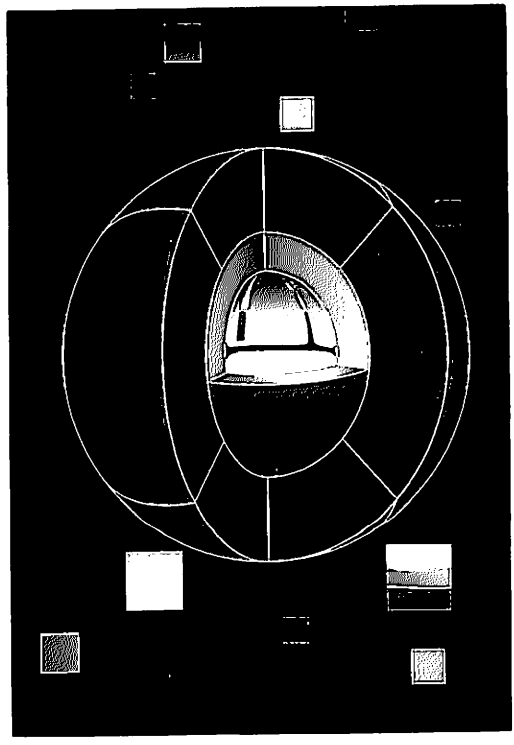
NCR System 3000
NCR UNIX SVR4 MP–RAS



NCR
An AT&T Company

Administrator Guide:
Command Line Interface
System Configuration (Volume 3)
Release 2.02

September 1993
D1–2266–D

The program product(s) described in this book is a licensed product of NCR Corporation.

UNIX is a registered trademark of UNIX Systems Laboratories.

It is the policy of NCR Corporation to improve products as new technology, components, software, and firmware become available. NCR Corporation, therefore, reserves the right to change specifications without prior notice.

All features, functions, and operations described herein may not be marketed by NCR in all parts of the world. In some instances, photographs are of equipment prototypes. Therefore, before using this document, consult your NCR representative or NCR office for information that is applicable and current.

To maintain the quality of our publications, we need your comments on the accuracy, clarity, organization, and value of this book. Address correspondence to:

Information Products
NCR Corporation
3325 Platt Springs Road
West Columbia, SC 29170
U.S.A.

Preface

Who Should Read This Book

This book is for the knowledgeable system administrator, that is, the person who administers the system using the operating system commands at the shell prompt (# for the **root** login).

Releases Covered In This Book

This book applies to Release 2.02 of the operating system for your NCR System 3000 computer.

What You Should Know

You should be familiar with the information in the following books:

- *User Guide: Introduction to NCR UNIX SVR4*
- *NCR UNIX SVR4 MP-RAS Administrator Guide: OA&M Menu Interface*

How To Use This Book

This book assumes you are familiar with the *NCR UNIX SVR4 MP–RAS Administrator Guide: OA&M Menu Interface* and takes you one step beyond that book to include:

- Command line procedures that perform some of the same tasks that the menu procedures perform in the *NCR UNIX SVR4 MP–RAS Administrator Guide: OA&M Menu Interface*

- Command line procedures for many more advanced tasks
- Diagnostics and trouble–shooting procedures

Rather than lead you through chapter by chapter, this book provides a reference in which you can look up a particular topic or task. Use the *NCR UNIX SVR4 MP–RAS Reference Manual* as needed to obtain additional information about a particular command.

This book consists of the following volumes:

Volume 1 — General Administration
This volume describes the following:

- System administration logins, duties, and schedules
- Changing run levels
- Using the Reference, Diagnostics, and boot flex diskettes
- Recovery procedures using the Maintenance (*/install*) file system
- Administering, checking, and repairing the file system
- Backing up and restoring information

Volume 2 — Devices and Networks
This volume describes the following:

- Installing additional disks, adapters, and terminals
- Setting up and managing the LP Print Service
- Setting up and administering the Basic Networking Utilities (BNU)

Volume 3 — System Configuration

This volume describes the following:

- The files and directories used in configuration
- The process of reconfiguring the system and building new kernels
- The tunable configuration parameters

Appendices and Glossary

The appendices describe the file system, device names and numbers, customizing the sysadm interface, using national characters on the console, and using sendmail. The glossary defines terms you encounter in this book.

Conventions Used In This Book

- The following type identifies text that you must enter exactly as shown:

```
login su
```

- The following type identifies output from a screen or command:

```
Press the Return key to continue.
```

- Path names and file names appear in italics. For example:

  The */etc/profile* file defines the standard environment for all users.

- Configuration parameters and shell variables appear in capital letters. Capital letters are used for emphasis also. For example:

  Press RETURN after specifying a value for the TERM variable. Do NOT leave this value blank.

- Utilities, commands, user names, and terminal keys appear in boldface type. For example:

  The **cpio**(1) command backs up files.

- Parameters for command line options appear in italics within the text. For example:

  Specify the class (**–c***class*) for the printer.

- The numbers and letters in parentheses after a command name indicate the type of command.

  (1)
  Standard command available to users at the $ prompt.

  (1M)
  Command available only to someone logged in as **root**

  (2)
  System call

  (3)
  Library function

Related Publications

For supplementary information, refer to the following books:

- *User Guide: Introduction to NCR UNIX SVR4*
- *NCR UNIX SVR4 MP–RAS Administrator Guide: OA&M Menu Interface*
- *NCR UNIX SVR4 Message Manual*
- *NCR 345x/35xx Unit Installation, Care, and Cleaning*
- *NCR UNIX SVR4 MP–RAS Reference Manual*

Contents

Volume 1 — General Administration

Chapter 1 **Common System Administration Tasks**

Accessing the System 1–2

Changing Run Levels 1–5

Starting Up and Shutting Down the System 1–7

Specifying Run Levels During System Startup 1–9

Setting Up Your System for Automatic or Manual File System Checks 1–12

Setting up an Alternate Boot Disk 1–17

Maintaining the Password and Shadow Files 1–28

Managing Software Packages 1–30

Managing Multiple Processors 1–32

Setting the Power Fail Strategy 1–37

Customizing Memory Dumps 1–44

Installing from a Remote Console (NCR 3450/3550 Only) 1–47

Chapter 2 **Using the Maintenance and Reference Diskettes**

What Are Maintenance and Reference Diskettes? 2–2
Booting from a Flex Diskette 2–4
Copying Maintenance and Reference Diskettes 2–6
Accessing the Maintenance File System 2–11

Chapter 3 **Recovery Techniques**

Recovering Files in the /stand File System 3–2
Recovering unix 3–6
Recovering the root Password 3–10
Recovering the /etc/passwd File 3–13
Recovering the /etc/shadow File 3–17
Recovering Files from the Installation Tape 3–21
Recovering Sector 0 3–25
Recovering the Boot Loader Program 3–29
Saving Memory to Dump Area 3–35
Saving Dump Area to Media 3–37
Examine the Contents of the Dump 3–39
Checking a File System from the Maintenance File System 3–40

Chapter 4 **Administering the File System**

General Administration of File Systems 4–2
Making (Creating) File Systems 4–7
Mounting File Systems 4–11
Unmounting File Systems 4–13

Displaying File System Information 4–15
Compacting Files in /stand 4–18
Using Quotas 4–20

Chapter 5 **Checking and Repairing the File System**

Minimizing File System Corruption 5–2
Using fsck to Check and Repair the File System 5–4
Checking and Repairing s5 File Systems 5–8
s5 File System Components Checked by fsck 5–11
Phases of fsck on s5 File Systems 5–19
Checking and Repairing ufs File Systems 5–38
ufs File System Components Checked by fsck 5–40
Phases of fsck on ufs File Systems 5–47
Checking bfs File Systems 5–78

Chapter 6 **Backing Up Information**

What You Should Know Before You Perform a Backup 6–2
Backing Up File Systems 6–4
Backing Up Individual Files and Directories 6–10
Backing Up Raw Devices 6–14
Backing Up File System and Disk Information 6–16
Backing Up the Entire System 6–19
Backing Up an Entire NCR 3600 6–20
Copying Files to NCR TOWERs 6–25
Retensioning a Cartridge Tape 6–26

Chapter 7

Restoring Information

Restoring File Systems 7–2

Restoring Individual Files and Directories 7–5

Restoring Raw Devices 7–8

Restoring an Entire Disk 7–10

Restoring the Entire System 7–20

Restoring a Corrupted Root File System 7–21

Restoring an Entire NCR 3600 7–27

Chapter 8

Administering the UNIX SVR4 Mail Subsystem

Understanding SVR4 Mail 8–2

Establishing a Smarter Host 8–5

Establishing Domain Addresses 8–6

Establishing a Mail Cluster or Gateway 8–7

Establishing Mail Service on a Networked File System (RFS or NFS) 8–8

Administering Alias Lists 8–11

Other Surrogate File Routines 8–12

Enabling SMTP 8–14

Stopping and Starting the SMTP Daemon 8–17

Disabling SVR4 Mail 8–18

Setting Up SMTP to Listen Over Multiple Networks . . 8–20

Reading the SMTP Log File 8–21

Appendix A **The File System**

The File System Hierarchy A–2

Symbolic Links A–5

File System Organization A–6

File System Types A–8

The s5 File System Type A–10

The ufs File System Type A–17

The bfs File System Type A–23

The cdfs File System Type A–27

Appendix B **Device Names and Numbers**

Major and Minor Device Numbers B–2

Device Names and Minor Numbers B–5

Appendix C **Customizing the sysadm Interface**

Interface Structure: a Hierarchy of Menus C–2

Writing Your Help Messages C–8

Creating or Changing a Menu Entry C–18

Creating or Changing a Task Entry C–25

Deleting a Menu or Task Entry C–31

Appendix D — **Using National Characters on the Console**

Changing Line Characteristics to 8 Bit D–2
Changing Keyboard Mapping D–4
Changing Screen Mapping D–6

Appendix E — **Sendmail – An Internetwork Mail Router**

Abstract E–2
Design Goals E–5
Overview E–8
Usage and Implementation E–13
Comparison with other Mail Programs E–19
Evaluations and Future Plans E–22
References E–25

Appendix F — **Sendmail Installation and Operation**

Introduction F–2
Basic Installation F–4
Normal Operations F–10
Arguments F–19
Tuning F–21
The Configuration File F–27
Command Line Flags F–48
Configuration Options F–50
Mailer Flags F–55
Other Configuration F–58

Summary of Support Files F–66
Error/Status Messages F–68

Volume 2 — Devices and Networks

Chapter 1

Configuring Disks

Using Disk Utilities 1–2
Adding Disks 1–3
Formatting Disks 1–6
Partitioning Disks 1–8
Writing Boot Code and Initializing Disks 1–13
Slicing Disks 1–14
Marking Bad Blocks 1–19
Displaying Slicing Information 1–23
Changing the Active Partition 1–24
Configuring Swap Space 1–25
Configuring the Dump Area 1–31
Replacing a Failed Disk 1–35

Chapter 2

Configuring Adapters

Configuring Adapters at System Startup 2–2
Configuring Adapters During System Operation 2–11
Installing a Serial Controller 2–18
Installing an Ethernet LAN Adapter 2–23
Adding a SCSI Controller to a Uniprocessor System ... 2–27
Adding a SCSI Controller to a Multiprocessor System . 2–34

Chapter 3

Configuring Serial Devices

The Service Access Facility 3–2

The Service Access Controller 3–3

The Port Monitor ttymon 3–8

Configuring a Serial Port for a Terminal 3–13

Configuring a Serial Port for a Modem 3–15

Configuring the EIA Mode for a Serial Port 3–18

Configuring a Second Serial Port 3–20

Configuring Serial Controller Ports for Terminals 3–22

Configuring a Serial Controller for Modems 3–25

Setting Modem Options 3–29

Enabling Dial-up Password Protection 3–32

Configuring a Serial Port for a Three–Wire Cable 3–35

Chapter 4

Using The LP Print Service

What is the LP Print Service? 4–2

Getting Started 4–10

Installing the LP Print Service 4–11

Adding Printers 4–12

Enabling Remote Printing 4–17

Managing Printer Classes 4–25

Setting Other Printing Options 4–27

Displaying the Status of Printers 4–37

Starting and Stopping the LP Print Service 4–39

Managing the Printing Load 4–41
Troubleshooting 4–49

Chapter 5 **Managing the LP Print Service**

Managing Queue Priorities 5–2
Managing Printer Faults 5–7
Providing Forms 5–12
Providing Character Sets or Print Wheels 5–24
Providing Filters 5–32
Cleaning Out the Request Log 5–46

Chapter 6 **Adjusting the LP Print Service**

PostScript Printers 6–2
Customizing the Print Service 6–12

Chapter 7 **Network Selection**

Network Selection 7–2
Name–to–Address Mapping 7–10

Chapter 8 **Basic Networking Utilities**

General Description of the Basic Networking Utilities . 8–2
Commands 8–5
Networking Daemons 8–8
Supporting Data Base 8–10
Administrative Support Files 8–12
Logs 8–16

Chapter 9

Managing BNU

Basic Procedures 9–2

Procedure for Setting Up a Link 9–5

Establishing the Physical Connection 9–7

Setting the Node Name of the Running Kernel 9–9

Identifying Systems for Communication 9–11

Adding uucp logins 9–20

Identifying Devices Used for Communication 9–21

Writing Dialing Information 9–30

Creating Access and Security Mechanisms 9–35

Editing Additional Files 9–48

Specifying File Transfer Protocols 9–60

Verifying the Link To Be Sure It Works 9–63

Maintaining BNU 9–64

Common Problems 9–71

Troubleshooting with cu 9–73

Troubleshooting with Uutry 9–74

Using Other Troubleshooting Tools 9–77

Basic Networking Utilities Error Messages 9–78

Volume 3 — System Configuration

Chapter 1 — **Monitoring System Activity**

Using Monitoring Utilities 1–2
Collecting Activity Data with sadc 1–5
Analyzing Activity Data with sar 1–7
Analyzing Activity with Process Accounting 1–35
Analyzing Specific Processes with time 1–45
Analyzing Specific Processes with timex 1–47
System Profiling 1–49

Chapter 2 — **Process Scheduling**

Using the Process Scheduler 2–2
Configuring the Scheduler 2–8
Changing Scheduler Parameters 2–21

Chapter 3 — **Configuring Kernels**

Configuration Files and Directories 3–2
Creating New Kernels 3–11
Loading/Booting the Kernel 3–15
Modules 3–17
Configuration Parameters 3–25

Chapter 4 — **Configuration Parameters**

Chapter 5

Tuning Basics

Reasons for Tuning 5–2
Basic Steps 5–3
Knowing When It's Time To Tune 5–6
System Resources 5–7
Potential Bottlenecks 5–10
Basic Tuning Procedure 5–13

Chapter 6

Memory Utilization

Memory Management 6–2
Demand Paging 6–3
Swapping 6–7
Flushing 6–10

Chapter 7

Disk Utilization

Factors Affecting Disk Performance 7–2
I/O Balancing 7–3
Buffer Utilization 7–4
Monitoring and Controlling Disk Usage 7–7
Directory Reorganization 7–14
Eliminating File System Fragmentation 7–16

Chapter 8

CPU Utilization

Workload and CPU Time 8–2

Efficient PATH Variables 8–3

Excessive Pathname Searches 8–5

Figures

Volume 1 — General Administration

Figure 1–1:	Run Levels (1 of 2)	1–5
Figure 1–2:	Run Levels (2 of 2)	1–6
Figure 1–3:	Relationships Between /etc/passwd and /etc/shadow	1–29
Figure 1–4:	Power Backup System Support Hardware	1–39
Figure 4–1:	Possible File System Block Sizes	4–8
Figure 5–1:	ufs File System Specific Options	5–39
Figure 8–1:	Status Values	8–24
Figure 8–2:	Error Codes	8–27
Figure A–1:	The Root File System	A–3
Figure A–2:	Root File System Contents	A–4
Figure A–3:	A UNIX File System	A–6
Figure A–4:	Adding the /usr File System	A–7

Figure A–5:	The UNIX View of an s5 File System .	A–11
Figure A–6:	The File System Address Chain for s5	A–15
Figure A–7:	The UNIX View of a ufs File System .	A–18
Figure A–8:	The File System Address Chain in a ufs File System	A–21
Figure A–9:	The UNIX View of a bfs File System .	A–23
Figure B–1:	Major and Minor Numbers in BASE O.S. (1 of 3)	B–2
Figure B–2:	Major and Minor Numbers in BASE O.S. (2 of 3)	B–3
Figure B–3:	Major and Minor Numbers in BASE O.S. (3 of 3)	B–4
Figure B–4:	Minor Number Bit Definitions	B–8
Figure B–5:	SCSI Device Name Components	B–9
Figure B–6:	Example 1 of SCSI Device Name and Number	B–10
Figure B–7:	Example 2 of SCSI Device Name and Number	B–10
Figure C–1:	Sysadm Main Menu	C–2
Figure C–2:	Sysadm Machine Menu	C–3
Figure C–3:	Item Help File for One Form	C–15
Figure C–4:	FACE Form Associated with Item Help File in Figure C–3	C–15

Figure C–5: Item Help File for Multiple Forms C–16

Figure C–6: Item Help File for Multiple Forms C–17

Figure C–7: Sample Menu Definition Form C–24

Figure C–8: Sample Task Definition Form C–30

Figure E–1: Sendmail System Structure E–6

Figure F–1: Rewriting Set Semantics F–37

Volume 2 — Devices and Networks

Figure 1–1: A Disk Containing a 100% NCR UNIX Partition 1–8

Figure 1–2: A Disk Containing Multiple Partitions . 1–9

Figure 1–3: The fdisk Utility Screen 1–11

Figure 1–4: Appropriate Slice Assignments 1–15

Figure 1–5: TAG Names and Descriptions 1–17

Figure 1–6: Initially Installed Values for Swap Space 1–25

Figure 1–7: Initial and Required Values for Dump Space 1–32

Figure 2–1: Board Base Addresses and Interrupt Levels 2–21

Figure 3–1: Example of sac, ttymon, login, and shell Relationships 3–10

Figure 4–1:	User Commands for the LP Print Service	4–8
Figure 4–2:	Administrator Commands for the LP Print Service	4–9
Figure 4–3:	Accepted content types	4–29
Figure 4–4:	Default Printer Port Characteristics	4–31
Figure 5–1:	Filter Keywords	5–39
Figure 5–2:	Information in Request Log (1 of 3)	5–47
Figure 5–3:	Information in Request Log (2 of 3)	5–48
Figure 5–4:	Information in Request Log (3 of 3)	5–49
Figure 6–1:	Non-PostScript Print Requests	6–4
Figure 6–2:	PostScript Translation Filters	6–7
Figure 6–3:	Special PostScript Filters	6–7
Figure 6–4:	Printer Port Characteristics	6–13
Figure 6–5:	Printer Definitions in terminfo (1 of 3)	6–15
Figure 6–6:	Printer Definitions in terminfo (2 of 3)	6–16
Figure 6–7:	Printer Definitions in terminfo (3 of 3)	6–17
Figure 6–8:	Exit Codes for Interface Programs	6–24
Figure 7–1:	Fields in netconfig Entries	7–3
Figure 7–2:	Sample netconfig File	7–8

Figure 8–1: Format of the Command Log 8–16
Figure 8–2: Format of System History Log 8–17
Figure 8–3: Format of Error Log 8–18
Figure 8–4: Format of Transfer Log 8–19
Figure 8–5: Output for File Transfer Requests 8–20
Figure 8–6: Format of Accounting Log 8–21
Figure 8–7: Format of xfer Entry in Security Log .. 8–22
Figure 8–8: Format of rexe Entry in Security Log .. 8–23
Figure 8–9: Performance Log for conn Type Record 8–24
Figure 8–10: Performance Log for xfer Type Record 8–25
Figure 8–11: Format of Foreign Log 8–26
Figure 9–1: Modem Settings for BNU Features 9–7
Figure 9–2: BNU Escape Characters 9–17
Figure 9–3: Escape Characters in the Dialers File .. 9–32

Volume 3 — System Configuration

Figure 1–1: Cron Entries for sadc 1–6
Figure 1–2: sar Display Options (1 of 2) 1–9
Figure 1–3: sar Display Options (2 of 2) 1–10

Figure 1–4:	Example crontab Entries	1–37
Figure 1–5:	Example crontrab Entry for Monacct	1–38
Figure 2–2:	How the Process Scheduler Works	2–4
Figure 2–3:	Scheduling Order and Global Priorities	2–9
Figure 3–1:	Configuration Files And Directories	3–3
Figure 3–2:	Options and Actions of the idtune Command	3–8
Figure 3–3:	Files in the Modules Directories	3–17
Figure 3–4:	Modules Available for the Kernel (1 of 7)	3–18
Figure 3–5:	Modules Available for the Kernel (2 of 7)	3–19
Figure 3–6:	Modules Available for the Kernel (3 of 7)	3–20
Figure 3–7:	Modules Available for the Kernel (4 of 7)	3–21
Figure 3–8:	Modules Available for the Kernel (5 of 7)	3–22
Figure 3–9:	Modules Available for the Kernel (6 of 7)	3–23
Figure 3–10:	Modules Available for the Kernel (7 of 7)	3–24
Figure 3–11:	General Kernel Parameters (1 of 2)	3–29

Figure 3–12: General Kernel Parameters (2 of 2) ... 3–30
Figure 3–13: File System Parameters 3–31
Figure 3–14: Paging Parameters 3–32
Figure 3–15: pageout Daemon Parameters 3–32
Figure 3–16: STREAMS Parameters 3–33
Figure 3–17: log (strlog) Parameters 3–34
Figure 3–18: IPC Message Parameters 3–35
Figure 3–19: IPC Semaphore Parameters 3–35
Figure 3–20: IPC Shared Memory Parameters 3–36
Figure 3–21: RFS Parameters 3–37
Figure 3–22: XENIX Parameters 3–38
Figure 3–23: Miscellaneous Parameters 3–38
Figure 3–24: Driver Parameters 3–39
Figure 3–25: ASYNCIO Parameters 3–40
Figure 3–26: EVENTS Parameters 3–41
Figure 3–27: Timer and Scheduler Parameters 3–42
Figure 3–28: Affinity Scheduling Parameters 3–42
Figure 3–29: Resource Limit Parameters 3–43
Figure 3–30: Overflow Buffers Parameters 3–44

Figure 3–31: Shared Memory Nailing GIDs 3–44
Figure 3–32: bdevcnt, cdevcnt Parameters 3–44
Figure 3–33: Security Audit Trail Parameter 3–45
Figure 3–34: Integrated LAN Driver Parameter 3–45
Figure 3–35: Level 5 Tunable Parameters 3–45
Figure 3–36: System Parameters 3–46
Figure 3–37: Hardware Parameters 3–47
Figure 7–1: Files and Directories That Grow 7–9

Chapter 1 Monitoring System Activity

To detect performance bottlenecks and their causes, you should monitor the system activity at periodic intervals. If you make any changes to try and improve performance, monitor the system activity both before and after the change to determine the effect of the change. You also may want to monitor the system activity during the execution of a process to determine its use of system resources and its effect on system performance.

