Contents

Foreword 1

PART ONE
Introduction to Solaris Internals **1**

Chapter 1 Introduction 3

1.1 The Solaris Release Model 4
1.2 Key Features of Solaris 10, Solaris 9, and Solaris 8 6
1.2.1 Solaris 10 6
1.2.2 Solaris 9 10
1.2.3 Solaris 8 12
1.3 Key Differentiators 14
1.4 Kernel Overview 17
1.4.1 Solaris Kernel Architecture 18
1.4.2 Modular Implementation 19
1.5 Processes, Threads, and Scheduling 22
1.5.1 A New Threads Model 23
1.5.2 Global Process Priorities and Scheduling 25
1.6 Interprocess Communication 27
1.6.1 Traditional UNIX IPC 27

1.6.2 System V IPC 28
1.6.3 POSIX IPC 29
1.6.4 Solaris Doors: Advanced Solaris IPC 30
1.7 Signals 30
1.8 Memory Management 31
1.8.1 Global Memory Allocation 32
1.8.2 The Cyclic Page Cache 33
1.8.3 Kernel Memory Management 33
1.9 Files and File Systems 34
1.9.1 File Descriptors and File System Calls 35
1.9.2 The Virtual File System Framework 35
1.10 Resource Management 38
1.10.1 Processor Controls and Domains 39
1.10.2 Solaris Resource Management 41
1.10.2.1 Solaris 8 Resource Manager 41
1.10.2.2 Solaris 9 Resource Management 42
1.10.2.3 Solaris 10 Resource Management 43
1.10.3 Internet Protocol Quality of Service 45
1.10.4 Resource Management and Observability 45

PART TWO
Tools **47**

Chapter 2 The Modular Debugger 49
2.1 Introduction to the modular debugger 49
2.1.1 MDB Features 50
2.1.2 Terms 51
2.1.3 Using MDB 51
2.2 Debugger Concepts 51
2.2.1 Building Blocks 52
2.2.2 Modularity 54
2.3 Beginners tour 56
2.3.1 Values 56
2.3.2 Expressions 56
2.3.3 dcmds 56
2.3.4 walkers 56

2.3.5 listing and searching for dcmds and walkers 56
2.3.6 Pipes 56
2.4 Advanced Techniques 57
2.4.1 Advanced pipes, ::grep ::eval etc 57
2.5 Debugging Kernel cores 57
2.5.1 Examining Kernel Core Summary Information 58
2.5.2 Examining The Message Buffer 58
2.5.3 Listing Processes 59
2.5.4 Global Memory Summary 59
2.5.5 Listing Network Connections 60
2.5.6 Listing all Kernel Threads 60
2.5.7 Obtaining a Stack Trace of the Running Thread 61
2.5.8 Looking at the Status of a CPU 61
2.5.9 Determining the Process of the Offending Thread 62
2.5.10 Disassembling the Suspect Code 63
2.5.11 Displaying General Purpose Registers 63
2.5.12 Parameter Passing on SPARC 66
2.5.12.1 Stack Format 67
2.5.12.2 A Tutorial: Finding arguments with an NFSv4 bug 67
2.6 Debugging Process cores 71
2.7 Debugging Processes within a Kernel Image 75
2.7.1 Enabling Process Pages in a Dump 76
2.7.2 Invoking MDB to examine the kernel image 76
2.7.3 Locating the Target Process 76
2.7.4 Switching MDB to a Debug a Specific Process 77
2.7.5 Constructing the Process Stack 78
2.7.6 Examine the process memory 80
2.8 Dcmd and Walker Reference 82
2.8.1 Commands 82
2.8.2 Comments 82
2.8.3 Expressions 82
2.8.4 Symbols 83
2.8.5 DCMDs 83
2.8.6 Variables 83
2.8.7 Read formats 83

2.8.8 Write formats 84
2.8.9 Search formats 84
2.8.10 General dcmds 84
2.8.11 Target-related dcmds 84
2.8.12 CTF-related 85
2.8.13 Kernel: proc-related 85
2.8.14 Kernel: thread-related 85
2.8.15 Kernel: synchronization-related 85
2.8.16 Kernel: CPU-related 86
2.8.17 Kernel: memory-related 86
2.8.18 Kernel: kmem-related 86
2.9 KMDB 87
2.9.1 Introduction 87
2.9.1.1 Layout 87
2.9.2 MDB components and their implementation in mdb 88
2.9.2.1 The target layer 88
2.9.2.2 Debugger module management 89
2.9.2.3 Terminal I/O 90
2.9.2.4 Other Stuff 90
2.9.3 Major kmdb design decisions 91
2.9.3.1 The Kernel/Debugger Interface (KDI) 91
2.9.3.2 Implementation as a kernel module 92
2.9.4 The structure of kmdb 94
2.9.5 MDB components and their implementation in kmdb 97
2.9.5.1 The target layer 97
2.9.5.2 Debugger module management 99
2.9.5.3 Module loads and unloads: the work request queue (WR) 99
2.9.5.4 libdl wrapper 101
2.9.5.5 Terminal I/O 101
2.9.5.6 Retrieving the terminal type 102
2.9.5.7 Terminal attributes 102
2.9.5.8 Console I/O 102
2.9.5.9 Interrupt (^C) management 103
2.9.6 Conclusion 104

