

CONTENTS

Preface to the second edition	v
1 Introduction: Examples and requirements	1
1.1 Inherently concurrent systems	2
1.1.1 Real-time systems	
1.1.2 Database management and transaction processing systems	
1.1.3 Operating systems and distributed operating systems	
1.1.4 Middleware	
1.1.5 Window-based interfaces	
1.1.6 The World Wide Web and information servers	
1.2 Potentially concurrent applications	15
1.2.1 Replicated code, partitioned data	
1.2.2 Pipelined processing	
1.2.3 Tree-structured algorithms	
1.2.4 Shared data	
1.2.5 Application areas	
1.2.6 Requirements for supporting concurrent applications	
1.3 Architectures for concurrent systems	19
1.3.1 System classification	
1.3.2 Conventional uniprocessors	
1.3.3 Shared-memory multiprocessors	
1.3.4 Multicomputer multiprocessors	
1.3.5 Dataflow (data-driven) architectures	
1.3.6 Architectures for functional languages	
1.3.7 Network-based systems	
1.3.8 Summary of hardware bases for concurrent systems	
1.4 A definition of a concurrent system	29
1.5 Requirements for implementing concurrent systems	30
1.6 Security, protection and fault tolerance in system design	31
Exercises	32
Further reading	33

PART I BACKGROUND AND FUNDAMENTALS **35**

2 System structure and dynamic execution	37
2.1 System structure	38
2.1.1 Modules and interfaces	
2.1.2 Abstract data types	
2.1.3 Objects	
2.1.4 Objects and object managers	
2.1.5 Object granularity	

2.2	Static structure and dynamic behaviour of a system	43
2.2.1	The process concept	
2.2.2	Multi-threaded processes	
2.3	Operating system functions	47
2.4	Operating system invocation and protection	49
2.4.1	Protection of the operating system	
2.5	Operating system structure	51
2.5.1	Layering	
2.5.2	Microkernels	
2.6	Object structuring within and supported by operating systems	56
2.6.1	Object naming	
2.6.2	Object protection, invocation and sharing	
2.6.3	Unified object mechanisms	
2.7	Distributed object systems, platforms and middleware	59
2.8	Security and protection	60
2.8.1	An object model for access control	
2.9	Summary	61
	Exercises	62
3	The hardware interface, I/O and communications	65
3.1	Overview	65
3.2	Device interfacing	66
3.2.1	Processor and device speeds	
3.2.2	CISC and RISC computers	
3.2.3	A simple device interface	
3.2.4	Polling and interrupts	
3.2.5	Interrupt handling: Priorities	
3.2.6	Interrupt vectors	
3.2.7	The RISC approach to interrupt handling	
3.2.8	Direct memory access (DMA) devices	
3.2.9	Memory-mapped I/O	
3.2.10	Timers	
3.3	Exceptions	78
3.3.1	Exceptions caused by a running program	
3.3.2	System (privileged) mode and user (unprivileged) mode	
3.3.3	The system call mechanism	
3.3.4	Summary of the user of the exception mechanism	
3.3.5	User-level exception handling	
3.4	Multiprocessors	82
3.5	User-level input and output	84
3.5.1	Buffers and synchronization	
3.5.2	Synchronous and asynchronous input and output	
3.6	Communications management	87
3.7	Communications networks, interfaces and drivers	88
3.7.1	Ethernet	
3.7.2	Ring-based LANs	
3.7.3	Examples of network interfaces	
3.8	Communications software	92
3.8.1	The ISO reference model for Open Systems Interconnection	
3.8.2	Connection-oriented and connectionless communication	
3.9	Communications handling within and above an operating system	96
3.10	Summary	98
	Exercises	98