This chapter contains the following information:

Using Monitoring Utilities 1–2

Collecting Activity Data with sadc 1–5

Analyzing Activity Data with sar 1–7

Analyzing Activity with Process Accounting 1–35

Analyzing Specific Processes with time 1–45

Analyzing Specific Processes with timex 1–47

System Profiling 1–49

Using Monitoring Utilities

Monitoring system activity requires two steps:

Step 1: Collect the system activity data.

Step 2: Analyze the system activity data.

Some reasons to monitor system activity include:

- Periodic system activity data collection provides information that you can analyze to determine if any bottlenecks exist on your system. If you collect data daily, you can analyze this data when you detect a performance problem.
- Data collection before and after a configuration change can help you determine the system resources used by that process and its effect on system performance.

sadc

The **sadc**(1M) command collects the system activity data in a file for later analysis. The data may be collected at periodic intervals, under user control (normally performed from cron entries), or data may be collected for the current system activity.

sar

The **sar**(1) command analyzes system activity data and can be used to analyze either a file created by **sadc** or the current system activity. The various options of **sar** may be used to analyze specific system activity. For example, the **–u** option displays CPU utilization.

Process Accounting

Another way to analyze system activity is the Process Accounting utilities which report on the activity of users and processes. For example, the reports may include the number of times each command was called daily, the system utilization of each user, and so on. This information may be used to detect system misuse or determine the usage of commands.

There are two ways to analyze specific processes:

time(1):
Displays the elapsed time (response time) and execution time for a process. No system activity data is collected.

timex(1):
Collects and displays a report of the system activity during the execution of a specific process. Also, **timex** displays the elapsed time and execution time for the process.

Analyzing Daily Activity

Normally, you collect daily system activity data for analysis later (on a regular basis or when a problem occurs).

Step 1: Collect the daily system activity data using **sadc** (invoke using **cron**).

Step 2: Analyze the data file using **sar.**

The **sar** output includes the activity for the time intervals when data was collected and an average data for each data category.

Later sections in this chapter explain how to use these commands.

Analyzing Current System Activity

The current system activity is normally analyzed to provide a baseline measurement for comparison. You can collect activity data for a specific combination of processes both before and after a configuration change.

Step 1: Use **sar** to display a system activity data report for a specific time interval, for example, to display an activity report once a minute for 10 minutes.

Step 2: Repeat the command several times for a more accurate picture of the system activity.

Step 3: Compile an average for each data category for comparison.

The section "Analyzing Activity Data with sar" in this chapter explains how to use these commands.

Analyzing Specific Process Activity

The activity generated by a specific process may be collected and analyzed for many reasons. For example, you may want to know what system resources the process uses because a large number of users on the system execute the process.

Step 1: Collect the system activity data during the execution of the process using the **timex** command.

Step 2: Note the elapsed time (response time) and execution time for later comparison.

Step 3: Analyze the output file using **sar.**

The section "Anlyzing Specific Processes with timex" in this chapter explains how to use these commands.

Collecting Activity Data with sadc

The **sadc** command collects activity data for a specified time interval and places the data in a file (or standard output). The data in the file may be analyzed later using the **sar** command.

sadc is normally executed using the **cron** facility; this makes it easy to execute **sadc** at specific time intervals. For example, you can add an entry in the *crontab* file for the **root** user so that **sadc** collects activity data once per hour during normal working hours.

sadc is also normally executed once at start–of–day to record the time that all counters were reset to zero. (A script that invokes **sadc** must be placed in *rc1.d*).

The data from **sadc** may be placed in any file (the default is standard output); the normal file is */var/adm/sa/sa***dd**, where **dd** is the day of the month.

Executing sadc at Start–of–Day

At start–of–day, all of the system activity counters are reset to zero. Therefore, for accurate analysis of system activity, the time of the system restart should be recorded in the data file.

If **sadc** is executed with only the data file name specified, an entry is made in the file to record the system restart. The following entry is in the script */etc/rc1.d/S21perf* to record the system restart:

```
su sys -c "/usr/lib/sa/sadc /var/adm/sa/sa`date +%d`&"
```

This command executes **sadc** as the user **sys**, placing the system restart record in the file */usr/adm/sa/sa***dd** (**dd** is the day of the month).

Executing sadc at Periodic Intervals

The shell script */usr/lib/sa/sa1* executes **sadc** and automatically places the output in the file */usr/adm/sa/sa**dd**. You can have **cron** invoke this shell script.

The first argument for the *sa1* shell script is the number of seconds for each sample and the second argument is the number of samples to take.

The entries in Figure 1–1 may be added to the root user's crontab entries to record the system activity data (Monday–Friday) every 20 minutes from 8 a.m. to 5 p.m., and once per hour otherwise.

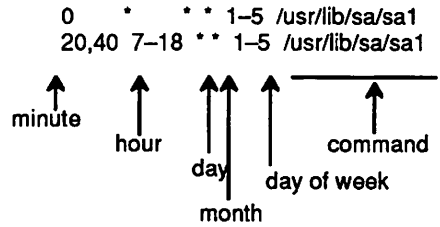


Figure 1–1: Cron Entries for sadc

Analyzing Activity Data with sar

The **sar** command displays system activity data in a report that is organized by system resource. This report may be used to analyze the system activity and determine the performance bottlenecks.

sar has two formats:

- The first format displays a report from the data in a previously created data file (normally */var/adm/sa/sa**dd***). If a configuration change occurs, **sadc** copies the old *sadd* file to *sadd.**hh:mm*** and creates a new *sadd*, which starts at ***hh:mm***.
- The second format displays a report on the current system activity for a specific time interval.

Format for sar Report from Previously Created File

```
# sar [options] [-s time] [-e time] [-i sec]
[-f file]
```

options
The display options for **sar**; described later in this section.

–s *time*
Starting time for the report: *hh*[:*mm*[:*ss*]]. If omitted, the report starts with the first record in the file.

–e *time*
Ending time for the report: *hh*[:*mm*[:*ss*]]. If omitted, the report ends with the last record in the file.

–i *sec*
Selects records at *sec* second intervals. If omitted, all records in the file are displayed.

–f *file*
The system activity data file. If omitted, */var/adm/sa/sa***dd** is used (**dd** is the day of the month).

The following command displays a full report on all of the data in the default file for today's date (*/var/adm/sa/sa***dd**):

```
# sar -A
```

The following command displays a full report on the data in the default file for today's date for the time between 8 a.m. and 5 p.m.:

```
# sar -A -s08 -e17
```

If a configuration change occurred at 11 a.m. and you run the previous command, only information from 11 a.m. to 5 p.m. is reported.

Format for sar Report on Current System Activity

```
# sar [options] [-P processor-id] [-o file] t [n]
```

–P *processor–id*
Reports activity that applies only to the *processor–id* specified; command line options that request information not specific to *processor–id* are silently ignored. Options that are effective with **–P** are **abcgmuwyD**. If the **–o** option is specified with the **–P** option, **sar** saves the processor–specific samples only in *file* in binary format.

options
The display options for **sar**; described later in this section.

–o *file*
Saves the data in *file* in binary format; this file may be analyzed later by **sar** (with the other format). If omitted, the report is displayed and the data is not saved.

t
The length of each sample interval in seconds.

n
Number of samples to take. If omitted, one sample is taken.

The following command takes 10 samples of 60 seconds each and displays a full report:

```
# sar -A 60 10
```

Display Options for sar

The **sar** display options in Figure 1–2 and Figure 1–3 are separated by category and presented with the most commonly used options first. Each option is described in detail on the pages that follow.

CPU / Memory Options	
–u	Reports CPU utilization.
–q	Reports the average queue length and the percentage of time occupied for the Run Queue.
–w	Reports system swapping and switching activity.
–p	Reports paging and fault activity.
–r	Reports unused memory and swap space.
–c	Reports system call activity.
–g	Reports page–out and memory freeing.
–k	Reports kernel memory allocation.

Figure 1–2: sar Display Options (1 of 2)

	Disk Options
–b	Reports buffer activity.
–d	Reports block device (disk) activity.
–a	Reports file access system routine activity.
	TTY Option
–y	Reports TTY device activity.
	Miscellaneous Options
–v	Reports status of Process, Active I–node, and Open File Tables
–m	Reports message queue and semaphore activity.
–A	Equivalent to specifying all display options.

Figure 1–3: sar Display Options (2 of 2)

sar –u

The **sar –u** command measures CPU utilization. At any given moment, the CPU is either busy (executing code) or idle (inactive).

When busy, the CPU may be executing either user code (from a process) or system code (from the kernel). All device drivers and system calls are part of the kernel code.

When idle, the CPU may be waiting for the I/O to complete, or it may truly have nothing to do.

Accordingly, the report generated by **sar –u** has four fields:

%sys
Percentage of time the CPU was executing system code

%usr
Percentage of time the CPU was executing user code

%wio
Percentage of time the CPU was waiting for I/O to complete

%idle
Percentage of time the CPU was truly idle

For example:

```
$ sar -u 10

UnixV  ed  4.0   2.0  i386/486  08/07/91

14:01:49   %usr  %sys  %wio  %idle
14:01:59     66    15    19      0
```

In this example, 66% of the CPU time was spent executing user code, 15% executing system code, 19% waiting for I/O and 0% truly idle.

In typical timesharing, *%sys* and *%usr* should have approximately the same value. In certain applications, however, either of these may be larger than the other without anything being abnormal. If *%usr* is greater than 50%, users are getting excellent service.

The following indicate a CPU utilization problem:

%wio > 7
May indicate a potential disk bottleneck. The file system distribution should be checked to balance I/O across disks, if possible.

%idle < 10
Indicates that the system is having trouble keeping up with the load. If this occurs consistently, you may need to redistribute the load so that the peak time is not as busy. It is possible that tuning will not help; you may have no alternative except offloading work to another system or getting a more powerful processor.

If you use **sar –u** along with the **–P *proc*** option to view CPU utilization on a per processor basis, the kernel does not log values for *%wio*. The *%idle* value is actually the total of idle waiting on I/O and truly idle. The *%wio* is not logged on a per processor basis because if the process running needs I/O which is not available at that time, the processor puts that process on a sleep queue and services another process.

sar –q

The **sar –q** command measures the utilization of the Run Queue.

The Run Queue is a list of all processes that are in memory and ready to run. The kernel uses this list to determine the next process to run.

Two factors determine the utilization of these queues: the size (adequate number of processes in the list) and the percentage of time that the queue had a process listed. Accordingly, the report generated by **sar –q** has four fields.

runq_sz
: Average number of processes in the Run Queue when it was active.

%runocc
: Percentage of time the Run Queue was active.

swpq_sz
: Average number of processes in the Swap Queue when it was active.

%swpocc
: Percentage of time the Swap Queue was active.

For example:

```
$ sar -q 10

UnixV  ed  4.0   2.0  i386/486  08/07/91

14:59:49  runq-sz  %runocc  swpq-sz %spwocc
14:59:59      1.1       81
```

In this example, the Run Queue was occupied 81% of the time with an average of 1.1 processes on the queue and the Swap Queue was not occupied at all.

sar –w

The **sar** –w command measures the swapping and process switching activity.

When it does not have enough space in memory to hold all of the processes on the system, the system swaps (writes) process data from memory to the swap area on disk. Swapping activity reduces performance because the system must spend time performing the data transfer.

Swapping activity consists of swap–outs and swap–ins. A swap–out request transfers data from memory to disk to make room for another process. A swap–in request transfers data from disk into memory, and includes the initial loading of a process.

Each swap request may consist of more than one physical data block.

The system switches between processes when it allocates the CPU to another process. The process switch may occur because the time slice has expired or because the current process requested I/O and must wait. Excessive switching decreases system performance because the CPU spends less time executing user processes.

The swapping activity is measured by the number of swap–in and swap–out requests and the number of blocks transferred per request. The switching activity is measured by the number of process switches. Accordingly, the report generated by **sar –w** has five fields:

swpin/s
: Average number of swap–in requests per second (transfers from the swap area into memory).

pswin/s
: Average number of physical blocks (normally 512–bytes each) transferred into memory per second from the swap device.

swpot/s
: Average number of swap–out requests per second (transfers from memory to the swap device).

bswot/s
: Average number of physical blocks (normally 512–bytes each) transferred from the swap device to memory per second.

pswch/s
: Average number of process switches per second.

For example:

```
$ sar -w 10

ed  ed  4.0   3.0  3550  08/07/91

15:26:47  swpin/s  pswin/s  swpot/s  bswot/s  pswch/s
15:26:57     0.00      0.0     0.00      0.0       12
```

In this example, there was no swap–in or swap–out activity and the number of process switches per second was 12.

The swap–in activity can include the swapping in of processes; therefore, the swap–out activity is used to determine if there is a swapping bottleneck.

The recommended limit for the average number of process switches per second depends on the processor type and the type of load. Most processors can handle 1–3 process switches per second for each user; therefore, if there are 10 users, the average number of process switches should be in the range of 10–30.

The following indicates excessive swapping activity:

swpot/s > 1.0
: Indicates that excessive swap–out activity is occurring. If this occurs consistently, you may need to increase the amount of system memory or decrease the size of the system buffer area (because the buffer area uses a large amount of memory).

sar –p

The **sar –p** command measures the demand paging activity, including protection and validity faults.

NOTE: This command has changed significantly from past releases due to the adoption of Virtual Memory.

With demand paging, a page is loaded into memory when the process references it (on demand). A page is the smallest amount of memory that may be allocated.

The paging activity is measured by the number of page faults per second (page referenced but not in memory), the number of illegal accesses to a page, the number of page faults satisfied by reading the page from the file system, and the number of pages reclaimed for the free list. Accordingly, the report generated by **sar –p** has six fields:

atch/s
: The number of page faults per second that are satisfied by reclaiming a page currently in memory (attaches per second).

pgin/s
The number of times per second the file systems receive page–in requests. (This replaces the old **sar –p** report of *pgfil/s*, which previously reported the validity faults per second satisfied by a page–in from the file system.

ppgin/s
The number of pages paged in per second. (A single page–in request may involve paging–in multiple pages.)

pflt/s
The number of page faults from protection errors (illegal access to a page) or from copy on writes; normally consists entirely of copy on writes. A copy on write is generated when a child process shares pages with its parent.

vflt/s
The number of address translation faults (also known as validity faults) per second (a request was made for a valid page, but the page was not in memory).

slock/s
Reports the number of faults per second caused by software lock requests requiring physical I/O. For example, a softlock request may be the transfer of data from a disk to memory. To ensure that the page which is to receive the data is not claimed and used by another process, it is locked by the system hardware.

For example:

```
$ sar -p 10

ed  ed  4.0   3.0  3550  08/07/91

15:36:22 atch/s  pgin/s  ppgin/s pflt/s  vflt/s  slock/s
15:36:32    .11    0.12     0.12   0.50    .12    .12
```

In this example, the number of page faults per second satisfied by reclaiming a page in memory was .11, the number of times per second the file systems received page–in requests was .12, the number of pages paged in per second was .12, the number of page faults from protection errors or from copy on writes was .50, the number of address translation faults per second was .12, and the number of faults per second caused by software lock requests requiring physical I/O was .12.

The report generated by **sar –p** is normally used for reference only; it is not normally used for performance tuning.

If *vflt/s* becomes much higher than 15, then examine the output of **sar –g** to determine if there is a memory shortage or if the S5 inode freelist is page bound.

sar –r

The **sar –r** command displays the number of available memory pages and the number of blocks available in the swap area.

These two values may be used to determine if the system is swapping information out to the swap area and, if so, how much space is left in the swap area. Accordingly, the report generated by **sar –r** has two fields:

freemem
The number of pages of memory available to user processes.

freeswap
The number of disk blocks available in the swap area (a disk block is normally 512–bytes).

For example:

```
$   sar -r  10

ed  ed  4.0   3.0  3550  08/07/91

15:48:19   freemem   freeswap
15:48:29      1671      16383
```

In this example, the number of available pages of memory is 1671 and the number of free disk blocks in the swap area is 16383.

The **sar –r** command quickly shows if the system is transferring information to the swap area and the amount of free memory and swap space.

sar –c

The **sar –c** command measures the total system call activity and the activity for the most frequently used system calls.

The **read**(2) and **write**(2) system calls generally account for about one–half of the total system calls. The **fork**(2) system call creates a new process and the **exec**(2) system call changes the executable image of a process. These system calls measure the number of new processes on the system. Accordingly, the report generated by **sar –c** has seven fields:

scall/s
The total number of system calls per second

sread/s
The number of **read** system calls per second

swrit/s
The number of **write** system calls per second

fork/s
The number of **fork** system calls per second

exec/s
The number of **exec** system calls per second

rchar/s
The number of characters (bytes) transferred by **read** system calls per second

wchar/s
The number of characters (bytes) transferred by **write** system calls per second

For example:

```
$   sar -c  10

ed  ed  4.0   3.0  3550  08/07/91

15:50:02  scall/s  sread/s  swrit/s  fork/s   exec/s    rchar/s  wchar/s
15:50:12    59        20        9      0.09   0.08       10571     3660
```

In this example, the total number of system calls per second is 59, the number of **read** system calls is 20 and they transferred 10571 characters, the number of **write** system calls is 9 and they transferred 3660 characters, the number of **fork** system calls per second is 0.09 and the number of **exec** system calls per second is 0.08.

The following indicate a potential bottleneck:

scall/s > 3000
: If the system is I/O bound, the *scall/s* value is close to the **pswch/s** value from a **sar –w** command. A system can typically handle 3000 to 3500 system calls (if the system is I/O bound).

fork/s > 30
: The *fork/s* value is typically between 2 and 8 per second for 16 to 64 concurrent processes. This value may be larger when shell scripts are running. If *fork/s* is greater than 20 to 30 per second, a significant amount of system time is being used to create new processes. If this occurs consistently, you may need to redistribute the load so that the peak time is not busy or the applications may be redesigned to not create as many processes.

(exec/s / fork/s) > 3
: If the ratio of *exec/s* to *fork/s* is greater than 3, check for inefficient PATH variables.

sar –g

The **sar –g** command measures the page–out and memory freeing activities.

The report generated by **sar –g** has five fields:

pgout/s
: The number of times per second that system receives page–out requests.

ppgout/s
: The number of pages that are paged–out per second. (A single page–out request may involve paging–out multiple pages.)

pgfree/s
: The number of pages per second that are placed on the freelist by the page–stealing daemon.

pgscan/s
: The number of pages per second scanned by the page–stealing daemon.

%s5ipf
: The percentage of s5 inodes taken off the freelist by iget which had reusable pages associated with them. These pages are flushed and can not be reclaimed by processes. Thus this is the percentage of igets with page flushes.

The following imply insufficient amounts of memory:

pgfree/s > 5
: More memory may be needed.

pgscan/s > 5
: The page–out daemon is spending a lot of time checking for free memory. More memory may be needed.

%s5ipf > 10
: The freelist of inodes is considered to be page–bound and the number of s5 inodes should be increased.

For example, the **sar –g** command produces a report similar to the following:

```
ed  ed  4.0   3.0  3550  08/07/91

14:13:12  pgout/s  ppgout/s  pgfree/s pgscan/s  %s5ipf
14:14:12     0.00      0.00      0.35     8.18    0.00
14:15:12     0.00      0.00      0.00     0.00    0.00
14:16:12     0.00      0.00      0.00     0.00    0.00

Average      0.00      0.00      0.12     2.72    0.00
```

sar –k

The **sar –k** command reports the activities of the Kernel Memory Allocator (KMA).

The KMA allows a kernel subsystem to allocate and free memory as needed. Rather than statically allocating the maximum amount of memory it is expected to require under peak load, the KMA divides requests for memory into three categories:

- Small (less than 256 bytes)
- Large (512 –4K bytes)
- Oversized (greater than 4K)

To satisfy small and large requests, the KMA keeps two pools of memory. It satisfies the oversized requests by allocating memory from the page allocator.

The report generated by **sar –k** has eight fields:

sml_mem
: The amount of memory in bytes the KMA has available in the small memory request pool (a small request is less than 256 bytes).

alloc
: The amount of memory in bytes the KMA has allocated from its small memory request pool to small memory requests.

fail
The number of requests for small amounts of memory that failed.

lg_mem
The amount of memory in bytes the KMA has available in the large memory request pool (a large request is from 512 bytes to 4K bytes).

alloc
The amount of memory in bytes the KMA has allocated from its large memory request pool to large memory requests.

fail
The number of requests for large amounts of memory that failed.

ovsz_alloc
The amount of memory allocated for oversized requests (those greater than 4K). These requests are satisfied by the page allocator; thus, there is no pool.

fail
The number of requests for oversized amounts of memory that failed.

For example, the **sar –k** command produces a report similar to the following:

```
ed  ed  4.0   3.0  3550  08/07/91

14:28:12  sml_mem  alloc  fail lg_mem  alloc fail ovsz_alloc fail
14:29:12    95323  73472     0  311296  198656   0  180224     0
```

sar –b

The **sar –b** command measures the efficiency of the system buffer area used for file I/O requests. The system buffer area is a section of memory reserved to store blocks for file systems to minimize the actual I/O that must occur to read and write files.

The efficiency of the buffer area is determined by the percent of the time the system can avoid I/O to satisfy read and write requests for files. If the requested data block is already in the buffer area, the actual disk I/O does not have to be performed. If the percentage is high, the system buffer area is being used efficiently. If the percentage is low, the system buffer is not being used to its full potential.

If a process uses the raw (character) disk device, the system performs the requested I/O WITHOUT using the system buffer area. Disk I/O requests that do not use the system buffer area are referred to as physical I/O requests.

To calculate the efficiency, the number of actual file I/O transfers must be compared to the number of file I/O requests. To determine the amount of disk I/O performed without the buffer area, you need to know the amount of physical I/O requests. Accordingly, the report generated by **sar –b** has eight fields:

bread/s
: The average number of data transfers per second from disks (or other block devices) to the system buffer area.

lread/s
: The average number of logical read requests per second from the system buffer area (the number of read accesses).

%rcache
: The percentage of logical read requests for which the read was performed from the system buffer area (avoiding the actual I/O).

bwrit/s
: The average number of data transfers per second from the system buffer area to disks (or other block devices).

lwrit/s
: The average number of logical write requests per second to the system buffer area (the number of write accesses).

%wcache
The percentage of logical write requests for which the write was performed only in the system buffer area (avoiding the I/O).

pread/s
The average number of physical I/O read requests (bypassing the system buffer area).

pwrit/s
The average number of physical I/O write requests (bypassing the system buffer area).

