



# Contents

---

<b>PART ONE</b>		
<b>Introduction to Solaris Internals</b>		<b>1</b>
<b>Chapter 1</b>	<b>Introduction</b>	<b>3</b>
	1.1 Key Features of Solaris 10, Solaris 9, and Solaris 8	4
	1.1.1 Solaris 10	5
	1.1.2 Solaris 9	8
	1.1.3 Solaris 8	10
	1.2 Key Differentiators	12
	1.3 Kernel Overview	15
	1.3.1 Solaris Kernel Architecture	16
	1.3.2 Modular Implementation	17
	1.4 Processes, Threads, and Scheduling	19
	1.4.1 A New Threads Model	20
	1.4.2 Global Process Priorities and Scheduling	22
	1.5 Interprocess Communication	24
	1.5.1 Traditional UNIX IPC	24
	1.5.2 System V IPC	25
	1.5.3 POSIX IPC	26
	1.5.4 Solaris Doors: Advanced Solaris IPC	26

---

1.6	Signals	26
1.7	Memory Management	27
1.7.1	Global Memory Allocation	28
1.7.2	The Cyclic Page Cache	29
1.7.3	Kernel Memory Management	29
1.8	Files and File Systems	30
1.9	Resource Management	32
1.9.1	Processor Controls and Domains	33
1.9.2	Solaris Resource Management	35
1.9.3	Internet Protocol Quality of Service	39
1.9.4	Resource Management and Observability	39
 <b>PART TWO</b>		
<b>The Process Model</b>		<b>41</b>
<b>Chapter 2</b>	<b>The Solaris Process Model</b>	<b>43</b>
2.1	Components of a Process	44
2.1.1	Thread Objects	44
2.1.2	Core Process Components	47
2.2	Process Model Evolution	48
2.2.1	Thread Model Evolution	49
2.2.2	Unified Process Model	50
2.3	Executable Objects	52
2.4	Process Structures	56
2.4.1	The <code>proc</code> Structure	56
2.4.2	User Area	68
2.4.3	Lightweight Processes (LWPs)	71
2.4.4	Kernel Threads	76
2.5	Kernel Process Table	82
2.5.1	Process Limits	84
2.5.2	Thread Limits	87
2.6	Process Resource Attributes	88
2.7	Process Creation	92
2.8	System Calls	102
2.8.1	System Calls on SPARC Architectures	103
2.8.2	A Tour Through a System Call	106

---

2.9	Process Termination	111
2.9.1	LWP and Kernel Thread Exit	113
2.9.2	Deathrow List	114
2.10	The Process File System	115
2.10.1	Procfs Implementation	118
2.10.2	Process Resource Usage	129
2.10.3	Microstate Accounting	130
2.11	Signals	134
2.11.1	Signals Implementation	140
2.11.2	Observing Signal Activity	155
2.11.3	Summary	156
2.12	Sessions and Process Groups	157
2.13	MDB Reference	164
<b>Chapter 3</b>	<b>Scheduling Classes and the Dispatcher</b>	<b>165</b>
3.1	Fundamentals	166
3.2	Processor Abstractions	170
3.2.1	Processor Observability	177
3.3	Dispatcher Queues, Structures, and Variables	181
3.3.1	Dispatcher Structures	182
3.3.2	Dispatcher Structure Linkage	187
3.3.3	Examining Dispatcher Structures	189
3.4	Dispatcher Locks	197
3.4.1	Dispatcher Lock Functions	200
3.4.2	Thread Locks	201
3.4.3	Thread Lock Functions	202
3.4.4	Lock Statistics	203
3.5	Dispatcher Initialization	204
3.6	Scheduling Classes	206
3.6.1	Scheduling Class Data	208
3.6.2	Scheduling Class Functions	213
3.6.3	Scheduling Class Dispatcher Tables	217
3.7	Thread Priorities	222
3.7.1	Global Priorities	223
3.7.2	User Priorities	224
3.7.3	Setting Thread Priorities	226