4	Support for processes	101
4.1	Use of processes in systems	102
4.2	Processes and processors	103
4.3	Process state	105
	4.3.1 Saving process state 4.3.2 Context switching	
4.4	Synchronizing with the hardware – events and the WAIT operation	107
	4.4.1 Race conditions 4.4.2 Event and process objects	
4.5	The process data structure	109
4.6	Scheduling – general approaches	110
	4.6.1 Unary, binary and general scheduling 4.6.2 Process behaviour and priority	
4.7	Scheduling for shared-memory multiprocessors	113
4.8	Process scheduling to meet real-time requirements	115
	4.8.1 System structure and response to events	
4.9	Process abstraction and implementation	118
4.10	Operating system structure and placement of processes	120
4.11	Multi-threaded process implementation	121
4.12	Processes in language systems and operating systems	123
4.13	Process state in language systems and operating systems	124
	4.13.1 Procedures and activation records 4.13.2 The heap and garbage collection	
4.14	Sequential programs with system calls	128
4.15	Evolution of concurrency in programming languages	129
	4.15.1 Examples 4.15.2 Concurrency from a sequential language 4.15.3 Coroutines	
	4.15.4 Processes 4.15.5 Issues arising	
4.16	Implementation of processes in language systems	140
	4.16.1 Specification, creation and suicide of processes	
	4.16.2 Parental control of processes	
	4.16.3 Exception handling in programming languages	
	4.16.4 Storage allocation for language-level processes	
4.17	Thread package architectures	143
4.18	Summary	147
	Exercises	149
5	Fundamentals of distributed systems	153
5.1	Introduction	153
5.2	Evolution of distributed systems for the workplace	154
5.3	Ubiquitous computing	156
5.4	Models and software architecture	157
5.5	Special characteristics of distributed systems	157

5.6	Time in distributed systems	158
5.6.1	Physical earth time	
5.6.2	Use of time by distributed processes	
5.6.3	Logical time – event ordering	
5.6.4	Algorithms for clock synchronization	
5.7	Naming	163
5.7.1	Creating unique names	
5.7.2	Pure and impure names	
5.7.3	An example: the internet Domain Name Service (DNS)	
5.7.4	Naming, name services and binding	
5.7.5	Attributes stored by name services	
5.8	Mobile users, computers and objects	168
5.8.1	Mobile computers	
5.8.2	Mobile users and objects	
5.9	Requirements for security in distributed systems	169
5.10	Summary	170
6	Memory management	171
6.1	Memory management	171
6.2	The memory hierarchy	172
6.3	The address space of a process	173
6.4	Dynamic relocation hardware	175
6.5	Protection hardware	175
6.6	Relocation and protection of a single contiguous segment	176
6.7	Several segments per process	176
6.8	Paging	180
6.9	Paged segments	182
6.9.1	Software-supported segments, hardware-supported pages	
6.10	Paging for large virtual address spaces	184
6.10.1	Inverted page tables	
6.10.2	Hierarchical and guarded page tables	
6.10.3	Copy-on-write paging	
6.11	Address translation in the storage hierarchy	187
6.12	An example of a memory management unit (MMU)	188
6.13	An example of operating system page fault handling	191
6.14	Memory management in system design	191
6.15	Summary	192
	Exercises	193
7	File management	195
7.1	File management	195
7.2	An overview of filing system functions	196
7.3	File and directory structure	197
7.3.1	Pathnames and working directories	
7.3.2	File sharing: access rights and links	
7.3.3	Existence control	

7.4	The filing system interface	201
7.4.1	The directory service as type manager	
7.4.2	The directory service interface	
7.4.3	The file service interface	
7.5	The filing system implementation	203
7.5.1	Hard links and symbolic links	
7.5.2	Locating a file on disk	
7.5.3	Storing new media types	
7.6	Network-based file servers	211
7.6.1	Open and closed storage architectures	
7.6.2	The storage service interface	
7.6.3	Location of function	
7.6.4	Stateless servers – NFS	
7.6.5	Write-mostly functionality and caching	
7.6.6	File identifiers and protection at the storage service level	
7.7	Integrating virtual memory and storage	220
7.7.1	File mapping	
7.7.2	Object mapping	
7.8	Summary	223
	Exercises	223

PART II SINGLE CONCURRENT ACTIONS **225**

8	System structure	229
8.1	Processes sharing an address space	230
8.1.1	Placement of processes within the subsystem	
8.2	Processes in separate address spaces	232
8.3	Sharing the operating system	234
8.4	Summary of process placement in the two models	235
8.5	Requirements for process interaction	237
8.6	Types of process interaction	239
8.7	A process interaction	241
8.7.1	Problems when processes share data in memory	
8.7.2	Problems when processes do not share data in memory	
8.7.3	Granularity of concurrency	
8.8	Definition of single concurrent actions	245
	Exercises	246
9	Low-level synchronization primitives: Implementation	249
9.1	Process synchronization compared with event signal and wait	250
9.2	Synchronization to achieve exclusive access to shared data	253
9.2.1	Hardware test-and-set or equivalent	
9.2.2	Trends in computer architecture	
9.3	<i>N</i> -process mutual exclusion without hardware support	260
9.3.1	Requirements	
9.3.2	The <i>N</i> -process mutual exclusion protocol of Eisenberg and McGuire (1972)	
9.3.3	The <i>N</i> -process bakery algorithm	