For example, the **sar -b 10** command produces a report similar to the following:

```
ed  ed  4.0   3.0  3550  08/07/91

15:50:19  bread/s  lread/s   %rcache   bwrit/s  lwrit/s   %wcache   pread/s   pwrit/s
15:50:29     3       166         98         3        31         92         0         0
```

In this example, the average number of read transfers per second is 3 and the number of logical read requests is 166, producing a read efficiency of 98%. The average number of write transfers per second is 3 and the number of logical write requests is 31, producing a write efficiency of 92%. There were no physical I/O requests.

The values that indicate the efficiency of the system buffer area are *%rcache* and *%wcache*. The higher the value in each of these fields, the more efficient the buffer area is.

The following indicates a potential bottleneck:

%rcache < 90 AND *%wcache* < 80
If the read efficiency of the buffer area is less than 90% and the write efficiency is less than 80%, the system buffer area is not being used as efficiently as possible. If this occurs consistently, you may be able to improve the performance by increasing the size of the system buffer area.

NOTE: The buffer area requires a large amount of memory. Increasing its size may also increase the amount of swapping/paging activity because there is less memory for processes. The swapping/paging activity may decrease performance more than the increase in the buffer area improves it.

sar –d

The **sar –d** command measures the SCSI activity for the block devices (disks and tapes).

To determine the activity for a block device, you need to know the percentage of time the device was busy performing I/O, the amount of data transferred, and the amount of time the requests were delayed by other requests and device positioning. Accordingly, the report generated by **sar –d** has seven fields:

device
The name of the block device. The device specification is system–dependent, however the name is normally displayed as *cwtxdysz*.

%busy
The percentage of time the device was servicing a transfer request (either read or write).

avque
The average number of outstanding requests when there was a request pending for the device.

r+w/s
The average number of read and write transfers per second for the device.

blks/s
The average number of data blocks (normally 512–bytes each) transferred per second for the device.

avwait
The average time in milliseconds that transfer requests wait idle when there are requests pending (this is the time the requests are delayed by other requests).

avserv
The average time in milliseconds for a transfer request to be completed by the device. For a disk, this includes the seek, rotational latency, and data transfer times.

For example, the **sar -d 10** command produces a report similar to the following:

```
ed  ed  4.0   3.0  3550  08/07/91

15:50:27  device              %busy  avque  r+w/s  blks/s  avwait  avserv
15:50:37  /dev/rdsk/c0t5d0s1    18    1.3     46     183     1.3     24.7
          /dev/rdsk/c0t6d0s2     7    1.1      7      27     0.7     24.5
```

In this example, the first disk is 18% busy with an average of 46 read/write requests and 183 blocks transferred per second, and requests wait an average of 1.3 milliseconds on the queue and 24.7 milliseconds for the transfer. The second disk is 7% busy with an average of 7 read/write requests and 27 blocks transferred per second, and requests wait an average of 0.7 milliseconds for other requests and 24.5 milliseconds for the transfer.

The *avque* value shows the number of simultaneous read and write requests the device must service. If *avque* is large (>5) and performance is poor, processes are probably performing a large number of small I/O requests (less than one block).

The *avserv* time is normally equal to the average access time for the disk. The sum of the *avwait* and *avserv* time is the total delay (in milliseconds) to service one I/O request after the system places the request on the device's service queue.

sar –a

The **sar –a** command measures the activity for the file access operations.

The file access operations are the internal kernel routines that are called to locate a file. When a file is accessed, the kernel must search the path name to locate the I–node of the file. This path name search includes the reading of directory data blocks and the I–node of each directory in the path.

The *namei* kernel routine performs the path name search for locating a file. The *iget* routine reads the I–node of a file into memory (*namei* calls *iget* for every I–node it reads). The *dirbk* routine reads a directory data block (*namei* calls *dirbk* for every directory data block it reads).

Accordingly, the report generated by **sar –a** has three fields:

iget/s
The average number of I–nodes read per second.

namei/s
The average number of path name searches per second.

dirbk/s
The average number of directory data blocks read per second.

For example:

```
$   sar -a  10

ed  ed  4.0   3.0  3550  08/07/91

15:50:41     iget/s      namei/s     dirbk/s
15:50:51         23            7          20
```

In this example, the average number of I–nodes read per second was 23, the average number of path name searches was 7, and the average number of directory data blocks read per second was 20.

Each of these values should be kept as small as possible. They may be decreased by changing processes to reference relative path names instead of full path names.

The following indicates a potential bottleneck:

dirbk/s > iget/s
If *dirbk/s* is greater than *iget/s* then many large directories are being searched (directories with more than one data block). If this occurs consistently, check the PATH variable for large directories.

sar –y

The **sar –y** command measures the utilization of terminal lines.

Normal terminal usage consists of three types of data:

raw input
The actual characters typed on a terminal. This includes the correction characters (backspace and "cancel line").

canonical input
Input as it appears after all correction characters are handled.

output
Characters sent to a terminal for output.

Another measure of usage for terminal lines is the number of interrupts caused by input, output, or modem control. At most, each character causes an interrupt.

Accordingly, the report generated by **sar –y** contains six fields:

rawch/s
The average number of raw input characters received from terminal devices per second.

canch/s
The average number of canonical characters handled per second (the number of characters in the canonical input queue after handling the correction characters).

outch/s
The average number of output characters transmitted to terminal devices per second.

rcvin/s
The average number of receive interrupts per second (interrupts caused by input characters).

xmtin/s
The average number of transmit interrupts per second (interrupts caused by output characters).

mdmin/s
The average number of modem interrupts per second.

For example:

```
$   sar -y  10

ed  ed  4.0    3.0  3550  08/07/91

10:18:40 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
            5       2      174       5       81       0
```

In this example, the average number of actual input characters per second is 5, the average number of canonical characters per second is 2, and the number of output characters is 174. The number of input character interrupts is 5, the number of output character interrupts is 81, and there were no modem character interrupts.

The **cu**(1) and **uucp**(1) utilities may be used to transfer data between systems using terminal lines, and therefore may increase the activity for terminal lines. Some general ranges for the I/O rates for a system with 8–10 concurrent users are:

Without **cu** and **uucp** activity:

- Input: 2–10 characters per second
- Output: 30–150 characters per second

With **cu** and **uucp** activity:

- Input: 20–100 characters per second
- Output: 80 –200 characters per second

The following indicate a potential bottleneck:

outch/s > 1000
: If *outch/s* is greater than 1000, output may be overusing the terminal character buffers, causing the input characters to be lost. If input characters are lost, the terminal lines are overloaded. If this occurs, reduce the amount of terminal output.

mdmin/s 0
: If *mdmin/s* is not equal to 0, check for faulty modems, lines, terminals, or ports.

rcvin/s > *rawch/s* OR *xmtin/s* > *outch/s*
: If *rcvin/s* is greater than *rawch/s* or *xmtin/s* is greater than *outch/s*, check for bad terminal lines. The number of receive interrupts per second should be less than or equal to the number of incoming characters and the number of transmit interrupts per second should be less than or equal to the number of outgoing characters.

sar –v

The **sar –v** command reports the usage and any overflows of the following system tables:

- Process Table
- Active I–node Table
- System Open File Table
- System Lock Table

The Process Table keeps track of all processes on the system. This table contains the information that is needed at all times for each process, for example, the process id, parent process id, location of the executable image.

The Active I–node Table keeps track of all I–nodes (files) that are currently in use. Every I–node being used for any reason has an entry in this table, for example, all open files, files being executed, current directories, root directories.

The System Open File Table keeps track of open files. Each **open**(2) or **creat**(2) system call that is executed creates an entry in this file. Entries in this table point to an entry in the Active I–node Table.

The System Lock Table keeps track of locked records in files. Each time a process locks a file or a record in a file, an entry from this table is used.

The sizes of these tables are determined by the system configuration parameters set when the kernel was created. If any of these tables become full, an error occurs (this is referred to as an overflow of the table). If any of these tables are under–utilized, memory may be saved by decreasing their size.

Accordingly, the report generated by **sar –v** contains seven fields:

proc–sz
: The number of entries in use at the time of the sample and the total size of the Process Table.

ov
: The number of overflows in the Process Table since the last sample was taken.

s5inod–sz
: The number of entries in use at the time of the sample and the total size of the Active I–node Table.

ov
The number of overflows in the Active I–node Table since the last sample was taken.

file–sz
The number of file table entries in use in the kernel at the time of the sample and the total size of the System Open File Table. The size is given as 0 because space for the file table is allocated dynamically.

ov
The number of overflows in the System Open File Table since the last sample was taken.

lock–sz
The number of entries in use at the time of the sample and the total size of the System Lock Table.

For example:

```
$   sar -v  10

ed  ed  4.0   3.0  3550  08/07/91

15:51:24  proc-sz   ov  s5inod-sz ov file-sz  ov  lock-sz
15:51:35  16/200    0   0/500      0   45/0    0    2/300
```

In this example 16 out of 200 Process Table entries are in use, 0 out of 500 Active I–node Table entries are in use, 45 System Open File Table entries are in use, and 2 out of 300 System Lock Table entries are in use. None of the system tables has overflowed since the last sample was taken.

To make sure that a table does not overflow, each table should always have approximately 10% of its entries free. If a table overflows at any time, its size should be increased.

If any of these tables consistently use less than 50% of their entries, you may want to decrease the size of the table. This would decrease the amount of memory required for the kernel, and therefore increase the amount of memory available for processes.

sar –m

The **sar –m** command reports the activity for the Inter–Process Communication (IPC) facilities of message queues and semaphores.

The message queue IPC facility permits processes to send and receive messages on a queue. The processes are independent; they may be started by different users, on different terminals.

The semaphore IPC facility permits independent processes to set a flag to prevent simultaneous activity. For example, setting a flag to make sure only one process is updating a shared memory segment.

The report generated by **sar –m** has two fields:

msg/s
: The average number of message queue operations (sends and receives) per second.

sema/s
: The average number of semaphore operations (increment or decrement) per second.

For example:

```
$   sar -m  10

ed  ed  4.0   3.0  3550  08/07/91

13:03:32   msg/s     sema/s
13:03:42    0.31       0.16
```

In this example, the average number of message queue operations per second is 0.31 and the average number semaphore operations per second is 0.16.

These values indicate the activity level for synchronizing the use of shared memory segments and/or cooperating processes.

Analyzing Activity with Process Accounting

You can use the Process Accounting utilities to keep track of user activity, line activity, and the most commonly used commands on the system. Also, you can use the information generated by the accounting utilities to charge users for system time.

The **runacct** command is most commonly used for performance tuning. This command generates five reports:

Daily Report
: Identifies the time period of the accounting report and the utilization for each terminal line. Useful to identify bad terminal lines.

Daily Usage Report
: Identifies the system utilization for each user, including the amount of CPU time used, disk blocks, number of logins, etc. Useful to identify heavy system users.

Daily Command Summary
: Identifies the usage of commands. Useful to identify the most commonly used commands that should be optimized, if possible.

Monthly Command Summary
: Same as the Daily Command Summary, but identifies the usage of commands for the current month (since the last monthly report).

Last Login
: Displays the last date that each login name was used. Useful for identifying unused login names.

Setting Up Process Accounting

Starting Accounting at start–of–day

In the */etc/rc2.d* directory, create a link called *S22acct*, which links to the file */etc/init.d/acct*. This file contains a script that starts process accounting whenever your system goes into multiuser mode:

```
ln /etc/init.d/acct /etc/rc2.d/S22acct
```

Shutting Down Process Accounting

In the */etc/rc0.d* directory, create a script that contains the following command to be performed when the system shuts down:

```
/usr/lib/acct/shutacct
```

Running Daily Accounting

There are three utilities required to perform daily accounting:

runacct
: Produces the five daily reports. The **dodisk** utility must be run before **runacct**.

dodisk
: Generates the disk usage information used by the **runacct** report.

ckpacct
: Checks the size of the process accounting file. If the file is larger than 500 blocks, the file is copied to a backup file. This prevents the file from growing too large.

The **cron** facility is commonly used to perform these utilities. The entries in Figure 1–4 are example entries; they should be added to the **crontab** entries for the **adm** user.

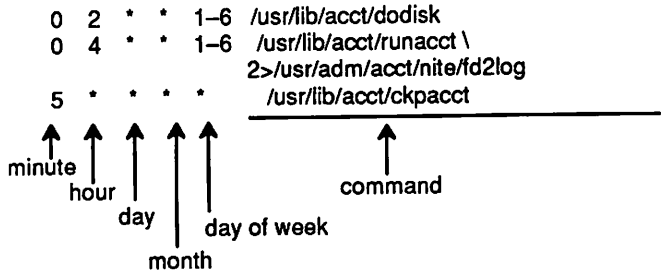


Figure 1–4: Example crontab Entries

The first entry executes the **dodisk** command Monday–Saturday at 2:00 a.m. The second entry executes the **runacct** command Monday–Saturday at 4:00 a.m. The third entry executes the **ckpacct** command at 5 minutes after every hour.

Printing Daily Accounting Reports

The daily accounting reports generated by **runacct** may be printed using the **prdaily** command. For example:

```
# /usr/lib/acct/prdaily | lp
```

Running Monthly Accounting

The **monacct** utility may be used to clean up all the daily reports and daily total accounting files and place one monthly total report and one monthly total accounting file in the */var/adm/acct/fiscal* directory. This report should be run once per month, after the daily **runacct** has executed.

The entry in Figure 1–5 may be added to the **crontab** entries for the **adm** user to execute **monacct** on the first of each month at 5:15 a.m.

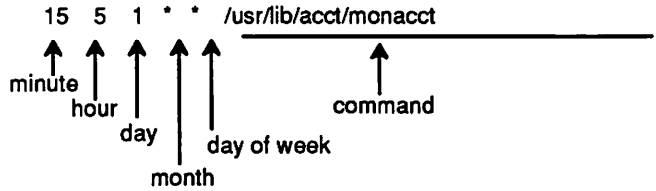


Figure 1–5: Example crontrab Entry for Monacct

Daily Report

The following is an example Daily Report, generated by the **runacct** command.

```
from    Fri  May  12  04:00:02  1991
to      Sat  May  13  04:00:03  1991

TOTAL DURATION IS 1440 MINUTES.
LINE        MINUTES    PERCENT    #SESS    #ON    #OFF
tty00         430         30        4       4        4
tty01         506         35        1       1        1
 .
 .
 .
TOTALS        2033         --       24      24       24
```

The first part of the report identifies the time period for the report. In addition, it would list any reboots, power fail recoveries, or shutdowns that occurred during the time period.

The second part of the report identifies the line utilization. The following describes each field in the report:

TOTAL DURATION
The number of minutes the system was in multiuser mode.

LINE
The terminal line.

MINUTES
The total number of minutes the line was in use.

PERCENT
the percentage of the total duration that the line was in use (MINUTES / TOTAL DURATION).

#SESS
The number of times a user logged in on this line.

#ON
Same as #SESS.

#OFF
The number of times a user logged off on this line AND the number of interrupts that occur on the line. Interrupts generally occur when the getty process is first invoked when the system enters multi–user mode. If #OFF exceeds #ON by a large factor (more than twice as large), it may indicate a bad multiplexor, modem, cable, or connection. Check for an unconnected cable.

Daily Usage Report

The following is an example Daily Usage Report, generated by the **runacct** command.

```
      LOGIN          CPU (MINS)          KCORE-MINS
UID   NAME       PRIME    NPRIME      PRIME    NPRIME
0     TOTAL      1              0      16             2
0     root       37             0      6              0
100   liz        35             5      1              0
104   tom                               1              1
.
.
.
CONNECT (MINS)            DISKS     # OF
PRIME    NPRIME           BLOCKS    PROCS
4100        116           112483    522
30            0           88400     10
430           0           15270     52
440          66           813       60
.
.
.
# OF      # DISK
SESS      SAMPLES       FEE
27        1             0
1         1             0
4         1             0
1         1             0
```

The fields in the report are as follows:

UID
: The numerical user id of the user.

LOGIN–NAME
: The login name of the user.

CPU–(MINS)
: The CPU usage, in minutes, of a user's processes. The time is divided into PRIME and NPRIME (nonprime) utilization. The file */usr/lib/acct/holidays* defines prime time.

KCORE-MINS
The average number of kilobytes of memory used per minute by the user's processes. This time is also divided into PRIME and NPRIME (nonprime) utilization.

CONNECT (MINS)
The amount of time that the user was logged into the system. This time is also divided into PRIME and NPRIME (nonprime) utilization. If CONNECT (MINS) is high and # OF PROCS is low, the user probably logged into the terminal in the morning but did not use the system much for the rest of the day. Users should be encouraged to log off the system when they are not using it.

DISK BLOCKS
The number of disk blocks being used by the user. This field only contains information if the disk accounting facilities (**dodisk**, etc.) have been performed.

OF PROCS
The number of processes invoked by the user.

OF SESS
The number of times the user logged into the system.

DISK SAMPLES
The number of times the disk accounting was run to obtain the average number of DISK BLOCKS listed earlier.

FEE
The total accumulation of charge factors for the user. This field is only valid if the chargefee command is used.

Daily and Monthly Command Summary

The daily and monthly command summaries are the same except that the Daily Command Summary summarizes one day's activity and the Monthly Command Summary summarizes one month's activity.

The following is an example Daily Command Summary, generated by the **runacct** command.

```
                        TOTAL COMMAND SUMMARY

COMMAND     NUMBER     TOTAL          TOTAL          TOTAL
NAME        CMDS       KCOREMIN       CPU-MIN        REAL-MIN

TOTALS      59         7.62           0.16           16.85

ls          7          0.92           0.02           0.04
sh          13         0.79           0.02           0.28
.
.
.
MEAN        MEAN       HOG            CHARS          BLOCKS
SIZE-K      CPU-MIN    FACTOR         TRNSFD         READ

47.00       0.00       0.01           123410         708

50.21       0.00       0.41           11504          99
34.99       0.00       0.08           5457           104
```

The fields in the report are as follows:

COMMAND NAME
: The name of the command. All shell scripts are listed under **sh**. Look for any unusual command names, excessive use of **a.out** or **core**, etc. This can help you locate users who are misusing the system.

NUMBER CMDS
: The total number of times this command was invoked.

TOTAL KCOREMIN
: The total amount of kilobytes of memory used by the command per minute of run time.

TOTAL CPU–MIN
The total CPU time the command accumulated.

TOTAL REAL–MIN
The total real–time the command accumulated (the time that a user waited for the command response). This does not include the time for any invocations that were started in the background.

MEAN SIZE–K
The average of TOTAL KCOREMIN / NUMBER CMDS for this command.

MEAN CPU–MIN
The average of TOTAL CPU–MIN / NUMBER CMDS for this command.

HOG FACTOR
This value gives a relative measure of the total available CPU time used by this command during its execution. It is computed with the formula: (total CPU time) / (elapsed time)

CHARS–TRANSFD
The total count of the number of characters read and written by the command.

BLOCKS READ
The total count of the number of physical blocks read and written by the command.

Last Login Report

The following is an example Last Login Report, generated by the **runacct** command.

```
00-00-00  adm    89-05-16  hybl   91-05-16   root
89-05-16  beth   89-05-16  john   91-05-11   sa
00-00-00  bin    89-05-16  liz    91-05-12   shutdown
89-04-12  connie 89-05-16  mary   91-05-16   startup
89-03-03  david  89-02-20  ncrm   91-05-16   tom
89-05-12  guest  89-03-28  netmgr 91-06-07   tommyd
```

This report identifies the last date that each login name was last used. This information may be used to identify unused logins (and their related files).

Analyzing Specific Processes with time

Many users measure system performance by the response time for their processes, that is, the amount of time it takes the system to complete their process.

The response time for a process is affected by two factors: the amount of actual CPU time the process requires and the other system activity.

When the CPU is executing a process, it is either executing the user's code (code from the executable object of the program) or kernel code (code from the kernel). Kernel code is executed when a process performs a system call (requests a service from the kernel).

The **time**(1) command executes a process and measures the three available execution times for the process. The following times are displayed on standard error:

real
: The elapsed time (actual clock time, or response time).

user
: The amount of time spent executing user code.

sys
: The amount of time spent executing kernel code for the process.

The **time** command may be used to measure the performance of the system before and after a configuration change. For accurate results, the other system activity should be eliminated or consistent for each execution of the command. This may be done by executing the command single–user mode or as the only user process in multi–user mode.

Syntax for the time Command

```
time command
```

where *command* is the command (and all arguments) to be timed.

Example

```
$   time   ls -l > /tmp/list

real  1.4
user  0.3
sys   0.5
```

In this example, the **ls –l** command completed in 1.4 seconds, used 0.3 seconds executing user code and 0.5 seconds executing system code.

Analyzing Specific Processes with timex

The **timex**(1) command executes a command and produces the following:

- The **real, user,** and **sys** times for the process (the same times as the **time**(1) command).
- A process accounting report of the activity for the command and all its children (optional).
- A report of the system activity during the execution of the command (optional).
- A report of the total number of blocks read or written and total characters transferred by the command and all its children (optional).

All output of **timex** is displayed on standard error.

You can use the **timex** command to measure the performance of the system before and after a configuration change. For accurate results, the other system activity should be eliminated or consistent for each execution of the command. This may be done by executing the command in single–user mode or as the only user process in multiuser mode.

The **timex** command provides more information than the **time** command; you may analyze the system activity generated by the command as well as the execution time.

Syntax for the timex Command

```
timex [ options ] command
```

-p
List the process accounting records for the command and all its children.

-o
List the total number of blocks read and written and total characters transferred by the command and all its children.

-s
List the total system activity (not just that due to the command) that occurred during the execution of the command. All of the data items for **sar**(1) are reported.

commands
The command (and all of its arguments) that is to be timed.

System Profiling

The system profiler, **profiler**(1M), facilitates an active study of the operating system. **profiler**(1M) consists of the following programs, each having a separate and distinct function.

prfld
: Initializes the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as read from the symbol table in the kernel.

prfstat
: Permits you to turn profiling on and off. It also reveals the number of text addresses being measured.

prfdc and **prfsnap**
: Copies the current value of all text address counters to a file where the data can be analyzed.

prfpr
: Formats the collected data, converting each text address to the nearest text symbol. If the percent activity for that range is greater than the specified limit, **prfpr** prints the address.

When profiling is enabled, the kernel checks at each clock interrupt to see what is executing. If a kernel routine is executing, the kernel searches the stored address file and increments the corresponding counter.

The profiler uses a pseudo driver that requires a kernel reconfiguration. Refer to **prf**(7). No additional hardware is required.

Caution

Before you begin kernel reconfiguration and change any files, SAVE THE ORIGINAL FILES.