2.9.7 Remaining components 104
2.9.7.1 The Debugger/PROM Interface (DPI) 104
2.9.7.2 Coordination of debugger entry 105
2.9.7.3 Manipulation of processor state 106
2.9.7.4 Source analysis for execution control 106
2.9.7.5 Management of Breakpoints and Watchpoints 106
2.9.7.6 Trap handling 107
2.10 Diagnosing with kmdb and moddebug 108
2.10.0.7 Starting kmdb from the Console 108
2.10.1 Booting with the Kernel debugger 108
2.10.2 Investigating Hangs 109
2.10.3 Collecting information about panics 110
2.11 A GDB to MDB Reference 112

Chapter 3 Dynamic Tracing 115
3.1 Introduction to DTrace 115
3.2 The Basics 116
3.2.1 D Program Structure 119
3.2.2 Providers and Probes 120
3.2.3 Aggregations 123
3.2.4 Variables 127
3.2.5 Probe Arguments 130
3.2.6 Mixing Providers 133
3.2.7 Accessing Global Kernel Data 135
3.2.8 Assorted Actions of Interest 137
3.3 DTrace Architecture 139
3.4 Summary 146

Chapter 4 Kernel Statistics 147
4.1 C-Level kstat Interface 147
4.1.1 Data Structure Overview 148
4.1.2 Getting Started 151
4.1.3 Data Types 151
4.1.3.1 `KSTAT_TYPE_RAW` 152
4.1.3.2 `KSTAT_TYPE_NAMED` 153
4.1.3.3 `KSTAT_TYPE_TIMER` 154

4.1.3.4 KSTAT_TYPE_INTR 154
4.1.3.5 KSTAT_TYPE_IO 155
4.1.4 kstat Names 157
4.1.5 Functions 158
4.1.6 Management of Chain Updates 158
4.1.7 Putting It All Together 159
4.2 Command-Line Interface 161
4.2.1 The kstat Command 162
4.2.2 Real-World Example That Uses kstat and nawk 163
4.3 Using Perl to Access kstats 166
4.3.1 The Tied-Hash Interface to the kstat Facility 166
4.3.2 The update() Method 167
4.3.3 64-Bit Values 167
4.3.4 Getting Started With Perl 168
4.3.5 netstatMulti Implemented in Perl 169
4.4 Adding Statistics to Solaris 172
4.4.1 A kstat Provider Walkthrough 172
4.4.2 I/O Statistics 176
4.5 Additional Information 179

PART THREE
The Process Model 181

Chapter 5 The Solaris Process Model 183
5.1 *Process Objects* 184
5.2 The Evolution of Solaris Threads 187
5.2.1 Architecture of a Process 188
5.2.2 Process Image 193
5.3 Process Structures 195
5.3.1 The Process Structure 196
5.3.2 The User Area 208
5.3.3 The Lightweight Process (LWP) 213
5.3.4 The Kernel Thread (kthread) 215
5.4 The Kernel Process Table 219
5.4.1 Process Limits 219
5.4.2 LWP Limits 221

5.5 Process Creation 222
5.6 Process Termination 232
5.6.1 The LWP/kthread Model 234
5.6.2 Deathrow 236

Chapter 6 Scheduling Classes and the Dispatcher 237
1.1 Introduction 238
1.2 Processor Abstractions 242
1.3 Dispatcher Queues, Structures & Variables 247
1.3.1 Examining Dispatcher Structures 254
1.4 Dispatcher Locks 262
1.4.1 Dispatcher Lock Functions 264
1.4.2 Thread Locks 265
1.4.3 Thread Lock Functions 266
1.4.4 Lock Statistics 267
1.5 Dispatcher Initialization 268
1.6 Global Priorities 270
1.7 Scheduling Classes 272
1.7.1 Scheduling Class Data 272
1.7.2 Scheduling Class Functions 274
1.7.3 Timeshare Scheduling Class 280
1.7.4 Interactive Scheduling Class 285
1.7.5 Realtime Scheduling Class 286
1.7.6 Fair Share Scheduling Class 288
1.7.7 Fixed Priority Scheduling Class 294
1.8 Dispatcher Functions 296
1.9 Scheduling - CPU and Thread Selection 297
1.10 Preemption 303
1.10.1 System Scheduling Class 309
1.10.1.1 The Heart of the Dispatcher: swtch() 321
1.11 The Kernel Sleep/Wakeup Facility 325
1.11.1 Condition Variables 326
1.11.2 Sleep Queues 328
1.11.3 The Sleep Process 330
1.11.4 The Wakeup Mechanism 334
1.12 Scheduler Activations 335

1.13 Putting It All Together 336

Chapter 7 Interprocess Communication 337

7.1 The System V IPC Framework 338
7.1.1 Objects 338
7.1.2 IPC Framework Design 339
7.1.3 Locking 341
7.1.4 Module Creation 346
7.2 System V IPC Resource Controls 348
7.2.1 The Solution 348
7.3 Configuring IPC Tuneables on Solaris 10 351
7.4 System V Shared Memory 352
7.4.1 Shared Memory Kernel Implementation 354
7.4.2 Intimate Shared Memory (ISM) 357
7.4.3 Dynamic ISM Shared Memory 360
7.4.3.1 DISM Performance 361
7.4.3.2 DISM Implementation 361
7.5 System V Semaphores 361
7.5.1 Semaphore Kernel Resources 362
7.5.2 Kernel Implementation of System V Semaphores 363
7.5.3 Semaphore Operations Inside Solaris 363
7.6 System V Message Queues 365
7.6.1 Kernel Resources for Message Queues 366
7.6.2 Kernel Implementation of Message Queues 367
7.7 POSIX IPC 370
7.7.1 POSIX Shared Memory 372
7.7.2 POSIX Semaphores 373
7.0.1 POSIX Message Queues 376
7.1 Solaris Doors 380
7.1.1 Doors Overview 381
7.1.2 Doors Implementation 382