9.4	Semaphores	266
9.5	Use of semaphores	267
	9.5.1 Mutual exclusion 9.5.2 Synchronization of cooperating processes	
	9.5.3 Multiple instances of a resource	
9.6	Implementation of semaphore operations	270
	9.6.1 Concurrency in the semaphore implementation	
	9.6.2 Scheduling the WAIT queue, priority inversion and inheritance	
	9.6.3 Location of IPC implementation	
9.7	Summary	275
	Exercises	276
10	Low-level primitives: Use in systems and languages	279
10.1	Introduction	279
10.2	An example of semaphores in system design: the THE system	280
10.3	The producer–consumer, bounded buffer problem	282
	10.3.1 Use of buffers 10.3.2 Definition of a cyclic or bounded buffer	
	10.3.3 Algorithm for a single producer and a single consumer	
	10.3.4 Algorithm for more than one producer or consumer	
10.4	Safety and liveness properties	285
10.5	The multiple readers, single writer problem	286
10.6	Limitations of semaphores	289
10.7	Eventcounts and sequencers	290
	10.7.1 Use of eventcounts for synchronization	
	10.7.2 Use of a sequencer to enforce mutual exclusion	
	10.7.3 Producer–consumer, bounded buffer with eventcounts and sequencers	
	10.7.4 Implementation of eventcounts and sequencers	
	10.7.5 Discussion of eventcounts and sequencers	
10.8	POSIX threads	293
	10.8.1 Objects, handles and attributes 10.8.2 Synchronization	
	10.8.3 pthread operations summary	
10.9	Summary	299
10.10	Case study with exercises: Management of a disk block cache	300
	10.10.1 Disk read and write and the requirement for buffers and a cache	
	10.10.2 Allocated and free buffers 10.10.3 The ststructure of a buffer	
	10.10.4 Outline of the algorithms for buffer access	
	Further exercises	306
11	Language primitives for shared memory	309
11.1	Critical regions at the language level	309

11.2	Monitors	311
11.2.1	Single resource allocator	
11.2.2	Bounded buffer manager	
11.2.3	Multiple readers, single writer	
11.2.4	Discussion of monitors	
11.3	Per-object exclusion	319
11.4	Synchronization at the granularity of operations	320
11.4.1	Path expressions	
11.4.2	Active objects	
11.5	Summary	323
	Exercises	324
12	IPC and system structure	327
12.1	Evolution of inter-process communication	327
12.2	Procedural system structure	329
12.3	System structure and IPC based on no shared memory	330
12.4	Processes in UNIX	330
12.5	Systems where shared-memory communication is appropriate	332
12.6	Systems where shared-memory communication is not appropriate	333
12.7	Overview of inter-process communication	333
12.8	Duality of system structures	335
12.9	Naming	337
12.10	Summary	337
	Exercises	338
13	IPC without shared memory	339
13.1	Introduction	339
13.2	Use of files for common data	340
13.3	UNIX pipes	341
13.3.1	Use of pipes by UNIX: Command composition	
13.3.2	Evaluation of the pipe mechanism	
13.4	Asynchronous message passing	343
13.5	Variations on basic message passing	346
13.5.1	Receiving from 'anyone'	
13.5.2	Request and reply primitives	
13.5.3	Multiple ports per process	
13.5.4	Input ports, output ports and channels	
13.5.5	Global ports	
13.5.6	Broadcast and multicast	
13.5.7	Message forwarding	
13.5.8	Specifying WAIT time	
13.5.10	Discarding out-of-date messages	
13.6	Implementation of asynchronous message passing	353
13.7	Unifying messages and interrupts	354
13.8	Synchronous message passing	355
13.9	Message passing in programming languages	357
13.9.1	occam channels for synchronous communication	
13.9.2	The Linda abstraction	
13.10	Multi-threaded servers	360