Making Profiling Functional

Perform the following steps to make profiling functional:

Step 1: Add an entry to the */etc/conf/node.d/prf* file which contains the following line:

```
prf   prf   c   0   0   3   644
```

Step 2: Change the second field in */etc/conf/sdevice.d/prf* from N to Y.

Step 3: Rebuild the kernel.

```
# /etc/conf/bin/idbuild
```

Step 4: Reboot the system.

```
#  shutdown  -i6  -g0  -y
```

NOTE: If you can not perform profiling, increase the value of the PRFMAX configuration parameter in the */etc/conf/cf.d/stune* file and repeat Steps 3 and 4.

Performing Profiling

You must perform the following steps to perform profiling:

Step 1: Perform the steps for "Making Profiling Functional."

Step 2: Invoke **prfld** to initialize the recording mechanism.

```
# /etc/prfld
```

Step 3: Turn on profiling.

```
# /etc/prfstat  on
```

Step 4: Record counters using either **prfdc** or **prfsnap**.

Step 5: Turn off profiling.

```
# /etc/prfstat  off
```

NOTE: Since profiling affects system performance, turn off profiling as soon as data collection is complete.

Step 6: Format the data collected in file(s) created by **prfpr** in Step 3.

Chapter 2 Process Scheduling

This chapter is addressed to administrators of systems that include the System V Release 4 scheduler.

Using the Process Scheduler 2–2

Configuring the Scheduler 2–8

Changing Scheduler Parameters 2–21

Using the Process Scheduler

Understanding the Process Scheduler

The UNIX system scheduler determines when processes run. It maintains process priorities based on configuration parameters, process behavior, and user requests; it uses these priorities to assign processes to the CPU.

NCR UNIX SVR4 MP–RAS gives users absolute control over the order in which certain processes run and the amount of time each process may use the CPU before another process gets a chance.

There are at least two reasons why administrators should understand the scheduler:

- The scheduler has an overriding effect on the performance and perceived performance of a system. The default scheduler is tuned to perform well in representative work environments, but you must understand how it operates to know whether you can reconfigure it to better suit local needs.
- A bug in a real time program or a malicious real–time user can lock out all other processing, including kernel processing. Users need root permission to create real–time processes, and presumably only trustworthy users have this permission. But administrators still should be aware that the scheduler functions introduce new ways to cause trouble, and should be prepared for accidents and for misuse of these functions.

For programming information on the scheduler, see the *UNIX SVR4 Integrated Software Developer Guide*. The primary user command for controlling process scheduling is **priocntl (1)**, which is described in the *User's Reference Manual*. The primary function call for controlling process scheduling is *priocntl(2)*, which is described in the *Programmer's Reference Manual*.

By default, the scheduler uses a time–sharing policy like the policy used in previous releases. A time–sharing policy adjusts process priorities dynamically in an attempt to provide good response time to interactive processes and good throughput to processes that use a lot of CPU time.

The scheduler offers a real–time scheduling policy as well as a time–sharing policy. Real–time scheduling allows users to set fixed priorities on a per–process basis. The highest–priority real–time user process always gets the CPU as soon as it is runnable, even if system processes are runnable. An application can therefore specify the exact order in which processes run. An application may also be written so that its real–time processes have a guaranteed response time from the system.

For most UNIX environments, the default scheduler configuration works well and no real–time processes are needed: administrators should not change configuration parameters and users should not change scheduler properties of their processes. However, when the requirements for an application include strict timing constraints, real–time processes sometimes provide the only way to satisfy those constraints.

NOTE: Real–time processes used carelessly can have a dramatic negative effect on the performance of time–sharing processes.

Figure 2–2 shows how the System V Release 4 process scheduler works:

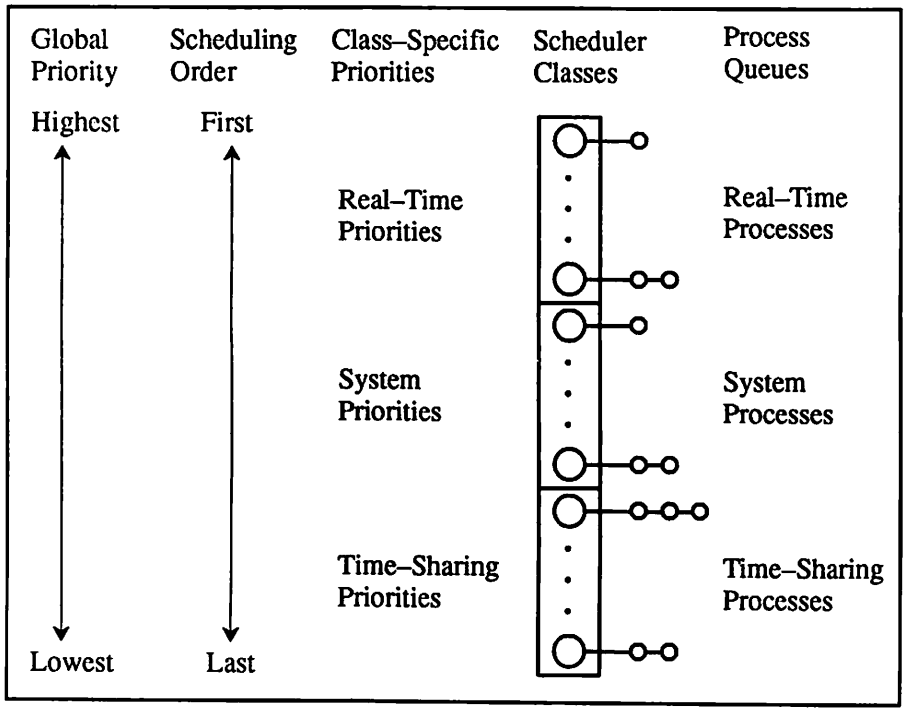


Figure 2–2: How the Process Scheduler Works

When a process is created, it inherits its scheduler parameters, including scheduler class and a priority within that class. A process changes class only as a result of a user request. The system manages the priority of a process based on user requests and a policy associated with the scheduler class of the process.

In the default configuration, the initialization process belongs to the time–sharing class. Because processes inherit their scheduler parameters, all user login shells begin as time–sharing processes in the default configuration.

The scheduler converts class–specific priorities into global priorities. The global priority of a process determines when it runs—the scheduler always runs the runnable process with highest global priority. Numerically higher priorities run first. Once the scheduler assigns a process to the CPU, the process runs until it uses up its time slice, sleeps, or is preempted by a higher–priority process. Processes with the same priority run round–robin.

Administrators specify default time slices in the configuration tables, but users may assign per–process time slices to real–time processes.

You can display the global priority of a process with the **–cl** options of the **ps(1)** command. You can display configuration information about class–specific priorities with the **priocntl(1)** command and the **dispadmin(1M)** command.

By default, all real–time processes have higher priorities than any kernel process, and all kernel processes have higher priorities than any time–sharing process.

NOTE: As long as there is a runnable real–time process, no kernel process and no time–sharing process runs.

The following sections describe the scheduling policies of the three default classes:

- Time–Sharing Class
- System Class
- Real–Time Class

Time–Sharing Class

The goal of the time–sharing policy is to provide good response time to interactive processes and good throughput to CPU–bound processes. The scheduler switches CPU allocation frequently enough to provide good response time, but not so frequently that it spends too much time doing the switching. Time slices are typically on the order of a few hundred milliseconds.

The time–sharing policy changes priorities dynamically and assigns time slices of different lengths. The scheduler raises the priority of a process that sleeps after only a little CPU use (a process sleeps, for example, when it starts an I/O operation such as a terminal read or a disk read); frequent sleeps are characteristic of interactive tasks such as editing and running simple shell commands. On the other hand, the time–sharing policy lowers the priority of a process that uses the CPU for long periods without sleeping.

The default time–sharing policy gives larger time slices to processes with lower priorities. A process with a low priority is likely to be CPU–bound. Other processes get the CPU first, but when a low–priority process finally gets the CPU, it gets a bigger chunk of time. If a higher–priority process becomes runnable during a time slice, however, it preempts the running process.

The scheduler manages time–sharing processes using configurable parameters in the time–sharing parameter table **ts_dptbl**. This table contains information specific to the time–sharing class.

System Class

The system class uses a fixed–priority to run kernel processes such as servers and housekeeping processes like the paging demon. The system class is reserved for use by the kernel; users may neither add nor remove a process from the system class. Priorities for system class processes are set up in the kernel code for those processes; once established, the priorities of system processes do not change. (User processes running in kernel mode are not in the system class.)

Real–Time Class

The real–time class uses a fixed–priority scheduling policy so that critical processes can run in predetermined order. Real–time priorities never change except when a user requests a change. Contrast this fixed–priority policy with the time–sharing policy, in which the system changes priorities in order to provide good interactive response time.

Privileged users can use the **priocntl** command or the **priocnt** system call to assign real–time priorities.

The scheduler manages real–time processes using configurable parameters in the real–time parameter table **rt_dptbl**. This table contains information specific to the real–time class.

Configuring the Scheduler

The default configuration includes both the time–sharing and the real–time scheduler classes. The time–sharing class is tuned for representative UNIX system workloads. Such workloads have a high proportion of interactive processes, which sleep early and often. The real–time class is configured for applications that need it.

For traditional time–sharing uses such as software development, office applications, and document production, real–time processes may be unnecessary. In addition, they may be undesirable. First, they consume memory that cannot be paged: the u–blocks of real–time processes are never paged out. Second, they introduce new ways to cause performance problems: a high–priority real–time process can block out all other processing. In a computing environment where only time–sharing is needed, you may want to remove the real–time scheduler class from your configuration, as described later in this chapter.

On the other hand, if a machine is running an application that has strict requirements on the order in which processes must run, then the real–time scheduler class provides the only way to guarantee that those requirements are met.

NOTE: Real–time processes can have a dramatic negative effect on time–sharing performance.

This section describes the parameters and tables that control scheduler configuration, and tells how to reconfigure the scheduler. A basic assumption is that your workload is reasonable for your system resources, such as CPU power, primary memory, and I/O capacity. If your workload does too much computation or too much I/O for your hardware, reconfiguring the scheduler won't help.

Default Global Priorities

Figure 2–3 shows the scheduling order and global priorities for each scheduler class.

Scheduling Order	Global Priority	Scheduler Class
first	159 . . . 100	Real–Time
	99 . . . 60	System
last	59 . . . 0	Time–Sharing

Figure 2–3: Scheduling Order and Global Priorities

When your system is built, it constructs this information from the tunable parameters and scheduler parameter tables described in the following sections. Although you are not forced to configure scheduler classes to produce consecutive, non–overlapping global priorities like the default priorities, we recommend that you do so for the sake of simplicity. Likewise, we recommend that you make all real–time global priorities greater than the global priorities of all other classes. These conventions simplify scheduler configuration, and they should be able to accommodate any requirements on the scheduler.

Kernel processes such as the swapper and the paging demon run in the system scheduler class. Kernel processes must compete with user processes for CPU time, and in the default configuration all real–time processes have higher priorities than all system processes. Therefore, real–time applications must be written carefully to ensure that the kernel gets the processing time it needs. Also, if you reconfigure the scheduler, make sure that the system class gets enough priority over the time–sharing class to give kernel processes the CPU time they need.

Tunable Parameters

This section describes the tunable parameters that control scheduler configuration. These parameters are specified in files in the */etc/master.d* directory.

The following parameters are specified in the *kernel* file:

- MAXCLSYSPRI is the maximum global priority of processes in the system class. When the kernel starts system processes, it assigns their priorities using MAXCLSYSPRI as a reference point.

  NOTE: MAXCLSYSPRI must be 39 or greater, because the kernel assumes it has at least that great a range of priorities below MAXCLSYSPRI. If you request a MAXCLSYSPRI below 39, it is changed to 39.

  The most important system processes get global priorities at or near MAXCLSYSPRI; the least important system processes get global priorities at or near (MAXCLSYSPRI – 39). The default value of MAXCLSYSPRI is 99, which gives all system processes higher priorities than all user processes.

- INITCLASS is the scheduler class assigned to the *init* process. This scheduler class is inherited by all descendants of *init*, which normally include all user login shells. By default, INITCLASS is TS; that is, all login shells are time–sharing processes in the default configuration.

- SYS_NAME is the character string name of the system scheduler class. The default value of SYS_NAME is SYS.

The following parameters are specified in the *ts* file, which controls the time–sharing policy:

- TSMAXUPRI specifies the range within which users may adjust the priority of a time–sharing process using the **priocntl** system call: the valid range is –TSMAXUPRI to +TSMAXUPRI. The default value of TSMAXUPRI is 20. (Configuring a value of 20 emulates the behavior of the older, less general scheduler interfaces **nice** and **setpriority**, which continue to work as in previous release.)

  The value of TSMAXUPRI is independent of the configured number of global time–sharing priorities, though we recommend configuring at least 40 time–sharing priorities, as explained below in the section on **ts_dptbl**. In the default configuration, there are 60 time–sharing priorities, but users may adjust their priorities only within a range of –20 to +20. The system may use the remaining priorities depending on process behavior.

- NAMETS specifies the character string name of the time–sharing scheduler class. This name is returned by the **priocntl** system call and it is assigned to the tunable parameter INITCLASS to specify the default scheduler class for user processes. The default value of NAMETS is TS.

The following parameter is specified in the **rt** file, which controls the real–time policy:

- NAMERT specifies the character string name of the real–time scheduler class. The default value of NAMERT is RT.

Real–Time Parameter Table rt_dptbl

The scheduler uses **rt_dptbl(4)**, the real–time scheduler (or dispatcher) parameter table, to manage real–time processes. A default version of **rt_dptbl** is delivered with the system, and an administrator may change it to suit local needs. **rt_dptbl** is specified in the **rt** file in the *master.d* directory. It is built into the kernel as part of system configuration if the *system* file contains the line

```
INCLUDE:RT
```

You may adjust the size and values of **rt_dptbl** depending on the applications on your system. Here is part of a simple **rt_dptbl**:

```
rt_glbpri       rt_qntm

100,            100,
101,            80,
102,            60,
103,            40,
104,            20,
105,            10
```

- The **rt_glbpri** column contains global priorities (the priorities that determine when a process runs). Higher numbers run first.

- The **rt_qntm** column contains the default time slice (or quantum) associated with the priority in the **rt_glbpri** column; this is the maximum amount of time a process with this priority may use the CPU before the scheduler gives another process a chance. This time slice is specified in clock ticks. (The system clock ticks HZ times per second, where HZ is a hardware–dependent constant defined in the **param.h** header file.)

The highest priority specified in this table is 105, so processes with priority 105 always run before any other processes. If it does not sleep, a process with priority 105 runs for 10 clock ticks before the scheduler looks for another process to run. (Because 105 is the highest priority, a process at this priority would be preempted after its time slice only if there were another process with priority 105.) Processes at priority 104 run for 20 clock ticks, and so on. The lowest real–time priority specified in this table is 100; a process with priority 100 runs for 100 clock ticks.

The default real–time priority is the lowest priority configured in **rt_dptbl**. This is the priority assigned to a process if it is changed to real–time process and no priority is specified. This is also the priority assigned to the **init** process and all its children if INITCLASS is set to RT.

Though **rt_dptbl** contains default time slices for real–time priorities, users with the appropriate privilege can set real–time priority and time slice independently. Users can specify any time slice they want for a real–time process, including an infinite time slice. The system assumes that real–time processes voluntarily give up the CPU so other work can get done.

Time–Sharing Parameter Table ts_dptbl

The scheduler uses **ts_dptbl(4)**, the time–sharing scheduler (or dispatcher) parameter table, to manage time–sharing processes. A default version of **ts_dptbl** is delivered with the system, and an administrator may change it to suit local needs. Save a backup of the default version of **ts_dptbl**. **ts_dptbl** is specified in the *ts* file in the *master.d* directory. It is automatically built into the kernel as part of system configuration.

You may change the size and values of **ts_dptbl** depending on your local needs, but only experienced administrators should make such changes. The default values have a long history of good performance over a wide range of environments. Changing the values is not likely to help much, and inappropriate values can have a dramatic negative effect on system performance.

If you do decide to change **ts_dptbl**, we recommend that you include at least 40 time–sharing global priorities. A range this large gives the scheduler enough leeway to distinguish processes based on their CPU use, which it must do to give good response to interactive processes. The default configuration has 60 time–sharing priorities. Here is part of a simple **ts_dptbl**:

```
glbpri    qntm      tqexp    slprt    mxwt    lwt6

0,        100,      0,       1,       5,      1,
1,        90,       0,       2,       5,      2,
2,        80,       1,       3,       5,      3,
3,        70,       1,       4,       5,      4,
4,        60,       2,       5,       5,      5,
5,        50,       2,       6,       5,      6,
6,        40,       3,       7,       5,      7,
7,        30,       3,       8,       5,      8,
8,        20,       4,       9,       5,      9,
9,        10,       4,       9,       5,      9,
```

- The **glbpri** column contains global priorities (the priorities that determine when a process runs). Higher numbers run first.

  In the table above, the global priorities run from a high of 9 to a low of 0.

- The **qntm** column contains the time slice (or quantum) associated with the priority in the **glbpri** column; this is the maximum amount of time a process with this priority may use the CPU before the scheduler gives another process a change. This time slice is specified in clock ticks. (The system clock ticks HZ times per second, where HZ is a hardware–dependent constant defined in the */usr/include/sys/param.h* header file.)

  In the table above, time slices run from 10 clock ticks for the highest–priority processes to 100 clock ticks for the lowest–priority processes.

- The **tqexp** column determines the new process priority for a process whose time slice expires before it sleeps. If a process at the priority in the **glbpri** column uses its whole time slice without sleeping, the scheduler changes its priority using the **tqexp** column as an index back into **ts_dptb**: the new priority is the global priority in the **tqexp** position in **ts_dptbl**. (In the default configuration, the index of an entry in **ts_dptbl** happens to match the global priority of that entry. However, this match is not necessary.)

  It is usually reasonable to lower the priority of a time–sharing process whose time slice expires, because the process is too CPU–bound for its current priority. A long, CPU–intensive process is an extreme example of such a process, and its priority should usually be lowered in favor of processes that sleep after a little CPU use, which are more likely to be interactive processes.

In the table above, process priorities are cut roughly in half when a time slice expires. The lowest priority (0) stays at 0, priority 1 is reduced to 0, priorities 2 and 3 are reduced to 1, and so on.

- The **slprt** column gives the priority assigned to a process when it returns from a sleep. A process may sleep voluntarily, as it does when it makes certain system calls, or involuntarily, as it does when the kernel puts it to sleep after a page fault, for example. It is usually reasonable to raise the priority of a process that has slept. It has shown desirable behavior (it gave up the CPU), so we reward it by giving it a higher priority when it awakes.

  In the table above, process priorities are incremented by 1 after they sleep, except that the highest time–sharing priority (9) stays the same.

- The **mxwt** column specifies the number of seconds a process can remain runnable before having its priority changed. (The priority is changed using the **lwt** column.

  In the table above, all priorities are recalculated after a wait of 5 seconds.

- The **lwt** column gives the new priority for a process that is runnable for **mxwt** seconds without getting its full time slice. It is usually reasonable to raise the priority of a process that is not getting any CPU time.

  In the example, process priorities are incremented by 1 when they have been runnable for 5 clock ticks, except that the highest priority time–sharing priority (9) stays the same.

The default global priority of a time–sharing process is the priority in the middle of **ts_dptbl**. This is the priority assigned to a process if it is changed to a time–sharing process with default parameters. This is also the priority initially assigned to the **init** process if INITCLASS is set to TS. The descendants of **init**, normally including all login shells and other user processes, inherit its class and current scheduler parameters.

Kernel–Mode Parameter Table ts_kmdpris

The scheduler uses the kernel–mode parameter table **ts_kmdpris** to manager sleeping time–sharing processes. A default version of **ts_kmdpris** is delivered with the system, and there is seldom a reason to change it. **ts_kmdpris** is specified in the *ts* file in the *master.d* directory. It is automatically built into the kernel as part of system configuration.

NOTE: The kernel assumes that it has at least 40 priorities in **ts_kmdpris**. It panics if it does not.

The kernel–mode parameter table is a one–dimensional array of global priorities. The kernel assigns these priorities to sleeping processes based on their reasons for sleeping. If a user process sleeps because it is waiting for an important resource, such as an inode, it sleeps at a priority near the high end of the **ts_kmdpris** priorities, so that it may get and free the resource quickly when the resource becomes available. If a user process sleeps for a less important reason, such as a wait for terminal input, it sleeps at a priority near the low end of the **ts_kmdpris** priorities.

The default kernel–mode parameter table is simply a one–dimensional array of the integers from 60 through 99, which means that time–sharing processes sleep at priorities between the default real–time priorities and the default time–sharing priorities.

In the default configuration, the priorities in **ts_kmdpris** happen to be exactly the same as the priorities used by system class processes, because the tunable parameter MAXCLSYSPRI is the same as the highest priority in **ts_kmdpris**. This overlap is designed to be consistent with the scheduler behavior of previous releases of the UNIX system, because these priorities produce good performance in most environments. But the overlap is not necessary. The System V Release 4 scheduler introduces a logical separation between the priorities of system processes and sleeping time–sharing processes; an administrator may configure a machine so that the two sets of processes have different ranges of global priorities.

Changing Scheduler Configuration

Changing scheduler configuration requires changing one or more of the tunable parameters or the configuration tables **rt_dptbl**, **ts_dptbl**, and **ts_kmdpris**. You can change any of these by changing the appropriate file in the *master.d* directory and rebuilding the kernel as described in the "Performance Management" chapter. Changes made in this way are permanent. This is the only way to change the size of the configuration tables.

See the "Changing Scheduler Parameters" section for a way to make a temporary change on a running system.

Removing a Scheduler Class

For systems that do not need real–time processes, it may make sense to remove the real–time class, thereby making it impossible to create real–time processes. By not having real–time processes, you avoid their non–pageable u–blocks and you avoid the possibility of a runaway process monopolizing the machine. To remove the real–time scheduler class:

- Replace the INCLUDE:RT line from the */etc/system* file with:

  ```
  EXCLUDE:RT
  ```

- Rebuild the kernel.

There should be no reason to remove the time–sharing scheduler class. An application can put its crucial processes into the real–time class, and thereby ensure that they always run before any time–sharing process. To remove the time–sharing class, perform the following steps:

Step 1: Replace the INCLUDE:TS line from the */etc/system* file with:

```
EXCLUDE:TS
```

Step 2: In the *kernel* file in *master.d*, change INITCLASS to RT. This makes **init** and all its descendants real–time processes.

Step 3: Rebuild the kernel.

Installing a Scheduler Class

By default, both the time–sharing and the real–time scheduler classes are installed. Therefore, you need to install a class only if you first remove it.