PART FOUR
Resource Management 389

Chapter 8 Zones 391

8.1 Introduction 391
8.1.1 Barriers to Consolidation 392
8.1.2 The Zones Solution 393
8.1.3 Zone Basics 395
8.1.4 Zone Principles 396
8.1.4.1 Isolation 396
8.1.4.2 Compatibility 397
8.1.4.3 Administration 397
8.1.4.4 Security 397
8.1.4.5 Resource Management 397
8.1.5 Terminology and Conventions 397
8.2 Zone Runtime 398
8.2.1 Zone State Model 398
8.2.2 Zone Names and Numeric IDs 400
8.2.3 Zone Runtime Support 400
8.2.3.1 zoneadmd(1M) 400
8.2.3.2 zsched 401
8.2.4 Listing Zone Information 401
8.3 Zone Administration 402
8.3.1 Zone Configuration 402
8.3.1.1 Configuration Data 403
8.3.2 Zone Installation 404
8.3.3 Virtual Platform Administration 404
8.3.3.1 Readying Zones 405
8.3.3.2 Booting Zones 405
8.3.3.3 Halting Zones 405
8.3.3.4 Rebooting Zones 406
8.3.3.5 Automatic Zone Booting 407
8.3.4 Zone Login 407
8.3.4.1 Zone Console Login 407
8.3.4.2 Interactive and Non-Interactive Modes 408
8.3.4.3 Failsafe Mode 409
8.3.4.4 Remote Login 410
8.3.5 Monitoring and Controlling Zone Processes 410
8.4 Administration within Zones 411

8.4.1 Node Name 411
8.4.2 Name Service Usage within a Zone 412
8.4.3 Default Locale and Timezone 412
8.4.4 Initial Zone Configuration 412
8.4.5 System Log Daemon 413
8.4.6 Commands 413
8.4.7 Internals of Booting Zones 414
8.4.7.1 zinit 414
8.5 Security 414
8.5.1 Credential Handling 415
8.5.2 Fine-Grained Privileges 415
8.5.2.1 Safe Privileges 415
8.5.2.2 Zone Privilege Limits 418
8.5.2.3 Privilege Escalation 419
8.5.3 Role-Based Access Control 420
8.5.4 Chroot Interactions 420
8.6 Process Model 421
8.6.1 Signals and Process Control 422
8.6.2 Global Zone Visibility and Access 422
8.6.3 /proc 423
8.6.4 Core Files 424
8.7 File Systems 424
8.7.1 Configuration 425
8.7.1.1 Zonecfg File System Configuration 425
8.7.2 Size Restrictions 425
8.7.3 File System-Specific Issues 426
8.7.4 File System Traversal Issues 428
8.8 Networking 429
8.8.1 Partitioning 430
8.8.2 Interfaces 431
8.8.3 IPv6 432
8.8.3.1 Address Auto-configuration 432
8.8.3.2 Address Selection 433
8.8.4 IPsec 433
8.8.5 Raw IP Socket Access 433

8.8.6 DLPI Access 434
8.8.7 Routing 434
8.8.8 TCP Connection Teardown 434
8.9 Devices 434
8.9.1 Device Categories 435
8.9.1.1 Unsafe Devices 435
8.9.1.2 Fully-Virtual Devices 436
8.9.1.3 Sharable-Virtual Devices 436
8.9.2 /dev and /devices Namespace 437
8.9.3 Device Management: Zone Configuration 437
8.9.4 Device Management: Zone Runtime 438
8.9.4.1 Read-Only mount of /dev 439
8.9.4.2 Device Privilege Enforcement 439
8.9.5 Zone Console Design 439
8.9.5.1 Pseudo-Terminals 440
8.9.6 ftpd 441
8.10 Inter-Process Communication 442
8.10.1 Pipes, STREAMS, and Sockets 442
8.10.2 Doors 443
8.10.3 Loopback Transport Providers 443
8.10.4 System V IPC 444
8.10.5 POSIX IPC 445
8.11 Resource Management and Observability 445
8.11.1 Performance and Observability 448
8.11.2 Solaris Resource Management Interactions 449
8.11.2.1 Accounting 449
8.11.2.2 Projects, Resource Controls and the Fair Share Scheduler 449
8.11.2.3 Resource Pools 450
8.11.3 kstats 450
8.11.3.1 DTrace 451

Chapter 9 Projects, Tasks, and Accounting 453
9.1 Projects and Tasks Framework 453
9.1.1 Why We Added Tasks to Solaris 455
9.2 Tasks and Projects 455

9.2.1 Solaris Resource Manager (SRM) 456
9.2.2 Implementation of Solaris Process Accounting 456
9.2.3 Interaction Between Projects and Roles 457
9.2.4 Summary 458
9.3 Interfaces for Projects and Tasks 458
9.3.1 New Data Types and System Calls 458
9.3.1.1 Project and Task System Calls 459
9.3.1.2 Accounting System Calls 460
9.3.2 Enhancements to Existing System Calls 460
9.3.3 Enhancements to `proc`(4) 461
9.3.4 Name Services 461
9.3.4.1 LDAP Schema for Project 463
9.3.4.2 Extensions for Resource Management 463
9.3.5 `libexacct` and the `exacct` File Format 463
9.3.5.1 Internal Representation 464
9.3.6 File Representation 465
9.3.7 `exacct` Catalog 466
9.3.8 Additional Library APIs 470
9.3.8.1 Name Service-Related Interfaces: `libc` and `libnsl` 470
9.3.9 New Commands 471
9.3.9.1 `projadd`(1M), `projdel`(1M), and `projmod`(1M) 471
9.3.9.2 `acctadm`(1M) 471
9.3.9.3 `newtask`(1) 471
9.3.9.4 `projects`(1) 472
9.3.10 Enhancements to Existing Commands 472
9.3.10.1 `ps`(1), `pgrep`(1), `pkill`(1) 472
9.3.11 `id`(1) 472
9.3.11.1 `at`(1), `batch`(1), `cron`(1) 473
9.3.11.2 Login Commands 473
9.3.11.3 `su`(1) 473
9.3.12 Solaris Accounting Commands 473
9.4 Further Information About Projects and Tasks 474

