294

10 Low-level synchronization: Implementation 297

10.1 Process synchronization compared with event signal and wait 298

10.2 Mutual exclusion 299

10.3 Hardware support for mutual exclusion 302

10.3.1 Mutual exclusion without hardware support 305

10.4 Semaphores 305

10.5 Styles of use of semaphores 307

10.5.1 Mutual exclusion 307

10.5.2 Condition synchronization 308

10.5.3 Multiple instances of a resource 309

10.6 Implementation of semaphore operations 311

10.6.1 Concurrency in the semaphore implementation 311

10.6.2 Scheduling the semWait queue, priority inversion and inheritance 313

10.6.3 Location of IPC implementation and process (thread) management 314

10.7 Summary 316

Study questions and exercises 316

11 Low-level synchronization: Algorithms 321

11.1 Introduction 321

11.2 An example of semaphores in system design: The THE system 322

11.3 The producer–consumer, bounded buffer problem 324

11.3.1 Use of buffers 324

11.3.2 Definition of a cyclic or bounded buffer 325

11.3.3 Algorithm for a single producer and a single consumer 326

11.3.4 Algorithm for more than one producer or consumer 328

11.4 Safety and liveness properties 330

11.5 The multiple readers, single writer problem 330

11.6 Limitations of semaphores 335

11.7 Eventcounts and sequencers 336

11.7.1 Use of eventcounts for synchronization 337

11.7.2 Use of a sequencer to enforce mutual exclusion 337

11.7.3 Producer–consumer, bounded buffer with eventcounts and sequencers 338

11.7.4 Implementation of eventcounts and sequencers 338

11.7.5 Discussion of eventcounts and sequencers 340

11.8 POSIX threads 340

11.8.1 Objects, handles and attributes 340

11.8.2 Synchronization 341

11.8.3 pthread operations summary 343

11.9 Summary 345

11.10 Case study with exercises: Management of a disk block cache 346
11.10.1 Disk read and write and the requirement for buffers and a cache 346
11.10.2 Allocated and free buffers 347
11.10.3 The structure of a buffer 348
11.10.4 Outline of the algorithms for buffer access 349
Study questions and exercises 352

12 IPC with shared memory 357

12.1 Critical regions in programming languages 358
12.2 Monitors 360
12.2.1 Java: Synchronization overview 363
12.2.2 Java: Critical regions 365
12.2.3 Java: Condition synchronization 366
12.2.4 Java monitors: An example 368
12.2.5 Some further issues relating to monitors 371
12.2.6 Single resource allocator 372
12.2.7 Bounded buffer manager 373
12.2.8 Multiple readers, single writer 375
12.2.9 Discussion of monitors 376
12.3 Synchronization at the granularity of operations 377
12.3.1 Path expressions 378
12.3.2 Active objects 379
12.4 Summary 384
Study questions and exercises 385

13 IPC and system structure 391

13.1 Styles of inter-process communication 391
13.2 System structure and IPC with shared memory 393
13.3 System structure and IPC without shared memory 394
13.4 Systems where shared-memory communication is appropriate 395
13.5 Systems where shared-memory communication is not appropriate 396
13.6 Examples from classical UNIX 396
13.7 Overview of inter-process communication 398
13.8 Duality of system structures 400
13.9 Naming 402
13.10 Summary 402
Study questions and exercises 403

14 IPC without shared memory 405

14.1 Introduction 405
14.2 Use of files for common data 406
14.3 UNIX pipes 408
14.3.1 Use of pipes by UNIX: Command composition 409
14.3.2 Evaluation of the pipe mechanism 409

14.4 Asynchronous message passing 409
14.5 Variations on basic message passing 412
14.5.1 Receiving from 'anyone' 412
14.5.2 Request and reply primitives 413
14.5.3 Multiple ports per process 414
14.5.4 Input ports, output ports and channels 415
14.5.5 Global ports 416
14.5.6 Broadcast and multicast 417
14.5.7 Message forwarding 418
14.5.8 Specifying WAIT time 418
14.5.9 Discarding out-of-date messages 419
14.6 Implementation of asynchronous message passing 419
14.7 Synchronous message passing 421
14.8 Message passing in programming languages 423
14.8.1 occam channels for synchronous communication 423
14.8.2 The Linda abstraction 424
14.9 Multi-threading in clients and servers 426
14.10 Summary 427
Study questions and exercises 428

15 Crash resilience and persistent data 429

15.1 Crashes 429
15.2 A model of a crash 430
15.3 Crash resilience or failure transparency 431
15.4 Idempotent (repeatable) operations 431
15.5 Atomic operations on persistent objects 432
15.5.1 Volatile, persistent and stable storage 433
15.6 Implementation of atomic operations 433
15.6.1 Logging 434
15.6.2 Shadowing 434
15.7 Non-volatile memory 436
15.8 A single operation on persistent data 437
15.9 Database management systems' requirements on operating systems 438
15.10 Summary 439
Study questions and exercises 440