13.11	Summary	361
	Exercises	362
14	Crash resilience and persistent data	363
14.1	Crashes	363
14.2	A model of a crash	364
14.3	Crash resilience or failure transparency	365
14.4	Idempotent (repeatable) operations	365
14.5	Atomic operations	366
	14.5.1 Volatile, persistent and stable storage	
14.6	Implementation of atomic operations	367
	14.6.1 Logging 14.6.2 Shadowing	
14.7	Non-volatile memory	370
14.8	A single operation on persistent data	371
14.9	Database management systems' requirements on operating systems	372
14.10	Summary	373
	Exercises	374
15	Distributed IPC	375
15.1	Client–server and object models for distributed software systems	375
15.2	Special characteristics of distributed systems	377
15.3	Distributed processes and distributed IPC	377
	15.3.1 Distributed message passing	
15.4	Synchronous (blocking) and asynchronous communication	380
15.5	Distributed programming paradigms	381
15.6	Remote procedure call (RPC)	381
	15.6.1 An RPC system 15.6.2 The RPC protocol with network or server congestion	
	15.6.3 The RPC protocol with node crash and restart	
	15.6.4 An example: CCLU RPC call semantics 15.6.5 RPC and the ISO reference model	
15.7	RPC–language integration	386
	15.7.1 Distribution transparency 15.7.2 Argument marshalling 15.7.3 Object-oriented systems	
	15.7.4 Type checking and consistency checking	
	15.7.5 Data representation for a heterogeneous environment	
15.8	Critique of RPC	392
15.9	Naming, location and binding	393
	15.9.1 Naming the objects used in IPC 15.9.2 Locating named objects	
	15.9.3 The ANSA approach to distributed programming	
15.10	Operating system support for local and remote IPC	398

15.11	Summary of Part II	399
	Exercises	399
<hr/>		
PART III	CONCURRENT COMPOSITE ACTIONS	401
16	Decomposable abstract operations	403
16.1	Composite operations	403
16.2	Composite operations in main memory	404
16.3	Composite operations involving main memory and persistent memory	405
	16.3.1 Examples from operating systems 16.3.2 An example from a database system	
16.4	Concurrent execution of composite operations	407
	16.4.1 Desirability of concurrent execution	
16.5	Potential problems	408
	16.5.1 Uncontrolled interleaving of sub-operations	
	16.5.2 Visibility of the effects of sub-operations 16.5.3 Deadlock	
16.6	Crashes	410
16.7	Summary	412
		412
17	Resource allocation and deadlock	415
17.1	Requirements for dynamic allocation	415
17.2	Deadlock	416
17.3	Livelock and starvation	417
17.4	Conditions for deadlock to exist	418
	17.4.1 Deadlock prevention	
17.5	The dining philosophers problem	420
17.6	Object allocation graphs	422
17.7	Data structures and algorithms for deadlock detection	423
	17.7.1 An algorithm for deadlock detection 17.7.2 Example	
	17.7.3 Action on detection of deadlock	
17.8	Deadlock avoidance	426
	17.8.1 Problems of deadlock avoidance	
17.9	Information on releasing resources – multiphase processes	428
17.10	Distributed deadlocks	429
	17.10.1 Distributed deadlock detection	
17.11	Summary	432
	Exercises	432

18	Transactions	435
18.1	Introduction	435
18.2	Transactions	437
	18.2.1 Commit and abort 18.2.2 Notation for transactions	
18.3	Serializability and consistency	439
18.4	The ACID properties of transactions	440
18.5	Indicating specific orderings of transactions	441
18.6	A system model for transaction processing	441
	18.6.1 Non-commutative (conflicting) pairs of operations 18.6.2 Condition for serializability	
18.7	Dependency graphs for transactions	445
18.8	Histories and serialization graphs	448
18.9	Dealing with aborts: More about the property of isolation	450
	18.9.1 Cascading aborts 18.9.2 The ability to recover state	
18.10	Summary	454
	Exercises	454
19	Concurrency control	457
19.1	Introduction	457
19.2	Concurrent composite operations in main memory only	458
	19.2.1 Objects in the main memory of a single computer	
	19.2.2 Objects in main memory in a distributed system	
	19.2.3 Systematic approaches to concurrent program development	
19.3	Structure of transaction management systems	461
19.4	Concurrency control through locking	462
	19.4.1 Two-phase locking (2PL) 19.4.2 An example of two-phase locking	
	19.4.3 Semantic locking 19.4.4 Deadlock in two-phase locking	
19.5	Time-stamp ordering (TSO)	466
	19.5.1 Cascading aborts and recovery of state	
19.6	Optimistic concurrency control (OCC)	469
19.7	Summary	477
	Exercises	479
20	Recovery	481
20.1	Requirements for recovery	481
20.2	The object model, object state and recovery	482
20.3	Concurrency, crashes and the properties of transactions	483