Chapter 10 Resource Controls 475
14.1 Introduction to Resource Controls 475

14.1.1 Overview 476
14.2 Why We Added Resource Controls 477
14.2.1 Standard UNIX Implementation of Resource Limits 477
14.2.2 What Is an `rctl`? 477
14.2.2.1 Numeric Values of Resource Controls 478
14.2.2.2 Survey of Constant Definitions 479
14.2.3 Policy 481
14.2.4 Additional Design Issues 482
14.2.4.1 Consequences of Exceeding an `rctl` 482
14.2.4.2 Signal and `siginfo` Semantics for Exceeded Controls 482
14.2.4.3 Syslog Message Format 483
14.2.4.4 Generalizing Hard and soft limits 483
14.2.4.5 Translating rlimits to rctls. 484
14.2.4.6 Resource Controls and the Task 484
14.2.4.7 Visibility Through `/proc`; Privileges and Ownership 485
14.3 Resource Control Administrative Scenarios 485
14.3.1 Resource Controls for Capacity Warnings 485
14.3.2 Controls for Resource Limiting 486
14.3.3 Programming Scenarios 487
14.3.3.1 Simplified Task Creation 487
14.3.3.2 Determining Active Projects 488
14.3.4 Controls in Multiprogramming Scenarios 488
14.4 New Interfaces and Modifications for Resource Controls 489
14.4.1 Project Name-Service Attributes 489
14.4.1.1 Attributes Originating Within Solaris 490
14.4.1.2 Attributes Originating Outside Solaris 490
14.4.1.3 Grammar for Attributes 490
14.4.1.4 Interpretation of rctl Attributes 490
14.4.1.5 An Example /etc/project 492
14.4.2 System Calls and Private Kernel Interfaces 492
14.4.3 Library Functions 492
14.4.3.1 `setproject`(3PROJECT) 492
14.4.3.2 `project_walk`(3PROJECT) 492

14.4.3.3 `rctl_walk`(3C) 493

14.4.3.4 The `getprojent()` Derived Family of Library Calls 493

14.4.3.5 `rctlblk` Manipulation Routines 493

14.4.4 Commands 493

14.4.4.1 `prctl`(1) 493

14.4.4.2 `rctladm`(1M) 493

14.5 Kernel Interfaces for Resource Controls 493

14.5.1 Data Structures 494

14.5.2 Operations Vector 495

14.5.3 Interface Overview 495

14.5.4 Interface Definitions 497

14.5.5 An Example Resource Control 498

PART FIVE

Memory 501

Chapter 11 Introduction to Solaris Memory 503

11.1 Virtual Memory Primer 503

11.2 Two Levels of Memory 504

11.3 Memory Sharing and Protection 504

11.4 Pages: Basic Units of Physical Memory 504

11.5 Virtual-to-Physical Translation 505

11.6 Physical Memory Management: Paging and Swapping 506

11.7 Virtual Memory As a File System Cache 506

11.8 New Features of the Virtual Memory Implementation 507

Chapter 12 Memory Monitoring and Analysis 511

12.1 Learning About the Tools 511

12.1.1 Using Tools for Memory Analysis 511

12.1.2 Explaining the `vmstat` Command 513

12.1.3 Obtaining Memory kstats 516

12.1.4 Using the Perl kstat API to Look at Memory Statistics 517

12.1.5 Using the `memstat` Command in `mdb` 518

12.1.6 Categorizing Process Memory: Virtual Size and Resident Set Size 519

12.1.7 Using pmap to Inspect Process Memory Usage 520
12.1.8 Using DTrace for Memory Analysis 521
12.1.9 Observing MMU Performance Impact with trapstat 523
12.2 Learning About Physical Memory Allocation 523
12.2.1 The Allocation Cycle of Physical Memory 524
12.2.2 Methods for Observing Physical Memory Allocation 527
12.2.3 Total Physical Memory 528
12.2.4 Free Memory 528
12.2.5 Free Memory and the Page Scanner 528
12.2.6 Scan Rate as a Health Indicator 529
12.2.7 Kernel Memory 530
12.2.8 Reference for System Memory Allocation kstats 532
12.3 Learning About Paging 534
12.3.1 File I/O Paging: “Good” Paging 534
12.3.2 Anonymous Memory Paging: “Bad” Paging 535
12.3.2.1 Using prstat To Estimate Memory Slowdowns 536
12.3.2.2 Using DTrace to Estimate Memory Slowdowns 536
12.3.3 Reference for System Paging kstats 538
12.4 Learning About Process Memory Utilization 541
12.4.1 Sharing Executables and Libraries 542
12.4.2 SPARC Address Spaces 543
12.4.3 x86 and x64 Address Space Layout 545
12.4.4 Growing the Heap 546
12.4.5 The Stack 548
12.4.5.1 Memory Mapped Files 548
12.4.6 Using pmap to Look at Mappings 550
12.4.7 Calculating Process Memory Usage with pmap 551
12.4.8 Displaying Page Size Information with pmap 555
12.5 Learning About Swap Space 556
12.5.1 Swap Allocation 556
12.5.2 Swap Statistics 558
12.5.3 Display of Swap Reservations With pmap 562