---

3.8	Dispatcher Functions	252
3.8.1	Dispatcher Queue Management	253
3.8.2	The Heart of the Dispatcher: <code>swtch()</code>	261
3.9	Preemption	265
3.10	The Kernel Sleep/Wakeup Facility	273
3.10.1	Condition Variables	274
3.10.2	Sleep Queues	276
3.10.3	The Sleep Process	278
3.10.4	The Wakeup Mechanism	282
3.11	Interrupts	283
3.11.1	Interrupt Priorities	285
3.11.2	Interrupts as Threads	286
3.11.3	Interrupt Thread Priorities	287
3.11.4	High-Priority Interrupts	287
3.11.5	Interrupt Management	288
3.11.6	Interrupt Monitoring	288
3.11.7	Interprocessor Interrupts and Cross-Calls	289
3.12	Summary	291
3.13	MDB Reference	292
<b>Chapter 4</b>	<b>Interprocess Communication</b>	<b>293</b>
4.1	The System V IPC Framework	294
4.1.1	Objects	294
4.1.2	IPC Framework Design	295
4.1.3	Locking	297
4.1.4	Module Creation	301
4.2	System V IPC Resource Controls	302
4.2.1	The Solution	303
4.3	Configuring IPC Tuneables on Solaris 10	306
4.4	System V Shared Memory	307
4.4.1	Shared Memory Kernel Implementation	309
4.4.2	Intimate Shared Memory (ISM)	312
4.4.3	Dynamic ISM Shared Memory	315
4.5	System V Semaphores	316
4.5.1	Semaphore Kernel Resources	317
4.5.2	Kernel Implementation of System V Semaphores	318

---

4.5.3	Semaphore Operations	319
4.6	System V Message Queues	320
4.6.1	Kernel Resources for Message Queues	321
4.6.2	Kernel Implementation of Message Queues	323
4.7	POSIX IPC	325
4.7.1	POSIX Shared Memory	327
4.7.2	POSIX Semaphores	328
4.7.3	POSIX Message Queues	331
4.8	Solaris Doors	335
4.8.1	Doors Overview	336
4.8.2	Doors Implementation	337
4.9	MDB Reference	344
<b>Chapter 5</b>	<b>Process Rights Management</b>	<b>345</b>
5.1	Then and Now	345
5.2	Least Privilege in Solaris	346
5.3	Process Privileges Models	347
5.3.1	The Traditional Solaris Superuser Model	348
5.3.2	Extending Solaris With Process Privileges	349
5.3.3	How the Solaris 10 Least Privilege Model Was Chosen	350
5.3.4	Other Unix Implementations	353
5.4	Privilege Awareness: The Details	356
5.4.1	Per-Process State	356
5.4.2	Privilege Awareness State Transitions	357
5.4.3	Privilege State Manipulation	358
5.4.4	Privilege Escalation Prevention	362
5.4.5	The Trouble With uid 0	363
5.4.6	Basic Privileges	364
5.4.7	Privileges and the Runtime Environment	365
5.4.8	Privileges and NFS	366
5.4.9	Privileges and Third-Party File Systems	367
5.5	Least Privilege Interfaces	367
5.5.1	The Conspiracy of Bit Sets and Constants	367
5.5.2	Privilege Names and Constants	368
5.5.3	Kernel Data Structures	369
5.5.4	Kernel Interfaces	372

5.5.5	System Call Interfaces	374
5.5.6	Library Interfaces	377
5.5.7	Using Privileges with Role-Based Access Control	381
5.5.8	Using Privileges with Role-Based Access Control	383
5.5.9	Using DTrace for Tracking Privileges	385
5.5.10	Enhancements to <code>proc(4)</code> and Core Dumps	385
5.5.11	Privilege Debugging	386
5.5.12	Privilege Auditing	387
5.5.13	Device Protection	388

## **PART THREE**

### **Resource Management 391**

<b>Chapter 6</b>	<b>Zones</b>	<b>393</b>
6.1	Introduction	393
6.1.1	Zone Basics	395
6.1.2	Zone Principles	396
6.2	Zone Runtime	397
6.2.1	Zone State Model	397
6.2.2	Zone Names and Numeric IDs	399
6.2.3	Zone Runtime Support	400
6.2.4	Listing Zone Information	401
6.3	Booting Zones	402
6.4	Security	407
6.4.1	Credential Handling	408
6.4.2	Fine-Grained Privileges	408
6.4.3	Role-Based Access Control	413
6.4.4	Chroot Interactions	413
6.5	Process Model	414
6.5.1	Signals and Process Control	414
6.5.2	Global Zone Visibility and Access	415
6.5.3	<code>/proc</code>	416
6.5.4	Core Files	417
6.6	File Systems	417
6.6.1	Configuration	418
6.6.2	Size Restrictions	419