16 Distributed IPC 443

16.1 Introduction 443
16.2 Special characteristics of distributed systems 444
16.3 Distributed IPC: Message passing 445
16.3.1 Distributing IPC 445
16.3.2 Distributed, asynchronous message passing 446
16.4 Integration of IPC with communications 448
16.4.1 IPC above sockets 448
16.5 Java's sockets and streams 451
16.5.1 The java.net package 454
16.5.2 The InterestRate client 455

16.5.3 *The client–server mechanism and asynchronous message passing* 455
16.6 Distributed programming paradigms 456
16.6.1 *Synchronous and asynchronous communication* 456
16.6.2 *Procedural versus message passing style* 457
16.7 Remote procedure call (RPC) 458
16.7.1 *An RPC system* 458
16.7.2 *The RPC protocol with network or server congestion* 459
16.7.3 *The RPC protocol with node crash and restart* 460
16.7.4 *An example: CCLU RPC call semantics* 461
16.7.5 *RPC and the ISO reference model* 462
16.8 RPC–language integration 463
16.8.1 *Distribution transparency* 463
16.8.2 *Argument marshalling* 465
16.8.3 *Object-oriented systems* 467
16.8.4 *Type checking and consistency checking* 467
16.8.5 *Data representation for a heterogeneous environment* 468
16.9 Java's RMI: RPC in the general object model 469
16.9.1 *RMI and the general object model* 469
16.9.2 *RMI mechanisms* 470
16.9.3 *RMI: An example* 473
16.9.4 *Comparison of RMI with stream and socket programming* 475
16.10 Critique of synchronous invocation 475
16.11 Naming, location and binding 477
16.11.1 *Naming the objects used in IPC* 477
16.11.2 *Locating named objects* 478
16.11.3 *The ANSA trader* 480
16.12 Summary of Part II 481
Study questions and exercises 482

PART III TRANSACTIONS **485**

17 Composite operations **487**

17.1 Composite operations 487
17.2 Composite operations in main memory 488
17.3 Composite operations involving main memory and persistent memory 489
17.3.1 *Examples from operating systems* 489
17.3.2 *An example from a database system* 490
17.4 Concurrent execution of composite operations 490
17.4.1 *Desirability of concurrent execution* 491
17.5 Potential problems 492
17.5.1 *Uncontrolled interleaving of suboperations* 492
17.5.2 *Visibility of the effects of suboperations* 494
17.5.3 *Deadlock* 494
17.6 Crashes 494
17.7 Summary 496
Study questions and exercises 496

18 Resource allocation and deadlock 499

18.1 Requirements for dynamic allocation 499
18.2 Deadlock 500
18.3 Livelock and starvation 501
18.4 Conditions for deadlock to exist 502
18.4.1 Deadlock prevention 503
18.5 The dining philosophers problem 504
18.6 Object allocation graphs 505
18.7 Data structures and algorithms for deadlock detection 506
18.7.1 An algorithm for deadlock detection 508
18.7.2 Example 508
18.7.3 Action on detection of deadlock 509
18.8 Deadlock avoidance 510
18.8.1 Problems of deadlock avoidance 511
18.9 Information on releasing objects: Multiphase processes 512
18.10 Distributed deadlocks 512
18.10.1 Distributed deadlock detection 513
18.11 Summary 515
Study questions and exercises 515

19 Transactions 519

19.1 Introduction 519
19.2 Transaction specification and programming 520
19.3 The definition of serializability and consistency 523
19.4 The ACID properties of transactions 525
19.5 Indicating specific orderings of transactions 526
19.6 A system model for transaction processing 526
19.6.1 Non-commutative (conflicting) pairs of operations 527
19.6.2 Condition for serializability 529
19.7 Dependency graphs for transactions 531
19.8 Histories and serialization graphs 534
19.9 Dealing with aborts: More about the property of isolation 536
19.9.1 Cascading aborts 536
19.9.2 The ability to recover state 537
19.10 Summary 539
Study questions and exercises 540

20 Concurrency control 543

20.1 Introduction 543
20.2 Concurrent composite operations in main memory only 544
20.2.1 Objects in the main memory of a single computer 544
20.2.2 Objects in main memory in a distributed system 545
20.2.3 Systematic approaches to concurrent program development 546
20.3 Structure of transaction management systems 547
20.4 Concurrency control through locking 548
20.4.1 Two-phase locking (2PL) 549
20.4.2 An example of two-phase locking 549
20.4.3 Semantic locking 550
20.4.4 Deadlock in two-phase locking 551