Chapter 13 Working with Multiple Page Sizes in Solaris 565
13.1 Determining When to Use Large Pages 565
13.2 Measuring Application Performance 566

13.2.1 Determination Allocated Page Sizes 568
13.2.2 Discovery of Supported Page Sizes 571
13.3 Configuring for Multiple Page Sizes 572
13.3.1 Enabling Large Pages 572
13.3.2 Advising Page-size Preferences With `ppgsz`(1M) 573
13.3.3 Interposing Shared Libraries With `libmpss.so` 573
13.3.4 Request Larger Page Sizes with the Compiler 575
13.3.5 Interfaces to Request Larger Page Sizes 576
13.3.6 CPU Specific Large Page Support 579

Chapter 14 Virtual Memory 583
14.1 Design Overview 583
14.2 Tracing the VM System 585
14.3 Virtual Address Spaces 587
14.3.1 Address Space Management 587
14.3.2 Address Space Callbacks 592
14.3.3 Virtual Memory Protection Modes 592
14.3.4 Page Faults in Address Spaces 592
14.4 Segment Drivers 599
14.4.1 The vnode Segment: seg_vn 603
14.4.1.1 Memory Mapped Files 604
14.4.2 Copy-on-Write 606
14.4.3 Page Protection and Advice 607
14.5 Anonymous Memory 607
14.6 The Anonymous Memory Layer 608
14.7 The swapfs Layer 611
14.7.0.1 swapfs Implementation 612
14.8 Virtual Memory Watchpoints 614
14.9 Changes to Support Large Pages 617
14.9.1 System View of a Large Page 617
14.9.2 Free List Organization 618
14.9.3 Page Allocation and Memory Fragmentation 619
14.9.4 Large-Page Faulting 620
14.9.5 Large-Page Freeing 623
14.9.6 Operations That Interfere with Large Pages 624
14.9.7 HAT Support 624

14.9.8 procfs Changes 625

Chapter 15 Physical Memory 627

15.1 Pages—The Basic Unit of Solaris Memory 627
15.1.1 The Page Hash List 628
15.1.2 MMU-Specific Page Structures 629
15.1.3 Free List and Cache List 631
15.1.4 Physical Page "memseg" Lists 631
15.1.5 The Page-Level Interfaces 631
15.1.6 The Page Throttle 634
15.1.7 Page Coloring 634
15.2 The Page Scanner 638
15.2.1 Page Scanner Operation 639
15.2.2 Page-out Algorithm and Parameters 640
15.2.2.1 Scan Rate Parameters 641
15.2.2.2 Not Recently Used Time 642
15.2.3 Shared Library Optimizations 643
15.2.3.1 Page Scanner CPU Utilization Clamp 643
15.2.4 Parameters That Limit Pages Paged Out 643
15.2.4.1 Summary of Page Scanner Parameters 644
15.2.5 Page Scanner Implementation 645
15.2.6 The Memory Scheduler 647
15.2.6.1 Soft Swapping 647
15.2.6.2 Hard Swapping 647

Chapter 16 Kernel Memory 649

16.1 Kernel Virtual Memory Layout 649
16.1.1 Kernel Address Space 650
16.1.2 The Kernel Text and Data Segments 652
16.1.3 Virtual Memory Data Structures 652
16.1.4 The UltraSPARC Kernel Nucleus 652
16.1.5 Loadable Kernel Module Text and Data 653
16.1.6 The Kernel Address Space and Segments 655
16.2 Kernel Memory Allocation 656
16.2.1 The Kernel Heap 657
16.2.2 The Kernel Memory Segment Driver 658

16.2.3 The Kernel Memory Slab Allocator 660
16.2.3.1 Slab Allocator Overview 660
16.2.3.2 Object Caching 663
16.2.3.3 General-Purpose Allocations 665
16.2.3.4 Slab Allocator Implementation 667
16.2.3.5 The CPU Layer 669
16.2.3.6 The Depot Layer 669
16.2.3.7 The Global (Slab) Layer 670
16.2.3.8 Slab Cache Parameters 671
16.2.3.9 Slab Allocator Statistics 673
16.3 The Vmem Allocator 677
16.3.1 Background 677
16.3.2 Vmem Objectives 678
16.3.3 Interface Description 678
16.3.3.1 Creating Arenas 678
16.3.3.2 Allocating and Freeing Resources 681
16.3.3.3 Importing From Another Arena 682
16.3.4 Vmem Implementation 682
16.3.4.1 Keeping Track of Segments 682
16.3.4.2 Allocating and Freeing Segments 683
16.3.4.3 Locking Strategy 685
16.3.4.4 Quantum Caching 685
16.3.4.5 Fragmentation 686
16.3.5 Vmem Performance 687
16.3.5.1 Microbenchmark Performance 687
16.3.5.2 System-Level Performance 688
16.3.6 Summary 688
16.4 Kernel Memory Allocator Tracing 689
16.4.1 Enabling KMA DEBUG Flags 689
16.4.2 Examining Kernel Memory Allocations with MDB 690
16.4.3 Detecting Memory Corruption 692
16.4.4 Freed Buffer Checking: 0xdeadbeef 692
16.4.5 Redzone: 0xfeedface 693
16.4.6 Uninitialized Data: 0xbaddcafe 696
16.4.7 Associating Panic Messages With Failures 697