To install the time–sharing class:

Step 1: Make sure that the TS module is in the */boot* directory.

Step 2: Insert the INCLUDE:TS line in the *system* file. (The TS module is automatically configured unless it is explicitly EXCLUDEd.)

Step 3: Build the kernel.

To install the real–time class:

Step 1: Make sure the the RT module is in the */boot* directory.

Step 2: Insert the INCLUDE:RT line is in the *system* file. (The RT module is not configured unless it is explicitly INCLUDEd.)

- Build the kernel.

When you re–install a scheduler class, you should also check the value of the tunable parameter INITCLASS to make sure that your configuration is assigning the default scheduler class you want.

Changing Scheduler Parameters

The **dispadmin(1M)** command allows you to change or retrieve scheduler information in a running system. Changes made using **dispadmin** do not survive a reboot. To make permanent configuration changes, you must change the scheduler parameter tables in the *master.d* directory as described in the section above on configuration. However, you can use **dispadmin** to get an effect equivalent to changing configuration tables by calling **dispadmin** from a startup script that changes the configuration automatically at boot time.

The **dispadmin** command has three forms:

- **dispadmin –l** lists the configured scheduler classes.
- **dispadmin –g** [–r *res*] **–c** *class* gets scheduler parameters for the specified class. By default, time slices are printed in milliseconds. You may optionally retrieve time slices at a resolution specified by the **–r** option.
- **dispadmin –s** *config_file* **–c** *class* sets scheduler parameters for the specified class from *config_file* (your configuration file).

Here is the output of the –l option for the default configuration.

```
$ dispadmin -l
CONFIGURED CLASSES

SYS  (System Class)
TS    (Time Sharing)
RT    (Real Time)
```

The following screen shows part of the output of **dispadmin –g** for the real–time class:

```
$ dispadmin -c RT -g
# list real-time parameters
# Real Time Dispatcher Configuration
RES=1000

# TIME QUANTUM    PRIORITY LEVEL
# (rt_quantum)

1000          #             0
1000          #             1
1000          #             2
1000          #             3
1000          #             4
1000          #             5
1000          #             6
1000          #             7
1000          #             8
1000          #             9
 800          #             10
 800          #             11
 800          #             12
 800          #             13
 800          #             14
 800          #             15
 800          #             16
 800          #             17
 800          #             18
 800          #             19
 ...                       ...
 100          #             50
 100          #             51
 100          #             52
 100          #             53
 100          #             54
 100          #             55
 100          #             56
 100          #             57
 100          #             58
 100          #             59
```

The following screen shows part of the output of **dispadmin –g** for the time–sharing class:

```
$ dispadmin -c TS -g   # list time-sharing parameters
# Time Sharing Dispatcher Configuration
RES=1000

#ts_quantum ts_tqexp ts_slpret ts_maxwait ts_lwait PRIORITY LEVEL
1000            0       10          5          10        #        0
1000            0       11          5          11        #        1
1000            1       12          5          12        #        2
1000            1       13          5          13        #        3
1000            2       14          5          14        #        4
1000            2       15          5          15        #        5
1000            3       16          5          16        #        6
1000            3       17          5          17        #        7
1000            4       18          5          18        #        8
1000            4       19          5          19        #        9
 800            5       20          5          20        #       10
 800            5       21          5          21        #       11
 800            6       22          5          22        #       12
 800            6       23          5          23        #       13
 800            7       24          5          24        #       14
 800            7       25          5          25        #       15
 800            8       26          5          26        #       16
 800            8       27          5          27        #       17
 800            9       28          5          28        #       18
 800            9       29          5          29        #       19
 ...           ..      ...         ...         ...                ...
 100           40       55          5          55        #       50
 100           41       55          5          55        #       51
 100           42       56          5          56        #       52
 100           43       56          5          56        #       53
 100           44       57          5          57        #       54
 100           45       57          5          57        #       55
 100           46       58          5          58        #       56
 100           47       58          5          58        #       57
 100           48       59          5          59        #       58
 100           49       59          5          59        #       59
```

By default, **dispadmin** reports time slices in milliseconds. If you specify the **–r** *res* option, **dispadmin** reports time slices in units of *res* intervals per second. For example, a *res* of 1000000 reports time slices in microseconds (millionths of a second).

The –s *config_file* option uses *config_file* to set scheduler parameters for the specified class. The configuration file must be in the class–specific format produced by the **–g** option. The meanings of the parameters are described in the sections above on the scheduler parameter tables; **ts_dptbl** hold time–sharing parameters and **rt_dptbl** holds real–time parameters.

The following examples show how to set the parameters for the default classes as specified in the configuration files **rt_config** and **ts_config**. The examples presuppose that these two files are in the correct formats.

```
$ dispadmin -c RT -s rt_config  # set real-time parameters
$ dispadmin -c TS -s ts_config  # set time-sharing parameters
```

The files that specify the new scheduler parameters must have the same number of priority levels as the current table that is being overwritten. To change the number of priority levels, you must change the *ts* file or the *rt* file in the *master.d* directory as described in the section above on configuration.

Chapter 3

Configuring Kernels

This chapter contains the following information and procedures necessary to configure new kernels:

Configuration Files and Directories 3–2

Creating New Kernels 3–11

Loading/Booting the Kernel 3–15

Modules 3–17

Configuration Parameters 3–25

Configuration Files and Directories

In NCR UNIX SVR4 MP–RAS, configuration information is located in a number of files rather than in one master file. All configuration files except the */etc/.new_unix* file are located under the */etc/conf* directory.

Figure 3–1 shows the location of the configuration files and directories within the file system hierarchy. The files and directories are described in detail on the pages following the diagram.

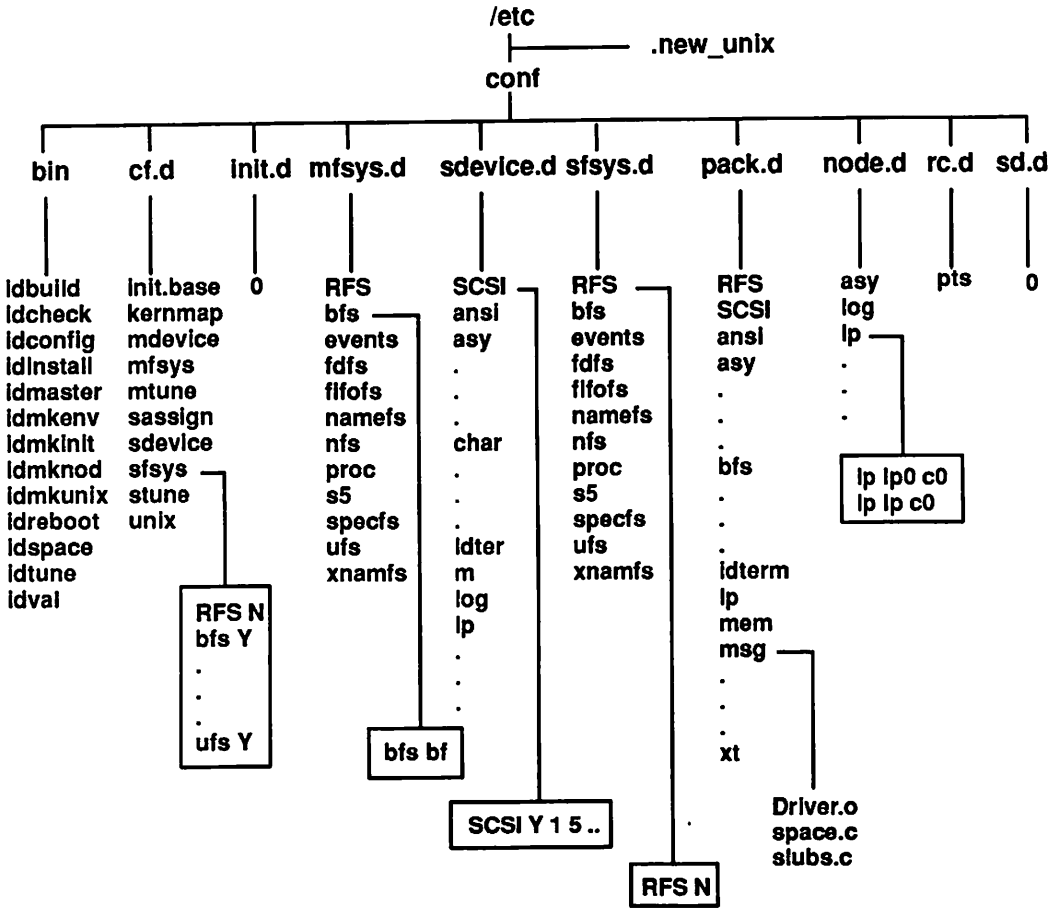


Figure 3–1: Configuration Files And Directories

The */etc/conf* Directory

The */etc/conf* directory contains the following subdirectories that, in turn, contain the system configuration files.

bin
: Contains the commands used in reconfiguration.

cf.d
: Contains the master and system configuration files used by the **idbuild** command to create a new kernel.

init.d
: Contains copies of init modules placed there by the Driver Software Package (DSP).

mfsys.d
: Contains master file system type information, one file for each file system type.

sdevice.d
: Contains system device driver information, one file for each driver.

sfsys.d
: Contains system file system information, one file for each file system type.

pack.d
: Contains subdirectories that contain kernel, file system, and device driver object modules.

node.d
: Contains node specifications which are copies of modules installed by the device Driver Software Package (DSP).

rc.d
: Contains any rc components placed there by the **idinstall** command.

sd.d
: Contains any shutdown components placed there by the **idinstall** command.

The */etc/conf/bin* Directory

The */etc/conf/bin* directory contains the commands used in reconfiguration, including commands to make new kernels, inittab files, and device nodes.

The commands in */etc/conf/bin* are:

idbuild(1M)
Builds a new kernel as follows:

- Concatenates information from the files in the */etc/conf/sdevice.d*, */etc/conf/sfsys.d*, and */etc/conf/mfsys.d* directories into the files */etc/conf/cf.d/sdevice*, */etc/conf/cf.d/sfsys* and */etc/conf/cf.d/mfsys*.
- Calls the **idconfig** command which creates new configuration files using the master and system files in the */etc/conf/cf.d* directory.
- Calls the **idmkunix** command which creates a new kernel in */etc/conf/cf.d/unix*. The **idmkunix** command uses the configuration files created by **idconfig** as well as the *object.o*, *space.c*, and *stubs.c* files in the */etc/conf/pack.d/** module subdirectories.
- Creates the */etc/.new_unix* file, an empty file used as a flag to cause the new kernel to be copied to */stand/unix* at shutdown.

idcheck(1M)
Returns selected information about configuration, such as whether a particular driver is installed or an interrupt vector is available.

idconfig(1M)
Creates new configuration files using the master and system files in the */etc/conf/cf.d* directory. These new configuration files are then used by **idmkunix** to create a new kernel.

idinstall(1M)
Adds, deletes, updates, or gets device driver configuration data.

idmaster(1M)
Updates the */etc/conf/cf.d/mdevice* file by adding or deleting entries.

idmkenv(1M)
Updates the */etc/idrc.d* and */etc/idsd.d* directories, the */etc/inittab* file, and the device nodes in */dev*.

Called by the **rc2** script when the system enters multi–user mode, **idmkenv** then calls **idmknod** and **idmkinit** to update */dev* and */etc/inittab*. After performing the updates, **idmkenv** removes */etc/.new_unix*. Another function of **idmkenv** is to recover the kernel environment if reconfiguration is aborted due to a power loss or someone pressing the power switch.

idmkinit(1M)
Processes files in the */etc/conf/init.d* directory and incorporates that information with the */etc/conf/cf.d/init.base* file to create a new *inittab* file in the */etc/conf/cf.d* directory.

idmkinit is called by **idmkenv** from the **rc2** script during the first system reboot after a new kernel is made. After **idmkinit** runs, **idmkenv** copies the new */etc/conf/cf.d/inittab* file to */etc/inittab* to establish the correct */etc/inittab* file for the new kernel.

idmkinit can be called as a user level command to test modification of *inittab* before a Driver Software Package is actually built. It is also useful in installation scripts that do not reconfigure the kernel, but need to create *inittab* entries. In this case, the *inittab* file generated by **idmkinit** must be copied to */etc/inittab* and a **telinit q** command must be run to make the new entry take affect.

idmknod(1M)
Processes files in the */etc/conf/node.d* directory to establish the correct number of device nodes in */dev*.

idmknod is called by **idmkenv** from the **rc2** script during the first system reboot after a kernel reconfiguration.

idmknod can be called as a user level command to test modification of the */dev* directory before a Driver Software Package is actually built. It is also useful in installation scripts that do not reconfigure the kernel, but need to create */dev* entries.

idmkunix(1M)
Creates a new bootable UNIX kernel (*/etc/conf/cf.d/unix*). **idmkunix** uses the configuration files created by **idconfig** as well as the *object.o*, *space.c*, and *stubs.c* files in the */etc/conf/pack.d/** module subdirectories.

idreboot(1M)
Reboots the system after the installation of drivers.

idspace(1M)
Investigates free space in */*, */usr*, and */tmp* to determine whether enough disk blocks and inodes exist in each of these three (potential) file systems.

idtune(1M)
Sets the value of a specified tunable parameter in the */etc/conf/cf.d/stune* file. The **idtune** command can be run by **root** or by a program such as an install script for an application package. Figure 3–2 shows the actions of **idtune** according to the options parameter values specified.

The syntax for the **idtune** command is:

```
/etc/conf/bin/idtune [ -f | -m ] parameter_name value
```

Option	New Value Specified	Change	Message Displayed
None	> Old value	Depends on response (**y** or **n**) to message	Yes
None	< Old value	Depends on response (**y** or **n**) to message	Yes
–m	> Old value	Yes	No
–m	< Old value	No	No
–f	> Old value	Yes	No
–f	< Old value	Yes	No

Figure 3–2: Options and Actions of the **idtune** Command

idval(1M)
Supports **idtune** by checking the desired parameter value against the minimum and maximum specified in the */etc/conf/cf.d/mtune* file.

The */etc/conf/cf.d* Directory

The */etc/conf/cf.d* directory contains the master and system configuration files used by the **idbuild** command to create a new kernel. It also contains the most recently created kernel and inittab file.

The files in */etc/conf/cf.d* are:

init.base
Used by **idmkinit** to create a new */etc/inittab* file each time the kernel is rebuilt.

kernmap
Contains the location in memory of the kernel text, data, etc.

mdevice
Contains the master copy of device specifications for all devices possible on the system. Installation of application packages can add device specifications to this file.

mfsys
Contains the master copy of file system type data. It is created by the **idbuild** command using information in the files in the */etc/conf/mfsys.d* directory.

mtune
Contains the master copy of the default, minimum and maximum values for the tunable parameters. Normally, you should make changes to default parameter values in the *stune* file rather than this one.

sassign
Contains the system–specific device assignments for **swap, dump, root, pipe**, etc.

sdevice
Contains the system–specific device specifications. The *sdevice* file is created by **idbuild** using information from the files in the */etc/conf/sdevice.d* directory.

sfsys
Contains system–specific file system type data. It is created by the **idbuild** command using information in the files in the */etc/conf/sfsys.d* directory.

stune
Contains system–specific specifications for tunable parameters. If you want to change the default value for a parameter, specify the desired value in this file. Values in the *stune* file override values in the *mtune* file. You can use **idtune** or a system editor to edit the *stune* file.

unix
Contains the most recently configured kernel. It is overwritten (not saved) when a new kernel is made.

The */etc/conf/sdevice.d* Directory

The */etc/conf/sdevice.d* directory contains system–specific device driver information, one file for each driver. The first two items in each driver file identify the driver and indicate whether it is to be configured into the system (Y or N).

When you invoke the **idbuild** command to build a new kernel, **idbuild** concatenates the information from the *sdevice.d* files into the */etc/conf/cf.d/sdevice* file. Master device driver information is located in the */etc/conf/cf.d/mdevice* file.

The */etc/conf/sfsys.d* Directory

The */etc/conf/sfsys.d* directory contains system file system information, one file for each file system type. Each file contains one line that identifies the file system type and indicates whether it is to be configured into the system (Y or N). When you invoke the **idbuild** command to build a new kernel, **idbuild** concatenates information from the files in *sfsys.d* into the */etc/conf/cf.d/sfsys* file. Master file system information is in the */etc/conf/cf.d/mfsys* file.

The */etc/conf/pack.d* Directory

The *pack.d* directory contains a subdirectory for each module that may be available on the system. For example, */etc/conf/pack.d/sem* is the module subdirectory for the **sem** module (semaphores). Within each module subdirectory are the necessary *object*, *space.c*, and *stubs.c* files for that module.

A new module directory must be created if you add a new driver or module to the system.

Refer to the "Modules" section for more information.

The */etc/.new_unix* File

The */etc/.new_unix* file is created by **idbuild** when a new kernel is made to flag the existence of the new kernel. If */etc/.new_unix* exists, the new kernel in */etc/conf/cf.d/unix* is copied to */stand/unix* at shutdown. When the system enters multiuser mode after rebooting, the */etc/.new_unix* file is removed by **idmkenv** (the **rc2** script calls **idmkenv**).

Creating New Kernels

The kernel is the program that defines the operating system on the machine. New kernels are created from the configuration files and commands in the */etc/conf* directory.

In NCR UNIX SVR4, the default kernel is */stand/unix* and the */unix* file is only a symbolic link to */stand/unix*. You can specify a different default kernel in the */stand/boot* file; however, this kernel should be located in */stand*.

Valid Kernel Names

The following names are valid for bootable kernels:

- Within */stand*, the kernel name can be any valid file name. The default is */stand/unix*, but you could use */stand/myfile* instead.
- Outside of */stand*, the kernel name may be */unix* or */etc/conf/cf.d/unix* but it can NOT have an extension. For example, */unix* but NOT */unix.old*.

Before You Start

You should take the following actions before creating a new kernel:

- Make a copy of any configuration files you plan to change.

  If the new kernel does not function properly, you can change back to the original values.

- Put the system in single user mode.

  You can modify configuration files and create a new kernel while others are using the system; however, you should be sure everyone is logged off the system before you reboot to install the new kernel.

Creating a New Kernel

Step 1: Save the existing kernel to a location outside of */stand.* For example:

```
# cp /stand/unix /etc/conf/cf.d/unix.old
```

Step 2: If necessary, edit the */etc/conf/cf.d/init.base* file to include any changes you have made to */etc/inittab* that you want to retain in the next *inittab* file.

Step 3: Edit the */etc/conf/cf.d/stune* and */etc/conf/cf.d/mtune* files to reflect the desired parameter values.

In addition, you may want to edit files in the */etc/conf/sdevice.d*, and */etc/conf/sfsys.d* directories to include or exclude drivers or file systems.

Step 4: Invoke the **idbuild**(1M) command to build a new kernel. For example:

```
# /etc/conf/bin/idbuild
```

Step 5: Shutdown and reboot the system to cause the newly created kernel to take effect. For example:

```
# cd /
# shutdown -i6 -g0 -y
```

What Happens After You Create a New Kernel

If an */etc/.new_unix* file exists, the final step of the procedure to create a new kernel causes the system to prepare to use the new kernel.

At shutdown:

- The **rc0** script saves the old kernel as */stand/unix.old* and copies */etc/conf/cf.d/unix* to */stand/unix* so the new kernel is used at the next reboot.
- It then creates a symbolic link from */unix* to */stand/unix*.

At startup:

- The system boots using the new */stand/unix* kernel
- The **rc2** script calls the **idmkenv** command which creates a new system environment for the new kernel:
- Updates the */etc/idrc.d* and */etc/idsd.d* directories.
- Calls **idmkinit** to update the */etc/inittab* file.
- Calls **idmknod** to update the device nodes in */dev*.
- Removes the */etc/.new_unix* file.

If You Have Problems

If you can not reboot using the new kernel, try the following in the order given:

- Reboot, and when you see the "Booting UNIX System" message, press the spacebar to interrupt the boot sequence and specify an alternate kernel. In addition to the latest *unix*, the */stand* directory normally contains the following three kernels. Try them in the order listed.
  - *unix.old* – copy of the last kernel before the rebuild
  - *umix.diag* – copy of the latest kernel
  - *unix.ncrm* – golden kernel which contains only basic functionality but should always be good
- If you can not reboot using an alternate kernel, recover a bootable kernel from the */install* file system on the BOOT flex and then copy your old kernel back to */stand/unix* after booting. Refer to the "Recovery Techniques" chapter for this procedure.

Loading/Booting the Kernel

During the boot procedure, the **boot** program reads the kernel from disk, loads it into memory, and executes it. In NCR UNIX SVR4, the default kernel is */stand/unix* and the */unix* file is only a symbolic link to */stand/unix*. You can load/boot the kernel (operating system) using either the auto boot or manual boot sequence.

NOTE: The file */stand/boot* defines parameters used in booting: for example, whether to perform an auto or manual boot and what kernel to use.

Auto Boot Sequence

The auto boot sequence automatically loads the default kernel (normally */stand/unix*). No user input is required.

If */stand/unix* does not exist, the loader attempts to load */unix* from the *root (/)* device. If */unix* is only a symbolic link to */stand/unix* (default), the system can not be loaded and the following message is displayed:

```
BFS not fully populated, REVERTING back to S5 1K
```

If the auto boot process fails, follow the suggestions in the "If You Have Problems" section of "Creating New Kernels."

Manual Boot Sequence

You can invoke the manual boot sequence by specifying it in the */stand/boot* file or by pressing the spacebar key to interrupt the automatic load sequence when the message "Booting the UNIX System" is displayed.

During the manual boot sequence, you receive the following prompt:

```
Enter the name of a kernel to boot:
```

You can enter any valid kernel name. Refer to the "Valid Kernel Names" section in "Creating New Kernels."

If you enter the name of a file that does not exist or is not a valid file name, the following messages are displayed and you may then enter a valid name:

```
boot:  Cannot load /unix.new:  file not opened.

Enter the name of a kernel to boot:
```

Modules

The */etc/conf/pack.d* directory contains one subdirectory for each module that can be included in the kernel. Within each module subdirectory are the necessary *object*, *space.c*, and *stubs.c* files for that module. Figure 3–3 describes the contents of these files.

Filename	Description
object.o	Object files associated with the module.
space.c	Files containing declarations for the **data** and **bss** (uninitialized data) space that must be allocated for the module.
stubs.c	If the module is optional and there are global references in object files or the *space.c* file, then this object file must contain declarations to resolve those global references.

Figure 3–3: Files in the Modules Directories

Including Modules in the Kernel

The */etc/conf/sdevice.d* and */etc/conf/sfsys.d* directories contain one file for each module that can be included in the kernel. In each module's file, the first item identifies the module and the second item indicates (by **Y** or **N**) whether it is to be included in the kernel. See Figure 3–1 for examples.