20.5 Time-stamp ordering (TSO) 551
20.5.1 Cascading aborts and recovery of state 554
20.6 Optimistic concurrency control (OCC) 555
20.7 Summary 563
Study questions and exercises 564

21 Recovery **569**
21.1 Requirements for recovery 569
21.2 The object model, object state and recovery 570
21.3 Concurrency, crashes and the properties of transactions 571
21.4 Logging and shadowing for crash resilience 572
21.5 Use of a recovery log 573
21.5.1 Log records and their use in recovery 573
21.5.2 Log write-ahead 574
21.5.3 Checkpoints and the checkpoint procedure 574
21.6 Idempotent undo and redo operations 575
21.7 Transaction states on a failure 576
21.8 An algorithm for recovery 577
21.9 Location databases for mobile objects 578
21.10 Summary 579
Exercises 580

22 Distributed transactions **583**
22.1 An object model for distributed systems 583
22.2 Distributed transaction processing 584
22.3 Communication 585
22.4 Concurrency control: Two-phase locking (2PL) 586
22.5 Concurrency control: Time-stamp ordering (TSO) 587
22.6 Optimistic concurrency control (OCC) 588
22.7 Commit and abort in a distributed system 589
22.8 Atomic commitment: The two-phase commit (2PC) protocol 590
22.9 Two-phase validation for OCC 593
22.10 Summary 595
Study questions and exercises 596

23 Distributed computations **599**
23.1 Introduction 599
23.2 Process groups 600
23.2.1 Leadership election 601
23.3 Consistency of data replicas 602
23.3.1 Quorum assembly for strong consistency 603
23.3.2 Large-scale systems 604
23.4 Ordering message delivery 605
23.4.1 Vector clocks 606
23.5 Distributed, *N*-process mutual exclusion 607
23.5.1 Algorithms 609
23.6 Summary of Part III 611
Exercises 612

PART IV CASE STUDIES 615

24 Classical UNIX 617

24.1 Introduction 617
24.2 Evolution of UNIX 618
24.3 System structure and the UNIX kernel 619
24.3.1 System calls and kernel execution 621
24.3.2 Device management 622
24.3.3 The buffer cache 623
24.4 File system interfaces 624
24.4.1 Mounting and unmounting file systems 625
24.4.2 Graph navigation and the current directory 626
24.5 File system implementation 626
24.5.1 Inodes 626
24.5.2 Directory files 628
24.5.3 Access protection 628
24.5.4 Allocation of disk blocks to a file system 630
24.5.5 File system data structures in main memory 631
24.5.6 Consistency issues 632
24.6 Process creation, termination and scheduling 633
24.6.1 Process creation 635
24.6.2 Parent–child synchronization on process termination 635
24.6.3 Process creation during system initialization 636
24.6.4 Process creation by the command interpreter 637
24.6.5 Process scheduling 639
24.6.6 The swapping algorithm 640
24.6.7 Process states 641
24.7 IPC: Pipes and signals 643
24.7.1 Pipes between commands 643
24.7.2 Signals 645
24.7.3 Overview of classical IPC mechanisms 646
24.8 Summary 648
Exercises 648

25 LINUX, Solaris and contemporary UNIX 651

25.1 Introduction 651
25.2 Kernel structure 652
25.2.1 Loadable kernel modules 652
25.2.2 Communication with kernel modules 653
25.2.3 Interrupt handling 654
25.2.4 Multiprocessor kernel structure 657
25.3 SVr4 IPC 658
25.3.1 Message queues 659
25.3.2 SVr4 semaphore sets 660
25.3.3 SVr4 shared memory segments 661
25.4 Sockets and file subsystem integrated IPC 663
25.4.1 Named pipes 663
25.4.2 Sockets API 663
25.4.3 Server process structure 667

25.5 Memory management 669
25.5.1 Memory-mapped files 670
25.5.2 Communication through shared memory 671
25.5.3 Physical memory allocation 671
25.6 Multiprocessor scheduling 673
25.6.1 UNIX scheduling for multiple CPUs 674
25.6.2 Exposing parallelism 674
25.7 Summary 676
Exercises 677

26 Extensible systems 679

26.1 Mechanisms for extensibility 680
26.2 Compile-time specialization 681
26.3 Microkernel operating systems 683
26.3.1 Mach 685
26.3.2 Mach IPC 686
26.3.3 Memory objects and object sharing in main memory 687
26.4 Downloadable code 688
26.5 Exokernels and vertical structuring 690
26.5.1 The exokernel architecture 691
26.5.2 Nemesis 692
26.6 Nested virtual machines 696
26.7 Extensible virtual machines 699
26.7.1 The Sceptre XVM 700
26.7.2 Java operating systems 702
26.8 Summary 704
Exercises 705