20.4	Logging and shadowing for crash resilience	484
20.5	Use of a recovery log	485
	20.5.1 Log records and their use in recovery	
	20.5.2 Log write-ahead	
	20.5.3 Checkpoints and the checkpoint procedure	
20.6	Idempotent undo and redo operations	487
20.7	Transaction states on a failure	488
20.8	An algorithm for recovery	489
20.9	Location databases for mobile objects	490
20.10	Summary	491
	Exercises	492
21	Distributed transactions	495
21.1	An object model for distributed systems	495
21.2	Distributed transaction processing	496
21.3	Communication	498
21.4	Concurrency control: Two-phase locking (2PL)	498
21.5	Concurrency control: Time-stamp ordering (TSO)	499
21.6	Optimistic concurrency control (OCC)	500
21.7	Commit and abort in a distributed system	501
21.8	Atomic commitment: the two-phase commit protocol (2PC)	502
21.9	Two-phase validation for OCC	505
21.10	Summary	507
	Exercises	508
22	Distributed computations	511
22.1	Introduction	511
22.2	Process groups	512
	22.2.1 Leadership election	
22.3	Consistency of data replicas	515
	22.3.1 Quorum assembly for strong consistency	
	22.3.2 Large-scale systems	
22.4	Ordering message delivery	518
	22.4.1 Vector clocks	
22.5	Distributed, <i>N</i> -process mutual exclusion	520
	22.5.1 Algorithms	
22.6	Summary of Part III	524
	Exercises	525

PART IV	SYSTEM CASE STUDIES	527
23	UNIX	529
23.1	Introduction	529
23.2	Evolution of UNIX	530
23.3	A summary of UNIX design features	531
23.4	Overview of kernel modules	531
23.5	The file system interface	532
	23.5.1 Graph navigation and the current directory 23.5.2 An aside: the readable password file	
23.6	The file system implementation	534
	23.6.1 Mounting and unmounting filing systems 23.6.2 Access protection	
	23.6.3 Allocation of disk blocks to a filing system	
	23.6.4 File system data structures in main memory 23.6.5 Consistency issues	
23.7	The execution environment of a process	542
23.8	Process creation and termination	543
	23.8.1 Loading and execution 23.8.2 Parent-child synchronization on process termination	
	23.8.3 Process creation during system initialization	
	23.8.4 Process creation by the command interpreter	
23.9	IPC: Pipelines of processes	547
	23.9.1 Redirection of standard input and output 23.9.2 Pipes	
	23.9.3 Named pipes in later systems 23.9.4 Pipes between commands	
23.10	IPC: Signals, sleep and wakeup	550
	23.10.1 Signals 23.10.2 Sleep and wakeup 23.10.3 IPC: Summary	
23.11	Aspects of I/O implementation	554
	23.11.1 The disk block cache 23.11.2 Device drivers	
	23.11.3 Low-level exception and interrupt handling	
23.12	Execution of the system by processes	556
23.13	Process scheduling and swapping	558
	23.13.1 Process scheduling 23.13.2 The swapping algorithm	
23.14	Process states and transitions	559
23.15	Discussion of basic UNIX, Edition 7	561
23.16	UNIX BSD 4.3	562
	23.16.1 Memory management 23.16.2 Sockets	
23.17	UNIX System V.4	567
	23.17.1 Messages 23.17.2 Shared memory	
	23.17.3 Semaphores for shared user-level memory	
	23.17.4 Discussion of System V IPC 23.17.5 Streams for networking in System V	
23.18	The POSIX standard	570
23.19	Summary	571
	Exercises	571