---

6.6.3	File System-Specific Issues	419
6.6.4	File System Traversal Issues	421
6.7	Networking	422
6.7.1	Partitioning	423
6.7.2	Interfaces	424
6.7.3	IPv6	425
6.7.4	IPsec	426
6.7.5	Raw IP Socket Access	426
6.7.6	DLPI Access	427
6.7.7	Routing	427
6.7.8	TCP Connection Teardown	427
6.8	Devices	427
6.8.1	Device Categories	428
6.8.2	/dev and /devices Namespace	430
6.8.3	Device Management: Zone Configuration	430
6.8.4	Device Management: Zone Runtime	430
6.8.5	Zone Console Design	432
6.8.6	ftpd	434
6.9	Interprocess Communication	434
6.9.1	Pipes, STREAMS, and Sockets	434
6.9.2	Doors	435
6.9.3	Loopback Transport Providers	435
6.9.4	System V IPC	436
6.9.5	POSIX IPC	437
6.10	Resource Management and Observability	437
6.10.1	Performance	439
6.10.2	Solaris Resource Management Interactions	440
6.10.3	kstats	442
6.11	MDB Reference	444
<b>Chapter 7</b>	<b>Projects, Tasks, and Resource Controls</b>	<b>445</b>
7.1	Projects and Tasks Framework	445
7.1.1	Introduction	445
7.1.2	Projects	446
7.1.3	Tasks	446
7.1.4	Why We Added Tasks to Solaris	447

7.2	The Project Database	448
7.3	Project and Task APIs	449
7.3.1	Interfaces for Projects and Tasks	449
7.4	Kernel Infrastructure for Projects and Tasks	450
7.4.1	System Call Interaction with Projects	451
7.4.2	proc(4)	452
7.4.3	In-Kernel Project Data Structures	452
7.5	Resource Controls	454
7.5.1	Introduction to Resource Controls	454
7.5.2	What Is an rctl?	455
7.5.3	Numeric Values of Resource Controls	456
7.5.4	Resource Control Definitions	456
7.5.5	Policy	459
7.5.6	Consequences of Exceeding an rctl	460
7.5.7	Signal and siginfo Semantics for Exceeded Controls	461
7.5.8	Generalizing Hard and soft limits	462
7.5.9	Resource Controls and the Task	462
7.5.10	Visibility Through /proc; Privileges and Ownership	463
7.1	Interfaces for Resource Controls	463
7.5.11	Project Name-Service Attributes	464
7.5.12	Attributes Originating Within Solaris	464
7.5.13	Grammar for Attributes	464
7.5.14	Interpretation of rctl Attributes	465
7.5.15	An Example /etc/project	467
7.5.16	System Calls and Private Kernel Interfaces	467
7.5.17	Library Functions	468
7.2	Kernel Interfaces for Resource Controls	469
7.5.18	Data Structures	470
7.5.19	Operations Vector	472
7.5.20	Interface Overview	472
7.5.21	Interface Definitions	474
7.5.22	An Example Resource Control	475