27 Windows 2000 707

27.1 Introduction to Windows 2000 707
27.1.1 Design principles 708
27.2 System structure 709
27.2.1 Executive components 711
27.3 The object model and object manager 712
27.3.1 How executive objects are used 713
27.3.2 The structure of an object 714
27.3.3 Object types 715
27.3.4 Object names and directories 716
27.3.5 Object handles 718
27.3.6 Type-specific object methods 718
27.3.7 Protecting objects 719
27.4 The kernel 719
27.4.1 Kernel objects, kernel process and thread objects 720
27.4.2 Thread scheduling and dispatching 720
27.4.3 Interrupt and exception handling 722
27.5 Processes, threads, fibres and concurrency control 723
27.5.1 Thread synchronization 725
27.5.2 Alerts and asynchronous procedure calls 725

	27.5.3 Address space management	726
	27.5.4 Sharing memory: Sections, views and mapped files	726
	27.5.5 Memory protection	728
27.6	The I/O subsystem	728
	27.6.1 I/O design features	729
	27.6.2 I/O processing	731
	27.6.3 Windows Driver Model	732
27.7	The NT filing system, NTFS	733
	27.7.1 NTFS requirements	733
	27.7.2 New features of NTFS	733
	27.7.3 NTFS design outline	734
27.8	Networking	736
27.9	Summary	738
	Exercises	739
28	**The World Wide Web**	**741**
28.1	A simple web-client, web-server interaction	742
28.2	Naming web pages	744
28.3	Communication using HTTP	745
28.4	Document representation	746
28.5	Executing programs at server and client	748
28.6	Security	750
28.7	Concurrency control	751
28.8	Scalability issues	751
28.9	Web-based middleware: XML and SOAP	752
28.10	Summary	754
	Exercises	754
29	**Middleware**	**757**
29.1	Middleware paradigms	757
	29.1.1 Evolution of object-oriented middleware	759
29.2	Java middleware	761
	29.2.1 Java for web programming	761
	29.2.2 Java remote method invocation (RMI)	762
	29.2.3 Jini, including JavaSpaces	764
	29.2.4 The Java component model, JavaBeans	766
	29.2.5 Java Messaging Service (JMS)	767
	29.2.6 JXTA	767
29.3	OMG and OMA	768
	29.3.1 OMA Reference Model component definitions	769
	29.3.2 The OMG Object Model	771
29.4	CORBA	772
	29.4.1 The ORB core	773
	29.4.2 OMG's Interface Definition Language (IDL)	773
	29.4.3 Other CORBA features	774
	29.4.4 CORBA support for asynchronous communication	776
29.5	ODMG	777
29.6	COM, DCOM and .NET	778
29.7	Message-oriented middleware (MOM)	780

	29.7.1 IBM's MQSeries	782
	29.7.2 TIB/Rendezvous	784
29.8	Summary	786
	Exercises	787
30	**Transaction processing monitors and systems**	**789**
30.1	Transaction processing monitors	790
	30.1.1 Use of processes and IPC	791
	30.1.2 Buffered transaction requests	794
	30.1.3 Monitoring system load and performance	795
30.2	Introduction to some electronic funds transfer (EFT) applications	795
	30.2.1 Paying by cheque	795
	30.2.2 Paying by credit card	797
	30.2.3 Paying by debit card: Point of sale transactions	798
	30.2.4 Some security issues	798
30.3	International inter-bank payments: SWIFT	799
30.4	Authentication by PIN	799
30.5	The international automatic teller machine (ATM) network service	802
	30.5.1 How bank accounts are held	803
	30.5.2 Local control of ATMs	803
	30.5.3 Remote use	804
30.6	Load and traffic in TP systems	805
30.7	Summary and trends	807
	Exercises	807
	Appendix: Evolution of computer systems	**809**
A.1	Evolution and exploitation of technology	809
A.2	Operating systems and distributed operating systems	814
	A.2.1 Multiprogramming batch	815
	A.2.2 Centralized time-sharing (interactive) systems	816
	A.2.3 Workstations and personal computers	819
A.3	Databases	820
A.4	Concurrency control	821
	A.4.1 Mutual exclusion without hardware support	821
	A.4.2 Requirements	822
	A.4.3 The N-process mutual exclusion protocol of Eisenberg and McGuire (1972)	823
	A.4.4 The N-process bakery algorithm	825
	Exercises	828
	Bibliography	**831**
	Glossary	**847**
	Author index	**853**
	Subject index	**857**