16.4.8 Memory Allocation Logging 697
16.4.8.1 Buftag Data Integrity 698
16.4.8.2 The bufctl Pointer 699
16.4.9 Advanced Memory Analysis 700
16.4.9.1 Finding Memory Leaks 700
16.4.9.2 Finding References to Data 701
16.4.10 Finding Corrupt Buffers With ::kmem_verify 702
16.4.11 Allocator Logging Facility 703

Chapter 17 Hardware Address Translation 707
17.1 HAT Overview 707
17.2 The x64 HAT 709
17.2.1 MMU Configuration 709
17.2.2 `struct mmu` 712
17.2.3 Virtual Address Space Layout 713
17.2.4 64 bit 715
17.2.5 32 bit 716
17.2.6 HAT IMPLEMENTATION 717
17.2.6.1 `struct hat` 717
17.2.6.2 Page Tables and `struct htable` 718
17.2.6.3 Physical memory and DMA 721
17.3 SPARC HAT 722
17.3.0.4 UltraSPARC-I and II Watchpoint Implementation 726

PART SIX
Files and File Systems **729**

Chapter 18 File System Framework 731
18.1 File System Framework 731
18.2 Process Level File Abstractions 733
18.2.1 File Descriptors 734
18.2.2 The open Code Path 735
18.2.3 Allocating and Deallocating File Descriptors 736
18.2.4 File Descriptor Limits 739
18.2.5 File Structures 740
18.3 Solaris File System Framework 741

18.3.1 Evolution of the File System Framework 743
18.3.2 The Solaris File System Interface 746
18.4 File System Modules 746
18.4.1 Interfaces for Mount Options 747
18.4.2 Module Initialization 748
18.5 The Virtual File System (vfs) Interface 749
18.5.1 vfs Methods 750
18.5.2 vfs Support Functions 754
18.5.3 The mount Method 759
18.5.4 The umount Method 760
18.5.5 Root vnode Identification 761
18.5.6 vfs Information Available with MDB 762
18.6 The Vnode 764
18.6.1 Object Interface 765
18.6.2 vnode Types 767
18.6.3 vnode Method Registration 767
18.6.4 vnode Methods 770
18.6.5 Support Functions for Vnodes 777
18.6.6 The Life Cycle of a Vnode 778
18.6.7 Vnode Creation and Destruction 779
18.6.8 The vnode Reference Count 779
18.6.9 Interfaces for Paging vnode Cache 780
18.6.10 Block I/O on vnode Pages 782
18.6.11 vnode Information Obtainable with MDB 783
18.6.12 DTrace Probes in the vnode Layer 785
18.7 File System I/O 790
18.7.1 Memory Mapped I/O 790
18.7.2 read() and write() System Calls 792
18.7.3 The seg_kpm Driver 793
18.7.4 The `seg_map` Driver 794
18.7.5 Interaction between segmap and segkpm 799
18.8 File systems and Memory Allocation 801
18.8.1 Solaris 8 - "Cyclic Page Cache" 802
18.8.2 The old allocation algorithm 802
18.8.3 The new allocation algorithm 806

18.8.4 Putting it all together: The Allocation Cycle 806
18.9 Path-Name Management 808
18.9.1 The lookuppn() Method 808
18.9.2 The vop_lookup() Method 809
18.9.3 The vop_readdir() Method 809
18.9.4 Path-Name Traversal Functions 811
18.10 The Directory Name Lookup Cache 813
18.10.1 DNLC Operation 813
18.10.2 Primary DNLC Support Functions 815
18.10.3 DNLC Negative Cache 816
18.10.4 DNLC Directory Cache 817
18.10.5 DNLC Housekeeping Thread 821
18.10.6 DNLC Statistics 821
18.11 The File System Flush Daemon 822
18.12 File System Conversion to Solaris 10 822

Chapter 19 The UFS File System 825
19.1 UFS Development History 825
19.2 UFS On-Disk Format 827
19.2.1 On-disk UFS Inodes 828
19.2.2 UFS Directories 832
19.2.3 UFS Hard Links 834
19.2.4 Shadow Inodes 834
19.2.5 The Boot Block 835
19.2.6 The Superblock 836
19.2.7 The Cylinder Group 838
19.2.8 Summary of UFS Architecture 839
19.3 The UFS Inode 842
19.3.1 In-core UFS Inodes 842
19.3.2 Inode Cache 843
19.3.3 Block Allocation 846
19.3.3.1 Layout Policy 846
19.3.3.2 Mapping Files to Disk Blocks 850
19.3.3.3 Reading and Writing UFS Blocks 851
19.3.3.4 Buffering Block Metadata 852
19.3.4 Methods to Read and Write UFS Files 852

19.3.4.1 ufs_read() 852
19.3.4.2 ufs_write() 854
19.4 Access Control in UFS 855
19.5 Locking in UFS 859
19.5.1 UFS Lock Descriptions 860
19.5.2 Inode Lock Ordering 862
19.5.3 UFS Lockfs Protocol 862
19.6 Logging 866
19.6.1 On-disk Log Data Structures 867
19.6.2 In-core Log Data Structures 871
19.6.3 Summary Information 875
19.6.4 Transactions 876
19.6.4.1 Reserving Space in the Log 876
19.6.4.2 Starting Transactions 878
19.6.4.3 Tracking Deltas During Transactions 879
19.6.4.4 Ending the Transaction 879
19.6.5 Rolling the Log 880
19.6.6 Redirecting Reads and Writes to the Log 882
19.6.6.1 lufs_read_strategy() Behavior 882
19.6.6.2 lufs_write_strategy() Behavior 883
19.6.7 Failure Recovery 883
19.6.7.1 Reclaim Thread 884
19.6.7.2 Recovery After a Failure 884