You can control whether a module is included in the kernel configuration by specifying **Y** or **N** in the module file.

Description of Available Modules

Figure 3–4, Figure 3–5, Figure 3–6, Figure 3–7, Figure 3–8, Figure 3–9, and Figure 3–10 on the following pages describes the modules available, indicates whether each module is required, and indicates any restrictions or dependencies between modules.

Module	Description	Notes
Architecture	NST Architecture module	
NST_All_Levels	Support for all NST Levels	
NST_Level3	Support module for NST Level 3 platforms	
NST_Level4	Support module for NST Level 4 platforms	
NST_Level5	Support module for NST Level 5 platforms	
SCSI	SCSI driver	Required
XENIX	XENIX module. Contains compatibility files and the Xenix File System Code.	Optional
acl	Driver for the NCR 8–Port Serial Controller	Required for NCR 8–Port Serial Controller
ansi	STREAMS module that translates ANSI X3.64 escape sequences to Terminal Control Language (i.e., makes the console an ANSI type device)	
arp	STREAMS module that provides address resolution protocol	Network package addition
asy	Serial port STREAMS driver	Required
asydcap		
async	Asynchronous file I/O. Allows applications to issue I/O requests and continue processing while the I/O completes.	Optional
bfs	Boot file system	Required

Figure 3–4: Modules Available for the Kernel (1 of 7)

Module	Description	Notes
char	STREAMS module that provides scan–code translation (IBM to ASCII) and screen mapping	Required for console
cat	CAT bus driver for NST hardware	Required
clist	Line discipline tty for character device drivers	
clone	STREAMS "clone" driver	
cmux	STREAMS multiplexor driver	Required for console
coff	Common Object File Format (old C format)	
connld	STREAMS module that establishes unique connection for STREAMS–based pipes	
cpyrt		
crypt	Security module (provides encryption and decryption for encoding and decoding sensitive data files.	Available only in U.S.
cram	CMOS RAM driver	Required
ctkdb	Kernel debugger	Optional
dbt		
dcap		
des	Encryption driver	Optional
datacpt	Data capture streams module	
dma	Direct memory access driver	Required
dosx	DOS execution format	Optional
elf	Executable and Linking File Format (new C format)	Required
events	Events file system	Optional
fd	Flex disk driver	Required
fdfs	File description file system	Optional

Figure 3–5: Modules Available for the Kernel (2 of 7)

Module	Description	Notes
fifofs	Pipes file system	Required
gendisp	Generic dispatcher	Required
gentty	Indirect driver for controlling tty (C driver)	Required
gvid		
hd	Hard disk driver	Optional
hrt	High resolution timers	Required
i286x	i286 execution format	Optional
fp	Floating point unit	Required
ild	STREAMS Integrated LAN Driver. Contains driver for Ethernet and Token Ring Controllers.	Network package addition; required for Ethernet or Token Ring.
intp	#! execution format	Required
ip	STREAMS IP multiplexor module	Network package addition
ipc	Inter–process communication core	Optional; must be included before **msg**, **sem**, or **shm**
jqf	Job Queueing Facility	
kd	Keyboard/Display STREAMS driver	Required
kdb	Kernel debugger (in house)	Optional
kdvm	VGA driver	Required
kernel	NCR UNIX kernel core	Required; automatically included
kma	Kernel memory allocator	Required
ktli	Kernel TLI–like functions	Optional

Figure 3–6: Modules Available for the Kernel (3 of 7)

Module	Description	Notes
ldterm	STREAMS module that provides line discipline for terminals	Required
kmacct	Kernel memory accounting	Optional
krpc	Kernel remote procedure calling	Optional
lockinfo	Lock debugging module	Always present
log	STREAMS log driver	Required
loop	STREAMS loop driver	Network package addition
lp	Parallel printer driver	Required
lser	Logitech 3–button serial mouse driver	Package addition
lsyn	Logitech 3–button synchronous mouse driver	Package addition
m320	Microsoft 2–button synchronous mouse driver	Package addition
mem	Pseudo device driver for **memory, null, zero,** etc.	Required
mp	Module for multi–processing lock support	Required
mse	Mouse driver	Package addition
merge	MERGE 386	
msg	IPC message queues	Optional; must also include **ipc**
namefs	Name file system	Optional
nfa	OPEN NET	
nfs	Network File System. Provides transparent, remote access to file system resources in a TCP/IP network.	Optional; NFS only.

Figure 3–7: Modules Available for the Kernel (4 of 7)

Module	Description	Notes
nxt	STREAMS driver for AT&T windowing terminals	Never required
osm	OS message driver	
nmi		
nsxt	STREAMS multiplexing driver for shell layers	Optional
otos	Kernel module specific to the system 3600	Optional
pckt	STREAMS module to packetize message on the read queue	
pic		
pipemod	STREAMS–based pipe flusher	
pramd	Physical RAM Disk Driver	Optional
prf	Pseudo driver for profiling	Optional
proc	*/proc* file system	
ptem	STREAMS pseudo terminal driver emulator	
ptm	STREAMS pseudo terminal master module	
pts	STREAMS pseudo tty slave module	
raio		
ramd	RAM disk driver used by install	Optional
raw	Raw TLI multiplexor	Network package addition
rmc		
rt	Real time process management class	
rtc	Real time clock	
s5	System V file system	

Figure 3–8: Modules Available for the Kernel (5 of 7)

Module	Description	Notes
sad	STREAMS administrator driver	Required for auto-push
sat	Security audit trail	
sem	Semaphores	Optional; must also include **ipc**
shm	Shared memory	Optional; must also include **ipc**
smse	Serial mouse driver	Package addition
smterm		
sockmod	Socket Interface Library cooperating module (STREAMS)	
specfs	**specfs** file system for special files	Required
stty	Synchronous tty	
sttydcap		
strm	Streams migrator module	Required
sxt	Character driver for shell layers (C driver)	Optional
sysmsg	System message driver	Required
tcp	TCP multiplexor (STREAMS)	Network package addition
ticots	TPI loopback transport provider, virtual circuit mode, connection–oriented type (STREAMS)	
ticotsor	TPI loopback transport provider, virtual circuit mode, connection–orderly release (STREAMS)	
timod	Transport interface library cooperating module (STREAMS)	

Figure 3–9: Modules Available for the Kernel (6 of 7)

Module	Description	Notes
tirdwr	Transport interface library read/write module (STREAMS)	
tr	Token ring driver contained in the ild package	Optional
tracev	Event tracing facility (debug module)	Always present
ts	Timesharing process management class	
ttcompat	STREAMS module for old termio interfaces for VT, 4BSD, and XENIX	Required
tubefs	Performance module	
udp	STREAMS UDP multiplexor	Network package addition
ufs	ufs file system	
vc		
vx		
weitek	Weitek floating point chip handler	
ws	Work station software for kd, cmux, char	
wtty		
xintr	Cross–Processor Interrupt support	Always required
xnamfs	XENIX name file system	
xout		
xque	Event queues	
xt	Driver for AT&T windowing terminals (C driver)	Never required

Figure 3–10: Modules Available for the Kernel (7 of 7)

Configuration Parameters

This section explains how to change the values of parameters and provides charts of the parameters values. It also describes the hard and soft limit parameters that provide dynamic allocation of resources for processes.

The */etc/conf/cf.d/stune* and/or */etc/conf/cf.d/mtune* files specify the values for tunable configuration parameters.

NOTE: Some parameters in the *mtune* file are not used by the kernel but are retained for compatibility with previous releases. These parameters are marked with an asterisk (*) in the charts later in this chapter.

Changing Parameter Values

The configuration parameters for your system are defined in the */etc/conf/cf.d/mtune* file. For each parameter, the *mtune* file specifies the default, minimum, and maximum value.

You can change the value of any parameter by specifying a new default value in the *stune* or *mtune* file. Whenever possible, specify changes in the *stune* file; however, to assign a value that is outside the minimum–maximum range, you must first change the minimum or maximum value in *mtune*.

You can make changes to *stune* using the **idtune** command or a system editor. The **idtune** command is described in the section "The /etc/conf/bin Directory" earlier in this chapter. You can make changes to *mtune* using a system editor.

The **idtune**(1M) command checks the new parameter value and refuses to make the change unless it is within the range specified in the *mtune* file. If you use a system editor to change a parameter value, no validity check is made until the new kernel is built.

The following example uses **idtune** to change the value of the MAXUP parameter in the *stune* file.

```
# /etc/conf/bin/idtune MAXUP 40
```

Using Hard and Soft Limit Parameters

Certain per process resources are now allocated according to hard and soft limits specified by parameter pairs.

- The soft limit defines the normal amount of the resource a process can use.
- The hard limit defines the maximum value the soft limit can attain.

If a process uses its soft limit and needs more of a resource, the process can increase its soft limit up to the hard limit. A soft limit can never exceed the corresponding hard limit.

The per process limits are given to process 0, after which child processes inherit the parent's hard and soft limits. However, when a process **exec**s a file whose setuid or setgid bit is set, the resource limits of that process are reinitialized to the default system limits.

With Recently Coded Programs

To take advantage of the hard and soft limits, you must code programs to check for situations where the resource limit is reached and use the **setrlimit** (see **getrlimit**(2) system call to increase the limit). Any increase remains in effect only for the duration of that process.

The **setrlimit** system call can be used to increase both hard and soft limits. Although any process can use **setrlimit** to increase its soft limit, only a process with an effective user ID equal to 0 (**root**) can raise its hard limit. A value equal to RLIMIT_INFINITY (0x1FFFFFF) indicates a resource without limitation.

With Previously Coded Programs

Programs that do not use **setrlimit** to dynamically increase resource limits will fail when they reach the soft limit. In this case, you must change the */etc/conf/cf.d/stune* and/or */etc/conf/cf.d/mtune* file to reflect the necessary hard and soft limits and relink the kernel before the program can run successfully. Raising limits is not always desirable, because it makes the new limits available for all processes at all times.

Defaults and Ranges of Configuration Parameters

The figures in this section categorize the configuration parameters according to function, which is the way they appear in the *mtune* file, and provide the default value and range for each parameter. The chapter on configuration parameters contains a detailed description of each parameter.

Parameters in Figure 3–36 and Figure 3–37 are defined in the */etc/conf/pack.d/kernel/space.c* file, not in the *mtune* file.

The following figures are provided:

Figure 3–11 General kernel parameters (1 of 2)

Figure 3–12 General kernel parameters (2 of 2)

Figure 3–13 File system parameters

Figure 3–14 Paging parameters

Figure 3–15 **pageout** daemon parameters

Figure 3–16 STREAMS parameters

Figure 3–17 log (strlog) parameters

Figure 3–18 IPC message parameters

Figure 3–19 IPC semaphore parameters

Figure 3–20 IPC shared memory parameters

Figure 3–21 RFS parameters

Figure 3–22 XENIX parameters

Figure 3–23 Miscellaneous parameters

Figure 3–24 Driver parameters

Figure 3–25 ASYNCIO parameters

Figure 3–26 EVENTS parameters

Figure 3–27 Timer and scheduler parameters

Figure 3–28 Affinity scheduling parameters

Figure 3–29 Resource limit parameters

Figure 3–30 Overflow buffer parameters

Figure 3–31 Shared Memory Nailing GIDs

Figure 3–32 bdevcnt, cdevcnt parameters

Figure 3–33 Security audit trail parameter

Figure 3–34 Integrated LAN Driver parameter

Figure 3–35 Level 5 tunable parameters

Figure 3–36 System parameters

Figure 3–37 Hardware parameters

NOTE: Parameters that are NOT used by the operating system are marked with an asterisk (*) in the figures that follow.

General Kernel Parameters]		
Parameter	**Default**	**Range**
ARG_MAX	5120	1024 – 51200
BDFLUSHR*	1	1 – 1
BUFHWM	0	0 – 65536
FDFLUSHR	1	1 – 1
FLCKREC	300	100 – 2000
MAXMINOR	0x3fffff	255 – 0x3fffff
MAXPMEM	0	0 – physical memory
MAXSLICE	100	25 – 100
MAXUP	30	15 – 400
NAUTOUP	60	0 – 120
NBUF	100	100 – 3000
NCALL	60	30 – 700
NCLIST	120	1 – 400
NCPU	8	1 – 32
NFILE*	150	100 – 5000
NGROUPS_MAX	16	1 – 16
NHBUF	64	32 – 1024
NMOUNT	25	25 – 25
NOFILES*	60	20 – 1000
NPBUF*	20	20 – 4000
NPROC	250	50 – 5104
NREGION*	210	210 – 350

Figure 3–11: General Kernel Parameters (1 of 2)

General Kernel Parameters (continued)		
Parameter	**Default**	**Range**
PIOMAP	50	50 – 50
PIOMAXSZ	64	4 – 64
PUTBUFSZ	2000	2000 – 10000
RSTCHOWN	0	0 – 1
SHLBMAX	3	2 – 6
SPTMAP*	400	50 – 600
TRACESZ*	8192	4096 – 8192
ULIMIT	65536	2048 – 65536
* Not used		

Figure 3–12: General Kernel Parameters (2 of 2)

File System Parameters		
Parameter	**Default**	**Range**
NINODE	500	100 – 5000
NS5INODE*	500	100– 1300
UFSNINODE	500	100– 5000
NDQUOT	200	100 – 400
NRNODE	300	100– 1300
S52KNBUF*	100	100– 400
S52KNHBUF*	32	32 – 256
CDFS File System Parameters		
Parameter	**Default**	**Range**
NCDINODE	250	100 – 2500
NCDFILSYS	500	200 – 5000
NCDEXTENT	5	1 – 20
* Not used		

Figure 3–13: File System Parameters

Paging Parameters		
Parameter	**Default**	**Range**
AGEINTERVAL*	9	2 – 100
GPGSHI*	40	1 – 40
GPGSLO	25	0 –25
GPGSMSK*	0x00000420	0x00000420 – 0x00000420
MAXFC*	1	1 – 1
MAXSC*	1	1 – 1
MAXUMEM*	2560	2560 – 32768
MINAKMEM	16	4 – 64
MINARMEM	100	100 – 160
MINASMEM	25	25 – 40
MINHIDUSTK*	4	4 – 32
MINPAGEFREE	25	25–50
MINUSTKGAP*	2	2 – 32
PAGES_UNLOCK	200	200 – 200
VHNDFRAC*	16	8 – 32
* Not used		

Figure 3–14: Paging Parameters

Pageout Daemon Parameters		
Parameter	**Default**	**Range**
DESFREE	0	0 – 256
LOTSFREE	0	0 – 512
MINFREE	0	0 – 128

Figure 3–15: **pageout** Daemon Parameters

STREAMS Parameters		
Parameter	**Default**	**Range**
MAXSEPGCNT*	1	0 – 32
NBLK parameters*	N/A	N/A
NMUXLINK*	87	1 – 87
NQUEUE*	96	1 – 4096
NSTREAM*	32	1 – 512
NSTREVENT*	256	256 – 512
NSTRPUSH	9	9 – 10
STRCTLSZ	1024	1024 – 1024
STRLOFRAC*	80	0 – 80
STRMEDFRAC*	90	80 – 100
STRMSGSZ	0	0 – 32767
STRTHRESH	0x200000	0 – 0x1000000
* Not used		

Figure 3–16: STREAMS Parameters

log (strlog) Parameters		
Parameter	**Default**	**Range**
MAXWCONSMSG	80	0 – 100
MAXWERRMSG	80	0 – 100
MAXWTRCMSG	80	0 – 100
NAUTOPUSH	32	32 – 32
NLOG	10	3 – 16
NSTRPHASH	64	16 – 512
NUMSAD	8	1 – 16
NUMSP	32	5 – 50
NUMTIM	128	1 – 2048
NUMTRW	16	1 – 2048
SOCK_HIWATER	2048	512 – 65535
SOCK_LOWATER	1024	128 – 65535
TIM_HIWATER	65536	512 – 1048576
TIM_LOWATER	16384	512 – 1048576
TRW_HIWATER	65536	512 – 1048576
TRW_LOWATER	16384	512 – 1048576

Figure 3–17: log (strlog) Parameters

IPC Message Parameters		
Parameter	**Default**	**Range**
MSGMAP	100	10 – 400
MSGMAX	2048	512 – 8192
MSGMNB	4096	4096 – 4096
MSGMNI	50	50 – 1000
MSGSEG	1024	1024 – 1024
MSGSSZ	8	8 – 8
MSGTQL	40	40 – 100

Figure 3–18: IPC Message Parameters

IPC Semaphore Parameters		
Parameter	**Default**	**Range**
SEMAEM	16384	16384 – 16384
SEMMAP	25	10 – 1000
SEMMNI	25	10 – 1000
SEMMNS	60	60 – 1000
SEMMNU	30	30 – 1000
SEMMSL	25	25 – 1000
SEMOPM	10	10 – 10
SEMUME	10	10 – 10
SEMVMX	32767	32767 – 32767

Figure 3–19: IPC Semaphore Parameters

IPC Shared Memory Parameters		
Parameter	**Default**	**Range**
SHMMAX	524288	131072 – 209715200
SHMMIN	1	1 – 1
SHMMNI	100	100 – 500
SHMSEG	6	6 – 15

Figure 3–20: IPC Shared Memory Parameters

RFS Parameters		
Parameter	**Default**	**Range**
MAXGDP	24	10 – 32
MAXSERVE	6	3 – 6
MINSERVE	3	3 – 3
NADVERTISE	25	1 – 25
NLOCAL*	0	0 – 10
NRCVD	150	40 – 500
NRDUSER	250	1 – 700
NREMOTE*	0	0 – 10
NSNDD	150	100 – 350
NSRMOUNT	20	1 – 50
RCACHETIME	10	–1 – 10
RFHEAP*	3072	1024 – 3072
RF_MAXKMEM	0	0 – 50
RFS_VHIGH*	2	1 – 2
RFS_VLOW*	1	1 – 1
* Not used		

Figure 3–21: RFS Parameters

XENIX Parameters		
Parameter	**Default**	**Range**
DSTFLAG	1	0 – 1
NEMAP	10	10 – 10
NSCRN	0	0 – 10
TIMEZONE	480	0 – 1440
XSDSEGS	25	0 – 100
XSDSLOTS	3	0 – 5
XSEMMAX	60	0 – 60

Figure 3–22: XENIX Parameters

Miscellaneous Parameters		
Parameter	**Default**	**Range**
DMAABLEBUF	70	10 – 100
DMAEXCL	1	0 – 1
DO386B1*	2	0–2
DO387CR3*	2	0–2
KDBSYMSIZE	100000	10000 – 400000
MAXDMAPAGE	0	0 – 65536
PIOSEGSZ	1024	1024 – 1024
SANITYCLK*	0	0 – 1
SEGMAPSZ	0	0 – 4096
* Not used		

Figure 3–23: Miscellaneous Parameters

Device Driver Parameters		
Parameter	**Default**	**Range**
CMF	1	0 – 1
COM2CONS*	0	0 – 1
MNR_ON*	0	0 – 1
NCPYRIGHT	10	10 – 10
NKDVTTY*	8	8 – 15
NUMSXT	6	1 – 6
NUMXT	3	1 – 3
PRFMAX	4096	2048 – 10240
RCMF	0	0 – 1
RIDEOUT*	180	30 – 36000
SANECNT*	120	120 – 600
USANEON*	1	0 – 1
* Not used		

Figure 3–24: Driver Parameters

ASYNCIO Parameters		
Parameter	**Default**	**Range**
AIOTIMEOUT	60	60 – 60
MAXAIOS	5	1 – 64
MINAIOS*	1	1 – 1
NAIOPROC*	10	10 – 10
NAIOSYS	300	0 – 512
* Not used		

Figure 3–25: ASYNCIO Parameters

EVENTS Parameters (not used)		
Parameter	**Default**	**Range**
EVDATA*	20480	20480 – 20480
EVFNHTS*	256	256 – 256
EVMAXEV*	20	20 – 20
EVMAXDPE*	1024	1024 – 1024
EVMAXMEM*	10240	10240 – 10240
EVMAXTRAPS*	25	25 – 25
EVMAXETERMS*	20	20 – 20
EVTIDHTS*	128	128 – 128
MEVDIRENTS*	50	50 – 50
MEVEXITS*	50	50 – 50
MEVEXPRS*	50	50 – 50
MEVEXREFS*	50	50 – 50
MEVKEVS*	50	50 – 50
MEVQUEUES*	50	50 – 50
MEVRETRYS*	50	50 – 50
MEVSEXPRS*	50	50 – 50
MEVSIGS*	50	50 – 50
MEVSTERMS*	100	100 – 100
MEVSTRDS*	50	50 – 50
MEVTERMS*	250	250 – 250
MEVTIDS*	100	100 – 100
* Not used		

Figure 3–26: EVENTS Parameters

Timer and Scheduler Parameters		
Parameter	**Default**	**Range**
HRTIME	50	50 – 50
HRVTIME	50	50 – 50
MAXCLSYSPRI	99	99 – 99
RTMAXPRI*	59	59 – 59
RTNPROCS	60	60 – 60
TSMAXUPRI	20	10 – 30
TSNPROCS	60	60 – 60
* Not used		

Figure 3–27: Timer and Scheduler Parameters

Affinity Scheduling Parameters		
Parameter	**Default**	**Range**
AFFIN_ON	0	0 – 1
AFFINDECAY	16	0 – 512

Figure 3–28: Affinity Scheduling Parameters

Resource Limit Parameters		
Parameter	**Default**	**Range**
SCORLIM	0x2000000	0x100000 – 0x7FFFFFFF
HCORLIM	0x2000000	0x100000 – 0x7FFFFFFF
SCPULIM	0x7FFFFFFF	60 – 0x7FFFFFFF
HCPULIM	0x7FFFFFFF	60 – 0x7FFFFFFF
SDATLIM	0x1000000	0x1000000 – 0x7FFFFFFF
HDATLIM	0x1000000	0x1000000 – 0x7FFFFFFF
SFNOLIM	0x40	0x20 – 0x400
HFNOLIM	0x400	0x20 – 0x400
SFSZLIM	0x7FFFFFFF	0x100000 — 0x7FFFFFFF
HFSZLIM	0x7FFFFFFF	0x100000 – 0x7FFFFFFF
SSTKLIM	0x1000000	0x2000 – 0x7FFFFFFF
HSTKLIM	0x1000000	0x2000 – 0x7FFFFFFF
SVMMLIM	0x1000000	0x1000000 – 0x7FFFFFFF
HVMMLIM	0x1000000	0x1000000 – 0x7FFFFFFF

Figure 3–29: Resource Limit Parameters

Overflow Buffer Parameters for pageio_setup		
Parameter	**Default**	**Range**
PGOVERFLOW	16	8 – 128
NOTPGOVER-FLOW	16	8 – 128

Figure 3–30: Overflow Buffers Parameters

Shared Memory Nailing GID's		
Parameter	**Default**	**Range**
SHM_NAILED_GID1	0	0 – 2147483647
SHM_NAILED_GID2	0	0 – 2147483647
SHM_NAILED_GID3	0	0 – 2147483647
SHM_NAILED_GID4	0	0 – 2147483647
SHM_NAILED_GID5	0	0 – 2147483647
SHM_NAILED_GID6	0	0 – 2147483647
SHM_NAILED_GID7	0	0 – 2147483647
SHM_NAILED_GID8	0	0 – 2147483647
SHM_NAILED_GID9	0	0 – 2147483647

Figure 3–31: Shared Memory Nailing GIDs

bdevcnt, cdevcnt Parameters		
Parameter	**Default**	**Range**
BMAX*	25	20 – 256
CMAX*	40	30 – 256
* Not used		

Figure 3–32: bdevcnt, cdevcnt Parameters

Security Audit Trail Parameter		
Parameter	**Default**	**Range**
SATENABLED	0	0 – 1

Figure 3–33: Security Audit Trail Parameter

Integrated LAN Driver Parameter		
Parameter	**Default**	**Range**
ILDMAXOPENS	64	16 – 512

Figure 3–34: Integrated LAN Driver Parameter

Level 4 & 5 Tunable Parameters		
Parameter	**Default**	**Range**
L5CPUenables	0xef	0 – 0xff
L5DMAenables	0xc0	0 – 0xff
L5McAddrenables	0xfd	0 – 0xff
L5McDataenables	0x87	0 – 0xff
L5NMIenables	0xff	0 – 0xff
L5SysInt_Type	5	4 – 5
L5SysIntenables	0	0 – 0xff
L5WatchDogPeriod	15	0 – 0xffffffff
L5WatchDogPFR-Period	1800	0 – 0xffffffff

Figure 3–35: Level 5 Tunable Parameters

Performance Group Tunables		
Parameter	**Default**	**Range**
JQF_CLIENTS	2500	100 – 3000
JQF_SIZE	40	40 –300
JQF_SESS	10	0 – 50
PW_MAX_ORA_PID	100	0 – 300
PW_MAX_EVENT	100	0 – 300
LISTIO_MAX_CNT	256	1 – 2048
TUBESIZE	1024	1024 – 4096
System Parameters		
Parameter	**Default**	**Range**
NODE	unix	N/A
REL	4.0	N/A
SRPC_DOMAIN	NULL	N/A
SYS	UNIX_ System_V	N/A
VER	3.0	N/A

Figure 3–36: System Parameters

Hardware Parameters		
Parameter	**Default**	**Range**
ARCHITECTURE	386/486/MC	N/A
HW_PROVIDER	NCR	N/A
HW_SERIAL	0	N/A

Figure 3–37: Hardware Parameters

Chapter 4 Configuration Parameters

This chapter contains a detailed page for each configuration parameter that is supported and used by the system. Parameters that appear in the */etc/conf/cf.d/mtune* file but are not supported or used appear only in the charts in the "Configuring Kernels" chapter.