## PART FOUR

### Memory

479

---

<b>Chapter 8</b>	<b>Introduction to Solaris Memory</b>	<b>481</b>
8.1	Virtual Memory Primer	481
8.2	Two Levels of Memory	482
8.3	Memory Sharing and Protection	482
8.4	Pages: Basic Units of Physical Memory	482
8.5	Virtual-to-Physical Translation	483
8.6	Physical Memory Management: Paging and Swapping	484
8.7	Virtual Memory As a File System Cache	484
8.8	New Features of the Virtual Memory Implementation	485
<b>Chapter 9</b>	<b>Virtual Memory</b>	<b>489</b>
9.1	Design Overview	489
9.2	Virtual Address Spaces	490
9.2.1	Sharing Executables and Libraries	492
9.2.2	Address Spaces on SPARC Systems	493
9.2.3	x86 and x64 Address Space Layout	495
9.2.4	Growing the Heap	496
9.2.5	The Stack	498
9.2.6	Using <code>pmap</code> to Look at Mappings	500
9.3	Tracing the VM System	501
9.4	Virtual Address Space Management	502
9.4.1	Address Space Management	503
9.4.2	Address Space Callbacks	507
9.4.3	Virtual Memory Protection Modes	508
9.4.4	Page Faults in Address Spaces	508
9.5	Segment Drivers	512
9.5.1	The <code>vnode</code> Segment: <code>seg_vn</code>	516
9.5.2	Copy-on-Write	519
9.5.3	Page Protection and Advice	520
9.6	Anonymous Memory	520
9.7	The Anonymous Memory Layer	521
9.8	The <code>swapfs</code> Layer	524
9.8.1	<code>swapfs</code> Implementation	525
9.9	Virtual Memory Watchpoints	527
9.10	Changes to Support Large Pages	530

---

9.10.1	System View of a Large Page	531
9.10.2	Free List Organization	531
9.10.3	Large-Page Faulting	532
9.10.4	Large-Page Freeing	535
9.10.5	Operations That Interfere with Large Pages	536
9.10.6	HAT Support	536
9.10.7	procfs Changes	537
9.11	MDB Reference	538
<b>Chapter 10</b>	<b>Physical Memory</b>	<b>539</b>
10.1	Physical Memory Allocation	539
10.1.1	The Allocation Cycle of Physical Memory	539
10.2	Pages—The Basic Unit of Solaris Memory	542
10.2.1	The Page Hash List	543
10.2.2	Page Structures	545
10.2.3	Free List and Cache List	546
10.2.4	Physical Page "memseg" Lists	546
10.2.5	The Page-Level Interfaces	547
10.2.6	The Page Throttle	549
10.2.7	Page Coloring	549
10.3	The Page Scanner	554
10.3.1	Page Scanner Operation	554
10.3.2	Page-out Algorithm and Parameters	555
10.3.3	Shared Library Optimizations	558
10.3.4	Parameters That Limit Pages Paged Out	559
10.3.5	Page Scanner Implementation	560
10.3.6	The Memory Scheduler	562
10.4	MDB Reference	564
<b>Chapter 11</b>	<b>Kernel Memory</b>	<b>565</b>
11.1	Kernel Virtual Memory Layout	565
11.1.1	Kernel Address Space	566
11.1.2	Kernel Text and Data Segments	568
11.1.3	Virtual Memory Data Structures	568
11.1.4	UltraSPARC Kernel Nucleus	568
11.1.5	Loadable Kernel Module Text and Data	569

---

11.1.6	The Kernel Address Space and Segments	571
11.2	Kernel Memory Allocation	572
11.2.1	The Kernel Heap	573
11.2.2	The Kernel Memory Segment Driver	574
11.2.3	The Kernel Memory Slab Allocator	576
11.3	The Vmem Allocator	593
11.3.1	Background	593
11.3.2	Vmem Objectives	594
11.3.3	Interface Description	594
11.3.4	Vmem Implementation	598
11.3.5	Vmem Performance	602
11.3.6	Summary	603
11.4	Kernel Memory Allocator Tracing	603
11.4.1	Enabling KMA DEBUG Flags	604
11.4.2	Examining Kernel Memory Allocations with MDB	605
11.4.3	Detecting Memory Corruption	607
11.4.4	Checking a Freed Buffer: 0xdeadbeef	608
11.4.5	Debugging With the Redzone Indicator: 0xfeedface	608
11.4.6	Detecting Uninitialized Data: 0xbaddcafe	611
11.4.7	Associating Panic Messages With Failures	612
11.4.8	Memory Allocation Logging	612
11.4.9	Analyzing Memory With Advanced Techniques	615
11.4.10	Finding Corrupt Buffers With ::kmem_verify	617
11.4.11	Using the Allocator Logging Facility	618
11.5	MDB Reference	621
<b>Chapter 12</b>	<b>Hardware Address Translation</b>	<b>623</b>
12.1	HAT Overview	623
12.2	The UltraSPARC HAT Layer	625
12.2.1	Introduction	625
12.2.2	struct hat	628
12.2.3	The Translation Table	631
12.2.4	The Translation Storage Buffer (TSB)	646
12.2.5	Intimate Shared Memory (ISM)	657
12.2.6	Synchronization in the HAT Layer	661
12.2.7	SPARC HAT Layer Kernel Tunables	666