PART SEVEN
Platform Specifics 885

Chapter 20 Support for NUMA and CMT Hardware 887
20.1 Memory Hierarchy Designs 888
20.1.1 What Is NUMA? 888
20.1.1.1 Why NUMA? 888
20.1.1.2 What Is Cache Coherent NUMA? 889
20.1.2 What Is CMT? 889
20.1.2.1 Why CMT? 890
20.1.2.2 CMT and Solaris 890
20.2 Memory Placement Optimization Framework 891

20.2.1 Latency Model 892
20.2.2 More Complex Models 893
20.3 Scheduling 894
20.4 Memory Allocation 895
20.5 Lgroup Implementation 896
20.5.1 Parameters Affecting MPO 898
20.5.2 MPO APIs 900
20.5.2.1 Informational APIs 900
20.5.2.2 MPO Advice APIs 901
20.5.2.3 Explicit Lgroup APIs 902
20.5.3 Verifying the Interface Version 902
20.5.4 Initialization of the Locality Group Interface 903
20.6 Locality Group Hierarchy 903
20.6.1 Locality Group Characteristics 904
20.6.2 Locality Groups and Thread and Memory Placement 905
20.7 MPO Statistics 905

Chapter 21 Locking and Synchronization 907
21.1 Synchronization 907
21.2 Parallel Systems Architectures 908
21.3 Hardware Considerations for Locks and Synchronization 911
21.4 Introduction to Synchronization Objects 917
21.4.1 Synchronization Process 917
21.4.2 Synchronization Object Operations Vector 919
21.5 Mutex Locks 920
21.5.1 Overview 920
21.5.2 Solaris Mutex Lock Implementation 923
21.6 Reader/Writer Locks 928
21.6.1 Solaris Reader/Writer Locks 929
21.7 Turnstiles and Priority Inheritance 933
21.7.1 Turnstiles Implementation 934
21.8 Kernel Semaphores 936
21.9 DTrace Lockstat Provider 939
21.9.1 Overview 939
21.9.2 Adaptive Lock Probes 940
21.9.3 Spin Lock Probes 941

21.9.4 Thread Locks 942
21.9.5 Readers/Writer Lock Probes 942

Chapter 22 SPARC 951
22.1 UltraSPARC Traps 951
22.1.1 UltraSPARC Trap Types 952
22.1.2 UltraSPARC Trap Priority Levels 954
22.1.3 UltraSPARC Trap Levels 954
22.1.4 UltraSPARC Trap Table Layout 955
22.1.5 Software Traps 956
22.1.6 A Utility for Trap Analysis 957
22.2 SPARC System Calls 959
22.2.1 Handling A System Call Trap 959
22.2.2 A Tour Through A System Call 961

PART EIGHT
Networking 969

Chapter 23 The Solaris Network Stack 971
23.1 STREAMS and the Network Stack 971
23.1.1 The STREAMS Model 972
23.1.2 Network Stack as STREAMs Module 975
23.1.2.1 Key Data Structures 977
23.1.2.2 IP as a Multiplexer 978
23.1.3 Issues with STREAMS-based Stacks 978
23.2 Solaris 10 Stack: Design Goals 979
23.3 Solaris 10 Network Stack Framework 980
23.3.1 Vertical Perimeter 981
23.3.2 IP Classifier 985
23.3.3 Synchronization Mechanism 986
23.4 TCP as an Implementation of the New Framework 987
23.4.1 The Interface Between TCP and IP 989
23.4.2 TCP Loopback 991
23.5 UDP 992
23.5.1 UDP Packet Drop Within the Stack 993
23.5.2 UDP Module 993

23.5.3 UDP and Socket Interaction 995
23.6 Synchronous STREAMS 995
23.6.1 TCP Synchronous STREAMS 995
23.6.2 STREAMS Fallback 996
23.7 IP 997
23.7.1 Plumbing NICs 998
23.7.2 IP Network Multipathing 998
23.7.3 Multicast 999
23.8 Solaris Device Driver Framework 999
23.8.1 GLDv2 and DLPI Drivers (Solaris 9 and prior) 999
23.8.2 A New Architecture: GLDv3 1000
23.8.2.1 GLDv3 Drivers 1001
23.8.2.2 MAC Services Module 1005
23.8.2.3 Data-Link Services Module 1006
23.8.2.4 Data-Link Driver 1006
23.8.3 GLDv3 Link Aggregation Architecture 1007
23.8.4 Checksum Offload 1009
23.9 Interrupt Model and NIC Speeds 1010
23.9.1 Solaris 9 and Earlier Releases 1010
23.9.2 Dynamic Switch Between Interrupt vs. Polling Mode 1011
23.9.3 Interrupt Load Spreading 1013
23.10 Summary 1014

PART NINE
Security **1015**

Chapter 24 Process Rights Management 1017
24.1 Then and Now 1017
24.2 Least Privilege in Solaris 1018
24.3 Process Privileges Models 1019
24.3.1 The Traditional Solaris Superuser Model 1020
24.3.2 Extending Solaris With Process Privileges 1021
24.3.3 How the Solaris 10 Least Privilege Model Was Chosen 1022
24.3.3.1 Trusted Solaris 1023
24.3.3.2 Root Set 1023