The parameters in this chapter are listed in alphabetical order. The page for each parameter includes the following information:

- Description

  Describes the parameter, including the default value and range (minimum and maximum values).

  The default value is the value the system uses. You can not specify a default value that is outside of the range. You must first change the minimum or maximum value for the parameter in the *mtune* file so that the range includes the value you plan to use.

- When To Change the Default

  Describes when you should change the value of the parameter from the default.

AFFIN_ON

Description

The AFFIN_ON parameter specifies whether affinity scheduling is enabled.

- **Range** – 0 – 1
- **Default** – 0

When to Change the Default

Change the value of this parameter to 1 if you want to enable affinity scheduling.

AFFINDECAY

Description

The AFFINDECAY parameter specifies the number of process switches that may occur before a process loses its affinity with a specific processor.

- **Range** – 0 – 512
- **Default** – 16

When to Change the Default

Normally, you do not need to change the value of this parameter.

AIOTIMEOUT

Description

The AIOTIMEOUT parameter specifies the timeout value for asynchronous I/O.

- **Range** – 60 – 60
- **Default** – 60

When to Change the Default

Normally, you do not need to change the value of this parameter.

ARCHITECTURE

Description

The ARCHITECTURE parameter specifies machine architecture information. The value is defined in the kernel */etc/conf/pack.d/kernel/space.c* file, not the *mtune* file.

The value of ARCHITECTURE is 386/486/MC for both 386 and 486 machines.

- **Range** – N/A
- **Default** – 386/486/MC

When to Change the Default

You do not need to change the value of this parameter.

ARG_MAX

Description

The ARG_MAX parameter specifies the maximum number of characters (including NULL characters) allowed in the argument and environment strings passed to an **exec** system call.

- **Range** – 1024 – 51200
- **Default** – 5120

When to Change the Default

Increase the value of this parameter to allow larger argument lists if you receive the error message:

```
Argument list too long
```

To be sure there is room for both the pointer arrays and the ordinary stack frames, ARG_MAX should not be more than 1/8th of SSTKLIM.

Do not decrease the value of this parameter.

BUFHWM

Description

The BUFHWM parameter specifies the maximum amount of memory (in kilobytes) that can be used by block I/O buffers. If the value of BUFHWM is 0 (the default), the kernel sets the value of BUFHWM to 25% of available memory.

In UNIX SVR4, buffers only contain inode, superblock(s) and file header information; they do not contain file data. The actual file data is stored in pages of virtual memory (the amount of virtual memory available for this purpose is determined by the SEGMAPSZ parameter).

There is a buffer header for each buffer. The initial number of block I/O buffer headers allocated is specified by the NBUF parameter. If a buffer header is needed but no free ones are available, the system dynamically allocates more buffer headers in chunks of NBUF at a time. When buffers have used the amount of memory specified by BUFHWM, no more buffer headers can be allocated.

- **Range** – 0 –65536
- **Default** – 0

When to Change the Default

Normally, you should not change the value of this parameter.

If the output of **sar –b** shows the buffer hit ratio to be low, you may want to try increasing BUFHWM; however, allocating more than 25% of memory for buffers is not usually advisable.

CMF

Description

The CMF parameter is a flag specifying whether to send messages to the system console.

- **Range** – 0 – 1
- **Default** – 1

When to Change the Default

Set the value of this parameter to 0 only if you want to prevent messages from being sent to the system console.

DESFREE

Description

The DESFREE parameter specifies the amount of free memory that is considered "desirable". The value is used to determine when pages in memory should be freed. If the amount of free memory falls below DESFREE pages, the **pageout** daemon is invoked to free pages.

If the value of DESFREE is 0 (the default), the kernel sets it to 1/16th of the pages of memory (not to exceed 128 pages).

The DESFREE parameter is closely related to the LOTSFREE and MINFREE parameters. Together, these three parameters control paging.

- **Range** – 0 –256
- **Default** – 0

When to Change the Default

Normally, you do not need to change this parameter.

DMAABLEBUF

Description

The DMAABLEBUF parameter specifies the number of pages to reserve exclusively for Direct Memory Access (DMA) I/O use. DMA is used by drivers to make memory transfers between processor boards without involving the processor.

- **Range** – 10 – 100
- **Default** – 70

When to Change the Default

Increase this parameter if performance of drivers using DMA I/O decreases due to lack of pages available for DMA I/O.

DMAEXCL

Description

The DMAEXCL parameter is a switch that indicates whether exclusive ownership of the Direct Memory Access (DMA) chip is needed. DMA is used by drivers to make memory transfers between processor boards without involving the processor.

A value of 1 indicates exclusive ownership of DMA is needed; a value of 0 indicates it is not.

- **Range** – 0 – 1
- **Default** – 1

When to Change the Default

Do not change the value of this parameter.

DSTFLAG

Description

The DSTFLAG parameter is a flag to indicate daylight savings time. A value of 1 indicates the system should use daylight savings time; a value of 0 indicates it should not.

- **Range** – 0 – 1
- **Default** – 1

When to Change the Default

If you move the system to a different location, you may need to change the value of DSTFLAG to reflect the appropriate use of daylight savings time for the new location.

FDFLUSHR

Description

The FDFLUSHR parameter determines the rate, in seconds, at which the **fsflush** daemon is called. FDFLUSHR operates together with the NAUTOUP parameter which specifies the number of seconds a delayed write buffer may age before being written out to disk. The default value of NAUTOUP is 60 seconds.

Each time it is called, the **fsflush** daemon writes some of the delayed–write buffers to disk. The number of pages the **fsflush** daemon writes each time is determined by the following formula:

```
pages_flushed = MEMORY * FDFLUSH/NAUTOUP
```

- **Range** – 1 – 1
- **Default** – 1

When to Change the Default

Normally, this parameter should not be changed. Increasing the value improves system performance, however it decreases system reliability.

FLCKREC

Description

The FLCKREC parameter specifies the maximum number of record locking regions (system–wide). One record locking region is required for each record (region) of a file that is locked.

- **Range** – 100 – 2000
- **Default** – 300

When to Change the Default

Change the FLCKREC parameter only if the **lockf**(2) system call is used. Increase its value if the number of regions in files to be locked exceeds or is approaching the current value. If your system uses record locking for an extremely large file or for many files, you may need to increase the FLCKREC parameter.

You may decrease the FLCKREC parameter if record locking is not used.

GPGSLO

Description

The GPGSLO parameter determines when processes are swapped out. If the amount of free memory is less than the value of GPGSLO, the swapper (**sched**) swaps out entire processes, one at a time, until the amount of free memory becomes greater than LOTSFREE.

The GPGSLO parameter does not affect paging. Paging is determined by the LOTSFREE, DESFREE, and MINFREE parameters.

The value of GPGSLO is specified in pages and should always be less than the value of MINFREE.

- **Range** – 0 –25
- **Default** – 25

When to Change the Default

Normally, you do not need to change the value of this parameter.

NOTE: When processes start up (at start–of–day or system start–up) the amount of free memory in the system may fluctuate between GPGSLO and LOTSFREE for about 20 minutes before stabilizing. This is a normal situation and probably can not be avoided.

HCORLIM

Description

The HCORLIM hard limit parameter specifies the maximum value of SCORLIM.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x100000 – 0x7FFFFFFF
- **Default** – 0x2000000

When to Change the Default

Increase the value of this parameter if you need to increase the value of SCORLIM beyond the value specified for HCORLIM.

HCPULIM

Description

The HCPULIM hard limit parameter specifies the maximum value of SCPULIM.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 60 – 0xFFFFFFF
- **Default** – 0x7FFFFFFF

When to Change the Default

You should not need to change the value of this parameter, because the default value provides unlimited resources.

HDATLIM

Description

The HDATLIM hard limit parameter specifies the maximum value of SDATLIM.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x1000000 – 0x7FFFFFFF
- **Default** – 0x1000000

When to Change the Default

Increase the value of this parameter if you need to increase the value of SDATLIM beyond the value specified for HDATLIM.

No matter how large HDATLIM is, the HVMMLIM parameter limits the total amount of address space that can be mapped to a process.

HFNOLIM

Description

The HFNOLIM hard limit parameter specifies the maximum value of SFNOLIM.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x20 – 0x400
- **Default** – 0x400

When to Change the Default

Increase the value of this parameter if you need to increase the value of SFNOLIM beyond the value specified for HFNOLIM.

HFSZLIM

Description

The HFSZLIM hard limit parameter specifies the maximum value of SFSZLIM.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x100000 – 0x7FFFFFFF
- **Default** – 0x7FFFFFFF

When to Change the Default

You should not need to change the value of this parameter, because the default value provides unlimited resources.

HRTIME

Description

The HRTIME parameter defines the size of the **hrtimes** array which keeps track of sleep and alarm requests for the standard, real–time clock.

- **Range** – 50 –50
- **Default** – 50

When to Change the Default

Normally, you should not need to change the value of this parameter.

HRVTIME

Description

Defines the size of the **itimes** array. This array keeps track of the alarm requests for the clocks measuring user process virtual time and process virtual time.

- **Range** – 50 – 50
- **Default** – 50

When to Change the Default

Normally, you should not need to change the value of this parameter.

HSTKLIM

Description

The HSTKLIM hard limit parameter specifies the maximum value of SSTKLIM.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x2000 – 0x7FFFFFFF
- **Default** – 0x1000000

When to Change the Default

Increase the value of this parameter if you need to increase the value of SSTKLIM beyond the value specified for HSTKLIM.

No matter how large HSTKLIM is, the HVMMLIM parameter limits the total amount of address space that can be mapped to a process.

HVMMLIM

Description

The HVMMLIM hard limit parameter specifies the maximum value of SVMMLIM.

The HVMMLIM parameter is closely related to the HDATLIM and HSTKLIM parameters.

Any process with an effective user ID of 0 (**root**) can use the **setrlimit** (see **getrlimit**(2)) system call to increase its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x2000 – 0x7FFFFFFF
- **Default** – 0x1000000

When to Change the Default

Increase the value of this parameter if you need to increase the value of SVMMLIM beyond the value specified for HVMMLIM.

HW_PROVIDER

Description

The HW_PROVIDER parameter specifies the hardware provider's name. The value is defined in the */etc/conf/pack.d/kernel/space.c* file, not in the *mtune* file.

- **Range** – N/A
- **Default** – NCR

When to Change the Default

You do not need to change the value of this parameter

HW_SERIAL

Description

The HW_SERIAL parameter can be used to specify the serial number of your machine; however, the default value is 0. The value is defined in the */etc/conf/pack.d/kernel/space.c* file, not in the *mtune* file.

- **Range** – N/A
- **Default** – 0

When to Change the Default

You do not need to change the value of this parameter.

ILDMAXOPENS

Description

The ILDMAXOPENS parameter identifies the maximum number of logical links that may be open in Ethernet operations. In Token Ring operations, ILDMAXOPENS identifies the maximum number of machines that may communicate on the ring.

- **Range** – 16 – 512
- **Default** – 64

When to Change the Default

You should not need to change the value of this parameter for Ethernet operations. For Token Ring operations, ILDMAXOPENS should be set to the same value as the NETBIOS NBSESSIONS parameter.

JQF_CLIENTS

Description

The maximum number of clients which can use a particular **jqf** queue at a time.

- **Range** – 100 – 3000
- **Default** – 2500

When to Change the Default

Increase when the desired number of clients exceeds the default setting.

JQF_SESS

Description

The maximum number of **jqf** instances which can be run simultaneously.

- **Range** – 10 – 50
- **Default** – 10

When to Change the Default

Increase when the desired number of sessions exceeds the default setting.

JQF_SIZE

Description

The maximum size of the data buffer that each slot can have for a **jqf** queue.

- **Range** – 40 –300
- **Default** – 40

When to Change the Default

Adjust this parameter to suit the particular applications using the JQF facitlity.

KDBSYMSIZE

Description

The KDBSYMSIZE parameter specifies the maximum size of the **kdb** symbol table. This table contains the name and address of each symbol used in the system, and is used by the kernel debugger.

- **Range** – 10000 – 400000
- **Default** – 100000

When to Change the Default

Increase the value of this parameter if there is not enough space for **kdb** symbols.

L5CPUenables

Description

The L5CPUenables parameter specifies how an ASIC is to be configured. This is a hardware configuration register value for NCR 3450 and 3550 computers.

- **Range** – 0 – 0xff
- **Default** – 0xef

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5DMAenables

Description

The L5DMAenables parameter specifies how an ASIC is to be configured. This is a hardware configuration register value for NCR 3450 and 3550 computers.

- **Range** – 0 – 0xff
- **Default** – 0xc0

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5McAddrenables

Description

The L5McAddrenables parameter specifies how an ASIC is to be configured. This is a hardware configuration register value for NCR 3450 and 3550 computers.

- **Range** – 0 – 0xff
- **Default** – 0xfd

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5McDataenables

Description

The L5McDataenables parameter specifies how an ASIC is to be configured. This is a hardware configuration register value for NCR 3450 and 3550 computers.

- **Range** – 0 – 0xff
- **Default** – 0x87

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5NMIenables

Description

The L5NMIenables parameter specifies how an ASIC is to be configured. This is a hardware configuration register value for NCR 3450 and 3550 computers.

- **Range** – 0 – 0xff
- **Default** – 0xff

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5SysIntenables

Description

The L5SysIntenables parameter specifies how an ASIC is to be configured. This is a hardware configuration register value for NCR 3450 and 3550 computers.

- **Range** – 0 – 0xff
- **Default** – 0

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5SysInt_Type

Description

The L5SysInt_Type parameter is a software switch for hardware error handling in NCR 3450 and 3550 computers.

- **Range** – 4 – 5
- **Default** – 5

When to Change the Default

Do not change the value of this parameter unless your NCR Representative instructs you to do so.

L5WatchDogPFRPeriod

Description

The L5WatchDogPFRPeriod is similar to the L5WatchDogPeriod parameter but is used to cover the time the memory is being saved to disk by the loader as a result of a power failure. This period must be long enough for this entire operation to take place since the loader does not give the Diagnostic Processor any acknowledgement. Note that the default is 1800 seconds or 30 minutes.

- **Range** – 0 – 0xffffffff
- **Default** – 1800

When to Change the Default

Normally, you do not need to change the value of this parameter.

L5WatchDogPeriod

Description

The L5WatchDogPeriod is the number of seconds the Diagnostics Processor will wait for an "I'm alive" acknowledgement from the system processors before resetting the system and dumping memory. The system processors normally give an acknowledgement every 10 ms in a clock interrupt. An expiring L5WatchDogPeriod means the heart beat of the system has stopped. In this event the system will be automatically restarted to maintain high availability.

- **Range** – 0 – 0xffffffff
- **Default** – 15

When to Change the Default

Normally, you do not need to change the value of this parameter.

LISTIO_MAX_CNT

Description

The maximum number of writes that can be done by a single *listio rwx()* system call.

- **Range** – 1 – 2048
- **Default** – 256

When to Change the Default

Increase this parameter when the desired number of writes exceeds the default setting.

LOTSFREE

Description

The LOTSFREE parameter specifies the amount of free memory that is considered enough. If the paging daemon finds fewer than LOTSFREE pages of free memory, it invokes **pageout** to free pages. If the value of LOTSFREE is 0, the kernel sets it to 1/8th of the pages of memory (not to exceed 256 pages).

The LOTSFREE parameter is closely related to the DESFREE and MINFREE parameters. Together, these three parameters control paging.

- **Range** – 0 – 512
- **Default** – 0

When to Change the Default

Normally, you do not need to change the value of this parameter.

MAXAIOS

Description

The MAXAIOS parameter indicates the total number of system daemons that can be created to perform asynchronous I/O.

- **Range** – 1 – 64
- **Default** – 5

When to Change the Default

The optimal number of asynchronous I/O daemons depends on the amount of asynchronous I/O activity and the number of devices to which asynchronous I/O is targeted. In general, you should allow two daemons per device used for asynchronous I/O.

Therefore, if more than two devices are to be used for asynchronous I/O, you should increase the value of MAXAIOS.

MAXCLSYSPRI

Description

The MAXCLSYSPRI parameter specifies the maximum global priority used by the system scheduling class for scheduling kernel processes. Changing this parameter changes the range of priorities used to schedule kernel processes and can have a significant effect on system performance.

- **Range** – 99 – 99
- **Default** – 99

When to Change the Default

In general, do not change this parameter unless you add new scheduling classes or reconfigure the priorities of other currently configured classes. If MAXCLSYSPRI is set to less than 39, the kernel automatically sets the value to 39 at boot time because it needs a range of 40 priorities for the system class (SYS).

MAXDMAPAGE

Description

The MAXDMAPAGE parameter specifies the maximum click number allowing Direct Memory Access (DMA). DMA is used by drivers to make memory transfers between processor boards without involving the processor.

- **Range** – 0 – 65536
- **Default** – 0

When to Change the Default

This parameter only needs to be modified for bus master adapters which only support 24–bit addresses. Currently, no such adapters are supported on the system, so this parameter should NOT require modification.

MAXGDP

Description

The MAXGDP parameter specifies the number of virtual circuits the computer can have open on the network. A circuit is set up when a machine mounts a resource of another machine.

- **Range** – 10 – 32
- **Default** – 24

When to Change the Default

Increase the value of this parameter if the number of virtual circuits needed exceeds the default value of MAXGDP.

MAXMINOR

Description

The MAXMINOR parameter specifies the largest minor number permitted on the system.

- **Range** – 255 – 0x3fffff
- **Default** – 0x3fffff

When to Change the Default

The configuration utilities (**mktable** for the Object Management Table, **findit** for ncrm) make use of the MAXMINOR parameter. If this number is altered, some system devices may NOT be located. Therefore, this parameter should, in most circumstances, NOT require modification.

MAXPMEM

Description

The MAXPMEM parameter specifies the maximum amount (in 4K pages) of physical memory to use. The default value of 0 means that all of physical memory can be used.

The maximum value of MAXPMEM depends on the amount of physical memory in the system: 4096 for a system with 16MB of memory, 8192 for a system with 32MB of memory, etc.

- **Range** – 0 – phys.mem.
- **Default** – 0

When to Change the Default

You can change this parameter to simulate the amount of memory in another system. If you have 32 MB of memory and you set MAXPMEM to 16 MB, programs will run as they would on a 16 MB memory system. If you do change the value to restrict the amount of physical memory used, be sure you allow enough memory to load the kernel.

As an alternative you can use the parameter MEMRANGE in the */stand/boot* file (**boot** (1M)). You can change the second number in the second field of the MEMRANGE parameter to increase or decrease the amount of memory. Note that changing the values for MEMRANGE in the */stand/boot* file does not require a kernel relink. You only have to reboot the system. Use the **memsize** command to show the amount of memory in the system and the **memsize –k** command to show the amount of memory configured for the kernel.

MAXSERVE

Description

The MAXSERVE parameter specifies the maximum number of servers on the system that can handle remote requests.

- **Range** – 3 – 6
- **Default** – 6

When to Change the Default

Modify the value of this parameter based on the output from sar–S which indicates server usage.

MAXSLICE

Description

The MAXSLICE parameter specifies the maximum value (in ticks) for the system time slice.

- **Range** – 25 – 100
- **Default** – 100

When to Change the Default

Normally, you should not change the value of this parameter.

MAXUP

Description

The MAXUP parameter specifies the maximum number of child processes permitted per parent process. This parameter does not limit the superuser (**root**).

The value of NPROC should be at least 10% greater than MAXUP.

- **Range** – 15 – 400
- **Default** – 30

When to Change the Default

To restrict the number of processes that each userid may create, decrease the MAXUP parameter.

If the same userid is used by more than one user, you may need to increase the MAXUP parameter. For example, if 12 people were logged in with the same userid, they would reach the limit very quickly.