---

12.2.8	SPARC Hat Layer kstats	667
12.3	The x64 HAT Layer	672
12.3.1	MMU Configuration	672
12.3.2	struct mmu Variable	674
12.3.3	Virtual Address Space Layout	675
12.3.4	64-bit Address Space Layout	677
12.3.5	32-bit Address Space Layout	678
12.3.6	HAT Implementation	679
12.4	MDB Reference	684
<b>Chapter 13</b>	<b>Working with Multiple Page Sizes in Solaris</b>	<b>685</b>
13.1	Determining When to Use Large Pages	685
13.2	Measuring Application Performance	686
13.2.1	Determination Allocated Page Sizes	688
13.2.2	Discovery of Supported Page Sizes	691
13.3	Configuring for Multiple Page Sizes	692
13.3.1	Enabling Large Pages	692
13.3.2	Advising Page-size Preferences With <code>ppgsz(1M)</code>	693
13.3.3	Interposing Shared Libraries With <code>libmpss.so</code>	693
13.3.4	Request Larger Page Sizes with the Compiler	695
13.3.5	Interfaces to Request Larger Page Sizes	696
13.3.6	CPU Specific Large Page Support	699
<b>PART FIVE</b>		
<b>File Systems</b>		<b>703</b>
<b>Chapter 14</b>	<b>File System Framework</b>	<b>705</b>
14.1	File System Framework	705
14.2	Process-Level File Abstractions	707
14.2.1	File Descriptors	708
14.2.2	The <code>open</code> Code Path	709
14.2.3	Allocating and Deallocating File Descriptors	710
14.2.4	File Descriptor Limits	713
14.2.5	File Structures	714
14.3	Solaris File System Framework	716
14.3.1	Evolution of the File System Framework	718

---

14.3.2	The Solaris File System Interface	721
14.4	File System Modules	721
14.4.1	Interfaces for Mount Options	722
14.4.2	Module Initialization	724
14.5	The Virtual File System (vfs) Interface	725
14.5.1	vfs Methods	725
14.5.2	vfs Support Functions	729
14.5.3	The mount Method	733
14.5.4	The umount Method	734
14.5.5	Root vnode Identification	735
14.5.6	vfs Information Available with MDB	736
14.6	The Vnode	738
14.6.1	Object Interface	739
14.6.2	vnode Types	741
14.6.3	vnode Method Registration	742
14.6.4	vnode Methods	744
14.6.5	Support Functions for Vnodes	751
14.6.6	The Life Cycle of a Vnode	752
14.6.7	vnode Creation and Destruction	753
14.6.8	The vnode Reference Count	753
14.6.9	Interfaces for Paging vnode Cache	754
14.6.10	Block I/O on vnode Pages	756
14.6.11	vnode Information Obtainable with mdb	757
14.6.12	DTrace Probes in the vnode Layer	759
14.7	File System I/O	764
14.7.1	Memory Mapped I/O	764
14.7.2	read() and write() System Calls	766
14.7.3	The seg_kpm Driver	767
14.7.4	The seg_map Driver	768
14.7.5	Interaction between segmap and segkpm	773
14.8	File Systems and Memory Allocation	775
14.8.1	Solaris 8 — Cyclic Page Cache	776
14.8.2	The Old Allocation Algorithm	776
14.8.3	The New Allocation Algorithm	778
14.8.4	Putting It All Together: The Allocation Cycle	779