24.3.3.3 As Many Sets As It Takes 1023
24.3.3.4 Privilege Awareness 1024
24.3.4 Other Unix Implementations 1025
24.4 Privilege Awareness: The Details 1028
24.4.1 Per-Process State 1028
24.4.2 Privilege Awareness State Transitions 1029
24.4.2.1 Becoming Privilege Aware 1029
24.4.2.2 Losing Privilege Awareness 1029
24.4.3 Privilege State Manipulation 1030
24.4.3.1 What Happens When User IDs Change? 1030
24.4.3.2 What Happens When a Process Is Created? 1030
24.4.3.3 What Happens When a Program Is Exec'd? 1030
24.4.3.4 Manipulating Privileges Directly 1031
24.4.3.5 Manipulating pas Directly 1033
24.4.3.6 Manipulating Privileges Through proc(4) 1033
24.4.4 Privilege Escalation Prevention 1034
24.4.5 The Trouble With uid 0 1035
24.4.6 Basic Privileges 1036
24.4.7 Privileges and the Runtime Environment 1037
24.4.8 Privileges and NFS 1038
24.4.9 Privileges and Third-Party File Systems 1039
24.5 Least Privilege Interfaces 1039
24.5.1 The Conspiracy of Bit Sets and Constants 1039
24.5.2 Privilege Names and Constants 1040
24.5.3 Kernel Data Structures 1041
24.5.4 Kernel Interfaces 1044
24.5.5 System Call Interfaces 1046
24.5.6 Library Interfaces 1049
24.5.6.1 Privilege Specific Library Interfaces 1049
24.5.6.2 User Credential Library Interfaces 1051
24.5.6.3 Private Daemon and Set-uid Interfaces 1052
24.5.7 Enhancements to Role-Based Access Control 1053
24.5.8 Enhancements to proc(4) and Core Dumps 1055
24.5.9 Privilege Debugging 1056
24.5.10 Privilege Auditing 1057

24.5.11 Device Protection 1058

PART TEN
Misc **1061**

Chapter 25 Timers and Clocks 1063
25.1 The Cyclic Subsystem 1063
25.1.1 Cyclic Subsystem Interface Overview 1064
25.1.2 Cyclic Subsystem Implementation Overview 1065
25.1.2.1 Heap Management 1066
25.1.2.2 Expiry Processing 1069
25.1.2.3 The Producer/Consumer Buffer 1069
25.1.2.4 Locking 1071
25.1.2.5 Resizing 1072
25.1.2.6 Removals 1073
25.1.2.7 Juggling 1074
25.1.3 Clients of the Cyclic Subsystem 1074
25.1.4 Cyclic Kernel At-Large Interfaces 1075
25.1.5 Cyclic Kernel Inter-Subsystem Interfaces 1083
25.1.6 Cyclic Backend Interfaces 1085
25.1.7 Cyclic Subsystem Backend-Supplied Interfaces 1085
25.2 The System Clock Thread 1090
25.2.1 Thread Tick Processing 1092
25.2.2 DTrace Providers for Tick Processing 1093
25.3 Callouts and Callout Tables 1093
25.4 System Time Facilities 1099
25.4.1 High-Resolution Timer 1099
25.4.2 Time-of-Day Clock 1100

Chapter 26 Task Queues 1103
26.1 Overview of Task Queues 1103
26.2 Dynamic Task Queues 1104
26.2.1 Why a *Dynamic* Task Queue? 1104
26.2.2 Problems Address by Dynamic Task Queues 1105
26.2.3 Task Pool Model 1106
26.2.4 Interface Changes to support Dynamic Task Queues 1107

26.3 Task Queues Kernel APIs 1109
26.4 DDI interface for task queues 1111
26.5 Task Queue Observability 1112
26.5.1 Kstat Counters 1112
26.5.2 DTrace SDT Probes 1113
26.6 Task Queue Implementation Notes 1114
26.6.1 Use of kmem caches. 1114
26.6.2 Use of vmem arenas. 1115
26.6.3 Hashed vmem arenas. 1116
26.6.4 Cached list of entries 1116
26.6.5 Problems with task pool implementation. 1117
26.6.6 Use of dynamic task pools in STREAMS. 1118

Chapter 27 Interrupts 1119
27.1 Access to Kernel Services 1119
27.2 Entering Kernel Mode 1120
27.2.1 Context 1121
27.2.1.1 Execution Context 1121
27.2.1.2 Virtual Memory Context 1122
27.2.2 Threads in Kernel and Interrupt Context 1122
27.3 Interrupts 1123
27.3.1 Interrupt Priorities 1124
27.3.1.1 Interrupts as Threads 1124
27.3.1.2 Interrupt Thread Priorities 1126
27.3.1.3 High-Priority Interrupts 1127
27.3.2 Interrupt Monitoring 1127
27.3.3 Interprocessor Interrupts and Cross-Calls 1128

1131

Appendix A Kernel Virtual Address Maps 1131

1139

Appendix B Adding A System Call to Solaris 1139
29.3.4 Picking a syscall number 1139
29.3.5 Writing the syscall handler 1140

29.3.6 Adding an entry to the syscall table 1140
29.3.7 Updating `/etc/name_to_sysnum` 1141
29.3.8 Updating truss(1) 1141
29.3.9 Updating `proc_names.c` 1142
29.3.10 Putting it all together 1142

Bibliography 1145