24	Microkernels: Mach and CHORUS	575
24.1	The evolution from RIG through Accent to Mach	576
	24.1.1 Accent	
24.2	Mach	580
	24.2.1 The Mach basic abstractions 24.2.2 Mach IPC	
	24.2.3 Memory objects and object sharing in main memory	
	24.2.4 Task and thread creation 24.2.5 The C threads package 24.2.6 Real-time extensions	
24.3	CHORUS	585
	24.3.1 The CHORUS V3 basic abstractions 24.3.2 CHORUS IPC	
	24.3.3 CHORUS system structure 24.3.4 CHORUS memory management	
24.4	Summary	590
	Exercises	591
25	Windows NT	593
25.1	Introduction to Windows NT (New Technology)	593
	25.1.1 Requirements 25.1.2 Design goals 25.1.3 Models used in the NT design	
25.2	NT structure	596
	25.2.1 Executive components 25.2.2 Getting started	
25.3	The NT object model and object manager	599
	25.3.1 Overview of NT objects 25.3.2 How NT objects are used	
	25.3.3 The structure of an object 25.3.4 Object types 25.3.5 Object names and directories	
	25.3.6 Object handles 25.3.7 Type-specific object methods 25.3.8 Protecting objects	
25.4	Processes, threads and concurrency control	607
	25.4.1 Thread synchronization 25.4.2 Alerts and asynchronous procedure calls	
25.5	The kernel	610
	25.5.1 Kernel objects, kernel process and thread objects 25.5.2 Thread scheduling and dispatching	
	25.5.3 Interrupt and exception handling	
25.6	Memory management	613
	25.6.1 Sharing memory: Sections, views and mapped files 25.6.2 Memory protection	
25.7	I/O	615
	25.7.1 I/O design features 25.7.2 I/O processing	
25.8	The NT filing system, NTFS	619
	25.8.1 NTFS requirements 25.8.2 New features of NTFS 25.8.3 NTFS design outline	
25.9	Networking	622
25.10	Summary	624
26	Middleware: CORBA and Java	625
26.1	Introduction	625
26.2	OMG and OMA	628
26.3	The OMG Object Model	630

26.4	CORBA	631
	26.4.1 The ORB core 26.4.2 OMG's Interface Definition Language (IDL)	
	26.4.3 Other CORBA features	
26.5	OMG developments	635
26.6	ODMG (Object Database Management Group)	636
26.7	Java: Introduction	636
26.8	The Java programming language and environment	638
	26.8.1 Concurrent programming in Java	
26.9	The Java platform and components (JavaBeans)	641
26.10	Java developments	642
26.11	Ubiquitous computing	642
26.12	General mobile applications	643
26.13	Scripting languages	645
26.14	Summary	646
27	Transaction processing monitors and systems	647
27.1	Transaction processing monitors	648
	27.1.1 Use of processes and IPC 27.1.2 Buffered transaction requests	
	27.1.3 Monitoring system load and performance	
27.2	Introduction to some electronic funds transfer (EFT) applications	653
	27.2.1 Paying by cheque 27.2.2 Paying by credit card	
	27.2.3 Paying by debit card: Point of sale transactions 27.2.4 Some security issues	
27.3	International inter-bank payments: S.W.I.F.T.	657
27.4	Authentication by PIN	658
27.5	The international automatic teller machine (ATM) network service	661
	27.5.1 How bank accounts are held 27.5.2 Local control of ATMs 27.5.3 Remote use	
27.6	Load and traffic in TP systems	664
27.7	Summary and trends	665
	Exercises	666
Appendix:	Evolution of computer systems	669
A.1	Introduction, technological and commercial context	669
A.2	Operating systems and distributed operating systems	674
	A.2.1 Multiprogramming batch A.2.2 Centralized time-sharing (interactive) systems	
	A.2.3 Workstations and personal computers	
A.3	Middleware, distributed services and ubiquitous computing	680
A.4	Databases	681

<i>Bibliography</i>	683
<i>Glossary</i>	699
<i>Author index</i>	703
<i>Subject index</i>	707

Trademark notice

The following are trademarks or registered trademarks of the company given in parentheses:

ARM and CHORUS (Acorn Computers)

ANSA (ANSA Internet Inc.)

OpenDoc (Apple Computers)

ACTIVEX.COM (used by CNET under license from owner – ACTIVEX.COM is an independent online service)

ULTRIX (Digital Equipment Corporation)

UP-UX (Hewlett-Packard Company)

Occam (INMOS Group of companies)

OS/2 and System 370 (International Business Machines Corporation)

Windows NT, Windows 95, Visual Basic, MS-DOS and ActiveX (Microsoft Corporation)

Tina-C (National Semiconductor Corporation)

Netscape Navigator (Netscape Communications Corporation)

CORBA (Object Management Group)

REXX (Oracle Corporation)

Seagate Elite 23 (Seagate Technology Products)

MIPS (Silicon Graphics, Inc.)

Java, JavaScript, NFS and SunOS (Sun Microsystems, Inc.)

Amoeba (Vrije Universiteit)

UNIX (X/Open Company Ltd)