---

14.9	Path-Name Management	780
14.9.1	The lookupn() Method	781
14.9.2	The vop_lookup() Method	781
14.9.3	The vop_readdir() Method	782
14.9.4	Path-Name Traversal Functions	783
14.10	The Directory Name Lookup Cache	785
14.10.1	DNLC Operation	785
14.10.2	Primary DNLC Support Functions	787
14.10.3	DNLC Negative Cache	788
14.10.4	DNLC Directory Cache	789
14.10.5	DNLC Housekeeping Thread	793
14.10.6	DNLC Statistics	793
14.11	The File System Flush Daemon	794
14.12	File System Conversion to Solaris 10	794
14.13	MDB Reference	796
<b>Chapter 15</b>	<b>The UFS File System</b>	<b>797</b>
15.1	UFS Development History	797
15.2	UFS On-Disk Format	799
15.2.1	On-disk UFS Inodes	799
15.2.2	UFS Directories	804
15.2.3	UFS Hard Links	806
15.2.4	Shadow Inodes	806
15.2.5	The Boot Block	807
15.2.6	The Superblock	807
15.2.7	The Cylinder Group	810
15.2.8	Summary of UFS Architecture	811
15.3	The UFS Inode	813
15.3.1	In-core UFS Inodes	813
15.3.2	Inode Cache	814
15.3.3	Block Allocation	817
15.3.4	Methods to Read and Write UFS Files	824
15.4	Access Control in UFS	828
15.5	Extended Attributes in UFS	831
15.6	Locking in UFS	833
15.6.1	UFS Lock Descriptions	833

---

15.6.2	Inode Lock Ordering	838
15.6.3	UFS Lockfs Protocol	838
15.7	Logging	840
15.7.1	On-disk Log Data Structures	841
15.7.2	In-core Log Data Structures	845
15.7.3	Summary Information	849
15.7.4	Transactions	850
15.7.5	Rolling the Log	855
15.7.6	Redirecting Reads and Writes to the Log	857
15.7.7	Failure Recovery	858
15.8	MDB reference	859
 <b>PART SIX</b>		
<b>Platform Specifics</b>		<b>861</b>
<b>Chapter 16</b>	<b>Support for NUMA and CMT Hardware</b>	<b>863</b>
16.1	Memory Hierarchy Designs	864
16.1.1	What Is NUMA?	864
16.1.2	What Is CMT?	865
16.2	Memory Placement Optimization Framework	867
16.2.1	Latency Model	868
16.2.2	More Complex Models	869
16.3	Initial Thread Placement	870
16.4	Scheduling	871
16.5	Memory Allocation	871
16.6	Lgroup Implementation	872
16.6.1	Parameters Affecting MPO	874
16.7	MPO APIs	876
16.7.1	Informational	876
16.7.2	Verifying the Interface Version	879
16.7.3	Initialization of the Locality Group Interface	879
16.8	Locality Group Hierarchy	880
16.8.1	Locality Group Characteristics	881
16.8.2	Locality Groups and Thread and Memory Placement	881
16.9	MPO Statistics	882
16.10	MDB Reference	883

---

<b>Chapter 17</b>	<b>Locking and Synchronization</b>	<b>885</b>
17.1	Synchronization	885
17.2	Parallel Systems Architectures	886
17.3	Hardware Considerations for Locks and Synchronization	889
17.4	Introduction to Synchronization Objects	894
17.4.1	Synchronization Process	895
17.4.2	Synchronization Object Operations Vector	897
17.5	Mutex Locks	898
17.5.1	Overview	898
17.5.2	Solaris Mutex Lock Implementation	901
17.6	Reader/Writer Locks	906
17.6.1	Solaris Reader/Writer Locks	907
17.7	Turnstiles and Priority Inheritance	911
17.7.1	Turnstiles Implementation	912
17.8	Kernel Semaphores	914
17.9	DTrace Lockstat Provider	917
17.9.1	Overview	917
17.9.2	Adaptive Lock Probes	918
17.9.3	Spin Lock Probes	919
17.9.4	Thread Locks	920
17.9.5	Readers/Writer Lock Probes	920
 <b>PART SEVEN</b>		
<b>Networking</b>		<b>923</b>
<b>Chapter 18</b>	<b>The Solaris Network Stack</b>	<b>925</b>
18.1	STREAMS and the Network Stack	925
18.1.1	The STREAMS Model	926
18.1.2	Network Stack as STREAMS Module	929
18.1.3	Issues with STREAMS-based Stacks	932
18.2	Solaris 10 Stack: Design Goals	933
18.3	Solaris 10 Network Stack Framework	934
18.3.1	Vertical Perimeter	935
18.3.2	IP Classifier	939
18.3.3	Synchronization Mechanism	940

---

18.4	TCP as an Implementation of the New Framework	941
18.4.1	The Interface Between TCP and IP	943
18.4.2	TCP Loopback	945
18.5	UDP	946
18.5.1	UDP Packet Drop Within the Stack	947
18.5.2	UDP Module	947
18.5.3	UDP and Socket Interaction	949
18.6	Synchronous STREAMS	949
18.6.1	TCP Synchronous STREAMS	949
18.6.2	STREAMS Fallback	950
18.7	IP	951
18.7.1	Plumbing NICs	952
18.7.2	IP Network Multipathing	952
18.7.3	Multicast	953
18.8	Solaris Device Driver Framework	953
18.8.1	GLDv2 and DLPI Drivers (Solaris 9 and prior)	953
18.8.2	A New Architecture: GLDv3	954
18.8.3	GLDv3 Link Aggregation Architecture	961
18.8.4	Checksum Offload	963
18.9	Interrupt Model and NIC Speeds	964
18.9.1	Solaris 9 and Earlier Releases	964
18.9.2	Dynamic Switch Between Interrupt vs. Polling Mode	965
18.9.3	Interrupt Load Spreading	967
18.10	Summary	968
18.11	MDB Reference	969
<b>PART EIGHT</b>		
<b>Kernel Services</b>		<b>973</b>
<b>Chapter 19</b>	<b>Clocks and Timers</b>	<b>975</b>
19.1	The System Clock Thread	975
19.1.1	Thread Tick Processing	977
19.1.2	DTrace Providers for Tick Processing	978
19.2	Callouts and Callout Tables	979
19.3	System Time Facilities	984
19.3.1	High-Resolution Timer	984

---

19.3.2	Time-of-Day Clock	985
19.4	The Cyclic Subsystem	985
19.4.1	Cyclic Subsystem Interface Overview	987
19.4.2	Cyclic Subsystem Implementation Overview	988
19.4.3	Clients of the Cyclic Subsystem	997
19.4.4	Cyclic Kernel At-Large Interfaces	998
19.4.5	Cyclic Kernel Inter-Subsystem Interfaces	999
19.4.6	Cyclic Backend Interfaces	999
19.4.7	Cyclic Subsystem Backend-Supplied Interfaces	999
<b>Chapter 20</b>	<b>Task Queues</b>	<b>1001</b>
20.1	Overview of Task Queues	1001
20.2	Dynamic Task Queues	1002
20.2.1	Why a <i>Dynamic</i> Task Queue?	1002
20.2.2	Problems Addressed by Dynamic Task Queues	1003
20.2.3	Task Pool Model	1004
20.2.4	Interface Changes to Support Dynamic Task Queues	1005
20.3	Task Queues Kernel APIs	1007
20.4	Device Driver Interface for Task Queues	1009
20.5	Task Queue Observability	1010
20.5.1	Kstat Counters	1010
20.5.2	DTrace SDT Probes	1011
20.6	Task Queue Implementation Notes	1012
20.6.1	Use of kmem Caches	1012
20.6.2	Use of vmem Arenas	1013
20.6.3	Hashed vmem Arenas	1014
20.6.4	Cached List of Entries	1014
20.6.5	Problems With Task Pool Implementation	1015
20.6.6	Use of Dynamic Task Pools in STREAMS	1016
<b>Chapter 21</b>	<b>kmdb Implementation</b>	<b>1017</b>
21.1	Introduction	1017
21.1.1	MDB Components	1017
21.1.2	Major kmdb Design Decisions	1020
21.1.3	The Structure of kmdb	1024
21.1.4	MDB Components and Their Implementation in kmdb	1026

---

21.1.5 Conclusion	1033
21.1.6 Remaining Components	1034
	1039
<b>Appendix A</b> Kernel Virtual Address Maps	1039
	1047
<b>Appendix B</b> Adding A System Call to Solaris	1047
	1053
<b>Appendix C</b> A Sample Procfs Utility	1053
	1059
<b>Appendix D</b> Bibliography	1059