MAXWCONSMSG

Description

The MAXWCONSMSG parameter specifies the maximum number of messages to the console.

- **Range** – 0 – 100
- **Default** – 80

When to Change the Default

Normally, you should not change the value of this parameter.

MAXWERRMSG

Description

The MAXWERRMSG parameter specifies the maximum number of error messages.

- **Range** – 0 – 100
- **Default** – 80

When to Change the Default

Normally, you should not change the value of this parameter.

MAXWTRCMSG

Description

The MAXWTRCMSG parameter specifies the maximum number of trace messages to be held by the kernel when the **strlogd** utility is NOT running.

- **Range** – 0 – 100
- **Default** – 80

When to Change the Default

Normally, you should not change the value of this parameter.

MINAKMEM

Description

The MINAKMEM parameter specifies the minimum available kernel memory (in 4K pages) required to avoid deadlock.

- **Range** – 4 – 64
- **Default** – 16

When to Change the Default

Normally, you should not change the value of this parameter.

MINARMEM

Description

The MINARMEM parameter specifies the minimum number of 4K pages of real memory reserved for the text and data segments of user processes. (This is the minimum number required to avoid deadlock.)

- **Range** – 100 – 160
- **Default** – 100

When to Change the Default

Increase this parameter if more user processes are added to the system and they seem to take longer to complete. No error message is generated if insufficient memory is available; however, processes may run slower because they wait and retry.

MINASMEM

Description

The MINASMEM parameter specifies the minimum number of 4K pages of memory and swap reserved for system purposes and, therefore, unavailable for the text and data segments of user processes. (This is the minimum number necessary to avoid deadlock.)

- **Range** – 25 – 40
- **Default** – 25

When to Change the Default

Increase this parameter as more processes are added to the system. An error message is generated if the system tries to exceed the reserved number of pages.

MINFREE

Description

The MINFREE parameter specifies the minimum amount (in pages) of free memory tolerable. At this value, only critical requests for memory are allowed.

If the value of MINFREE is 0, the kernel sets it to 1/32 of memory (not to exceed 64 pages).

The MINFREE parameter is closely related to the LOTSFREE and DESFREE parameters. Together, these three parameters control paging.

- **Range** – 0 – 128
- **Default** – 0

When to Change the Default

Normally, you do not need to change the value of this parameter.

If the value of MINFREE is less than the value of MINPAGEFREE, the system automatically sets MINFREE to MINPAGEFREE and increases LOTSFREE and DESFREE by the same amount.

MINPAGEFREE

Description

The MINPAGEFREE parameter specifies the number of pages that absolutely must be kept free.

This parameter is related to the LOTSFREE, DESFREE, and MINFREE parameters. If the value of MINFREE is less than the value of MINPAGEFREE, the system automatically sets MINFREE to MINPAGEFREE and increases LOTSFREE and DESFREE by the same amount.

- **Range** – 25 – 50
- **Default** – 25

When to Change the Default

Normally, you do not need to change the value of this parameter.

MINSERVE

Description

The MINSERVE parameter specifies the minimum number of servers that must be running on the system at all times. It is used by Remote File Sharing (**RFS**).

- **Range** – 3 – 3
- **Default** – 3

When to Change the Default

Modify the value of this parameter based on the output from **sar–S** which indicates server usage.

MSGMAP

Description

The MSGMSP parameter specifies the number of entries in the control map used to manage message segments. Each entry in this map represents a free area in the message buffer area.

- **Range** – 10 – 400
- **Default** – 100

When to Change the Default

Modify the MSGMAP parameter if the maximum number of message segments on the system (the MSGSEG parameter) is modified.

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

MSGMAX

Description

The MSGMAX parameter specifies the maximum size of a message (in bytes).

- **Range** – 512 – 8192
- **Default** – 2048

When to Change the Default

Increase this parameter if the maximum size of a message needs to be larger.

Decrease the value of this parameter to limit the size of messages.

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

MSGMNB

Description

The MSGMNB parameter specifies the maximum length (in bytes) of a message queue.

- **Range** – 4096 – 4096
- **Default** – 4096

When to Change the Default

Increase the value of this parameter if the maximum number of bytes on a message queue needs to be larger.

Decrease the value of this parameter to limit the number of bytes per message queue.

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

MSGMNI

Description

The MSGMNI parameter specifies the maximum number of message queues on the system. Each queue may contain many messages.

- **Range** – 50 –1000
- **Default** – 50

When to Change the Default

Increase this parameter to permit more message queues on the system.

Decrease the value of this parameter to limit the number of message queues.

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

MSGSEG

Description

The MSGSEG parameter specifies the number of message segments permitted on the system. Each message on a message queue consists of one or more message segments (the size of each segment is specified by the MSGSSZ parameter).

- **Range** – 1024 –1024
- **Default** – 1024

When to Change the Default

Increase the value of this parameter if all the messages that need to be sent do not fit in the space permitted for messages (MSGSEG * MSGSSZ).

Decrease the value of this parameter to limit the number of messages segments.

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

NOTE: When changing this value you are changing the amount of kernel memory buffer space reserved for messages. The amount of kernel memory buffers reserved for messages can be calculated by multiplying (MSGSEG * MSGSSZ). This value must be less than or equal 131,072 bytes (128 KB).

MSGSSZ

Description

The MSGSSZ parameter specifies the size (in bytes) of a message segment. Each message consists of a set of a contiguous set of message segments large enough to hold the text of the message.

- **Range** – 8 – 8
- **Default** – 8

When to Change the Default

Increase the value of this parameter if all the messages are a standard size (set MSGSSZ to that size).

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

NOTE: When changing this value you are changing the amount of kernel memory buffer space reserved for messages. The amount of kernel memory buffers reserved for messages can be calculated by multiplying (MSGSEG * MSGSSZ). This value must be less than or equal 131,072 bytes (128 KB).

MSGTQL

Description

The MSGTQL parameter specifies the number of system message headers. Each outstanding (unread) message requires one header; therefore, this parameter limits the number of outstanding messages.

- **Range** – 40 – 100
- **Default** – 40

When to Change the Default

Increase the value of this parameter if more outstanding messages are required.

Decrease the value of this parameter to limit the number of outstanding messages.

NOTE: Changing this parameter has no effect if the **msg** module is not configured into the kernel.

NADVERTISE

Description

The NADVERTISE parameter specifies the number of directories that a system can advertise as available for mounting by other systems using RFS.

- **Range** – 1 – 25
- **Default** – 25

When to Change the Default

Normally, you should not need to change the value of this parameter.

NAIOSYS

Description

The NAIOSYS parameter specifies the maximum number of outstanding asynchronous I/O system calls that can be active at any given time.

- **Range** – 0 – 512
- **Default** – 300

When to Change the Default

The optimal number of outstanding system calls for NAIOSYS depends on the amount of asynchronous I/O activity and the number of configured asynchronous I/O daemons (MAXAIOS). In general, there should be at least ten available outstanding asynchronous I/O system calls for each configured asynchronous I/O daemon. In other words, NAIOSYS = 10 * MAXAIOS.

Since the default value of NAIOSYS can handle MAXAIOS values up to 30, you should not have to change the value of NAIOSYS.

NAUTOPUSH

Description

The NAUTOPUSH parameter specifies the number of devices that can be configured to be auto–pushed onto a stream.

- **Range** – 32 –32
- **Default** – 32

When to Change the Default

The default value should be adequate for most systems. Increase the value of NAUTOPUSH to permit additional devices to be auto–pushed.

NAUTOUP

Description

The NAUTOUP parameter specifies the age, in seconds, that a delayed write buffer attains before **fsflush** causes it to be written to disk. It also determines the frequency for the automatic sync of inodes.

System buffers, other dirty pages, and other cached file attributes such as inodes are written to the hard disk when they have been memory–resident for the interval specified by the NAUTOUP parameter.

- **Range** – 0 – 120
- **Default** – 60

When to Change the Default

Normally, you do not need to change the value of this parameter.

Specifying a smaller value increases system reliability by writing the buffers to disk more frequently, but it decreases system performance. Specifying a larger value increases system performance at the expense of reliability.

NOTE: Do NOT set the value of NAUTOUP to 0; this causes a system panic.

NBUF

Description

The NBUF parameter specifies the number of system buffer headers initially available for block I/O.

When blocks are read from or written to disk, the information is buffered. That is, frequently used blocks stay in memory and writes of partial blocks occur in memory. This helps the system avoid time consuming physical I/O for disk access.

Each buffer must have a buffer header. When a buffer header is needed, but no free ones are available, the system dynamically allocates more buffer headers in chunks of NBUF at a time. Buffer headers are returned to the NBUF pool when no longer needed; however, unused buffer headers can not be returned to memory except through a system reboot.

Although there is no limit to the total number of buffer headers in the system, the BUFHWM parameter limits the total number of kilobytes that can be used by buffers and this effectively limits the number of buffer headers.

In UNIX SVR4, NBUF determines the number of system buffer headers; BUFHWM determines the amount of memory that can be used for buffers; and SEGMAPSZ determines the amount of memory available to store actual file data.

In previous releases, all information (inode, superblock, etc. as well as file data) was placed in the memory reserved for system buffers. In UNIX SVR4, buffers generally contain only inode, superblock(s) and file header information but not file data. The actual file data is stored in pages of virtual memory (the amount of virtual memory available for this purpose is determined by the SEGMAPSZ parameter).

- **Range** – 100 – 3000
- **Default** – 100

When to Change the Default

Since the system dynamically allocates additional chunks of buffer headers as needed, you probably do not need to increase the value of this parameter. Additionally, if you allocate the buffer headers in large chunks, you may reserve a large chunk of memory that is never needed.

If the value of NBUF is set larger than the default and the output of **sar –b** shows you are getting cache hits all the time, you have allocated more memory than necessary for block I/O buffers and should reduce the value of NBUF.

NCALL

Description

The NCALL parameter specifies the maximum number of call–out table entries to allocate (system–wide). The call–out table is used by device drivers to provide a timeout to make sure that the system does not hang when a device does not respond to commands.

- **Range** – 30 – 700
- **Default** – 60

When to Change the Default

If a new device driver that times actions is added to the system, you may need to increase this parameter.

A call–out table overflow causes the system to crash and output the following message on the system console:

```
PANIC: Timeout table overflow
```

NCDEXTENT

Description

The parameter NCDEXTENT specifies the number of **cdfs** file system extents allocated by the system. This value should be greater than or equal to NCDINODE.

- **Range** – 1 – 20
- **Default** – 5

When to Change the Default

If you previously changed the value of the NCDINODE parameter, you need to adjust the value of NCDEXTENT to be greater or equal to NCDINODE. If the system console displays a message that no more **cdfs** file system extents can be allocated or if no more **cdfs** files can be opened, you will also need to increase the value of NCDEXTENT.

NCDFILSYS

Description

The parameter NCDFILSYS specifies the number of **cdfs** file systems that can be mounted concurrently.

- **Range** – 200 – 5000
- **Default** – 500

When to Change the Default

Increase the value of NCDFILSYS in the case you need more than the default number of cdfs file systems mounted on your system.

NCDINODE

Description

The NCDINODE parameter specifies the number of **cdfs** inode table entries, and thereby limits the number of files in **cdfs** file systems that can be open at one time.

- **Range** – 100 – 2500
- **Default** – 250

When to Change the Default

If you receive a message on the console indicating that no more **cdfs** files can be opened or if you receive persistent messages that there is a **cdfs** file system inode table overflow, you should increase this parameter.

NCLIST

Description

The NCLIST parameter specifies the number of 64–character list buffers (cblocks). In earlier releases of NCR UNIX, the character list buffers were used as a temporary holding area for terminal I/O. Since TTY buffering is now handled through STREAMS, the NCLIST parameter is normally not used.

- **Range** – 1 – 400
- **Default** – 120

When to Change the Default

Normally, you do not need to change the value of this parameter.

NCPU

Description

The NCPU parameter specifies the maximum number of CPUs that the kernel supports.

- **Range** – 1 – 32
- **Default** – 8

When to Change the Default

Normally, you should not change the value of this parameter.

NCPYRIGHT

Description

The NCPYRIGHT parameter specifies the size of the OEM copyright table.

- **Range** – 10 – 10
- **Default** – 10

When to Change the Default

Increase the value of this parameter if the OEM copyright table needs to be larger.

NDQUOT

Description

The NDQUOT parameter limits the number of quota structures permitted for **ufs** file systems. There is one quota structure for each user, therefore NDQUOT should be greater than the maximum number of users that can be logged onto the system times the number of **ufs** file systems that are mounted.

If quotas are in effect, quota structures limit the amount of disk space each user can use.

- **Range** – 100 – 400
- **Default** – 200

When to Change the Default

If there are no available entries in the kernel quota table, the following message is displayed on the system console:

```
dquot table full
```

Increase the value of NDQUOT if you receive this message.

If quotas are not in effect or if you do not use **ufs** file systems, you may set the value of NDQUOT to 0.

NEMAP

Description

The NEMAP parameter specifies the maximum number of mappings allowed for characters in the TTY subsystem line discipline.

- **Range** – 10 – 10
- **Default** – 10

When to Change the Default

Normally, you should not need to change the value of this parameter.

NGROUPS_MAX

Description

The NGROUP_MAX parameter specifies the maximum number of groups to which a process may belong. See **getgroups**(1).

- **Range** – 1 – 16
- **Default** – 16

When to Change the Default

Increase the value of this parameter only if you want to increase the number of groups in which a process may have membership.

NHBUF

Description

The NHBUF parameter specifies the number of hash table entries that can be allocated in the system. The hash table is used in searching for buffer headers (and, therefore, for buffers).

The value of NHBUF must be a power of 2.

- **Range** – 32 – 1024
- **Default** – 64

When to Change the Default

Increase the value of NHBUF to make searching for buffers more efficient.

NINODE

Description

The NINODE parameter specifies the maximum number of inodes that can be in use in s5 file systems at any one time. The value of this parameter should always be greater than **ncsize** in the */etc/conf/pack.d/kernel/space.c* file.

The UFSNINODE parameter determines the number of inodes for **ufs** file systems.

- **Range** – 100 – 5000
- **Default** – 500

When to Change the Default

There must be an inode for each file and directory in the file system; therefore, you may need to increase the value of NINODE if you create a lot of files in **s5** file systems.

If **sar –v** shows that table overflows are occurring or if **sar –g** shows that **%s5ipf** is greater than 10 percent, you should increase the value of NINODE.

If **sar –v** consistently shows that the inode table is underutilized, you may lower the value of NINODE.

NLOG

Description

The NLOG parameter specifies the number of minor devices to be configured for the log driver.

- **Range** – 3 – 16
- **Default** – 10

When to Change the Default

Increase the value of this parameter if you add daemons for applications that may submit log messages.

NMOUNT

Description

The NMOUNT parameter specifies the maximum number of file systems that can be mounted. This includes the *root* file system.

- **Range** – 25 – 25
- **Default** – 25

When to Change the Default

Increase this parameter if the number of file systems that must be mounted is greater than the current value (remember that the first entry is always used for the *root* file system).

NODE

Description

The NODE parameter specifies the node name of the system. If you are on a network, NODE should be a unique name because it is used to identify your system to the network.

NODE is defined in the */etc/conf/pack.d/kernel/space.c* file, not in the *mtune* file. You can use the **setuname** command to change the node name.

- **Range** – N/A
- **Default** – unix

When to Change the Default

Be careful about changing the NODE parameter if you are using a network. If the node name changes, your system may not be known to the network.

NOTPGOVERFLOW

Description

The NOTPGOVERFLOW parameter specifies the number of buffers reserved for scheduler I/O during low memory situations.

- **Range** – 8 – 128
- **Default** – 16

When to Change the Default

Increase the value of this parameter if you need more buffers for scheduler I/O during low memory situations.

NPROC

Description

The NPROC parameter specifies the number of process table entries to allocate and, therefore, the maximum number of processes that can be created (system–wide).

Each process table entry represents an active process. The swapper is always the first entry and **/sbin/init** is always the second entry.

The value of NPROC should be at least 10% greater than MAXUP.

- **Range** – 50 – 5104
- **Default** – 250

When to Change the Default

If processes are prevented from forking (being created), increase the NPROC parameter.

NOTE: Be sure that the prevention of forking is a system–wide problem, not just a user ID problem (see the MAXUP parameter).

NRCVD

Description

The NRCVD parameter specifies the number of receive descriptors configured. A receive descriptor is allocated for each file, directory, or process waiting for a remote request.

- **Range** – 40 – 500
- **Default** – 150

When to Change the Default

Increase the value of this parameter if you receive errors when accessing remote information.

NRDUSER

Description

The NRDUSER parameter specifies the maximum number of receive descriptor user entries that can be allocated. There can be multiple receive descriptor user entries for each receive descriptor entry.

- **Range** – 1 – 700
- **Default** – 250

When to Change the Default

The value of NRDUSER should be about 1.5 times the value of NRCVD.

NRNODE

Description

The NRNODE parameter specifies the maximum number of remote nodes the kernel can communicate with using NFS.

- **Range** – 100– 1300
- **Default** – 300

When to Change the Default

Increase the value of this parameter if the number of NFS clients exceeds the default value.

NSCRN

Description

The NSCRN parameter specifies the number of virtual terminals permitted on the system. It is present only for compatibility with XENIX drivers and applications.

- **Range** – 0 – 10
- **Default** – 0

When to Change the Default

Increase the value of this parameter to permit additional screens.

NSNDD

Description

The NSNDD parameter specifies the maximum number of send descriptors that can be allocated. A send descriptor is allocated for each remote resource a user references.

- **Range** – 100 – 350
- **Default** – 150

When to Change the Default

Increase the value of this parameter if you receive error messages for user commands.

NSRMOUNT

Description

The NSRMOUNT parameter specifies the maximum number of remote mounts allowed by clients.

- **Range** – 1 – 50
- **Default** – 20

When to Change the Default

Increase the value of this parameter if the number of mounts attempted by remote clients exceeds the configured limit.

NSTRPHASH

Description

The NSTRPHASH parameter specifies the size of the auto–push hash table. The value must be a power of 2.

- **Range** – 16 – 512
- **Default** – 64

When to Change the Default

You should not need to change this parameter unless the number of drivers on the system gets extremely large.

NSTRPUSH

Description

The NSTRPUSH parameter specifies the maximum number of modules that may be pushed onto a single stream. This is used to prevent an errant user process from consuming all of the available queues on a single stream.

- **Range** – 9 – 10
- **Default** – 9

When to Change the Default

You should not need to change the value of this parameter. Existing applications normally push no more than 4 modules onto a stream.

NUMSAD

Description

The NUMSAD parameter specifies the maximum number of STREAMS administrator driver devices that can be opened at any given time.

- **Range** – 1 – 16
- **Default** – 8

When to Change the Default

Increase the value of this parameter if you need to open more than the default number of administrator driver devices.

NUMSP

Description

The NUMSP parameter specifies the maximum number of stream pipes available if the multiplexing STREAMS pipe driver (**sp**) is included in the kernel.

- **Range** – 5 – 50
- **Default** – 32

When to Change the Default

Increase the value of this parameter if you need more than the default number of STREAMS pipes.

NUMSXT

Description

The NUMSXT parameter specifies the number of shell layer sessions that may be in progress at any one time (system–wide).

- **Range** – 1 – 6
- **Default** – 6

When to Change the Default

Normally, you do not need to change the value of this parameter. However, you may increase the value if many users use the shell layers feature.

NUMTIM

Description

The NUMTIM parameter specifies the maximum number of **timod** STREAMS modules available if the STREAMS transport interface cooperating module (**timod**) is included in the system.

- **Range** – 1 – 2048
- **Default** – 128

When to Change the Default

The value of NUMTIM should be at least as large as the maximum number of simultaneous transport interface connections that the system must handle. Increase the value of this parameter if you expect more than the default number of **timod** STREAMS modules.

NOTE: Changing the value of this parameter has no effect unless the **timod** module is configured into the kernel.

NUMTRW

Description

The NUMTRW parameter specifies the maximum number of transport interface read/write STREAMS modules available if the STREAMS transport interface read/write module (**tirdwr**) is included in the system.

- **Range** – 1 – 2048
- **Default** – 16

When to Change the Default

Increase the value of this parameter if you need additional **tirdwr** STREAMS modules. The default value should be adequate for most systems.

NOTE: Changing the value of this parameter has no effect unless the **tirdwr** module is configured into the kernel.

NUMXT

Description

The NUMXT parameter specifies the maximum number of **xt** terminals to support.

- **Range** – 1 – 3
- **Default** – 3

When to Change the Default

The default value should be adequate for most systems.

PAGES_UNLOCK

Description

The PAGES_UNLOCK parameter specifies the maximum number of pages that can be locked in memory at one time.

- **Range** – 200 –200
- **Default** – 200

When to Change the Default

You should not need to change the value of this parameter.

PGOVERFLOW

Description

The PGOVERFLOW parameter specifies the number of buffers reserved for the **pageout** daemon to use during low memory situations.

- **Range** – 8 – 128
- **Default** – 16

When to Change the Default

Increase the value of this parameter if the **pageout** daemon needs more buffers in low memory situations in order to maintain system operation.

PIOMAP

Description

The PIOMAP parameter specifies the number of map structures used to manage the physical I/O address space used during raw I/O.

- **Range** – 50 – 50
- **Default** – 50

When to Change the Default

Normally, you do not need to change the value of this parameter.

PIOMAXSZ

Description

The PIOMAXSZ parameter specifies the maximum number of pages of data that can be transferred in one I/O operation during raw I/O.

- **Range** – 4 – 64
- **Default** – 64

When to Change the Default

Normally, you do not need to change the value of this parameter.

PIOSEGSZ

Description

The PIOSEGSZ specifies the size (in pages) of a segment for raw I/O.

- **Range** – 1024 – 1024
- **Default** – 1024

When to Change the Default

Normally, you do not need to change the value of this parameter.

PRFMAX

Description

The PRFMAX parameter specifies the number of entries in the symbol table that is used for kernel profiling. Each entry in this table is a counter that corresponds to a text symbol in the kernel symbol table. Thus, PRFMAX determines the number of unique text addresses that can be monitored (analyzed) at any one time by the kernel–level profiler.

- **Range** – 2048 – 10240
- **Default** – 4096

When to Change the Default

Increase the value of this parameter if kernel–level profiling can not be performed.

PUTBUFSZ

Description

The PUTBUFSZ parameter specifies the size (in bytes) of the kernel common error message buffer. If the buffer is filled, the oldest messages are dropped as new ones are added.

- **Range** – 2000 – 10000
- **Default** – 2000

When to Change the Default

Normally, you should not need to change the value of this parameter.

PW_MAX_EVENT

Description

The maximum number of events that can be waited on by a Post Wait Driver instance.

- **Range** – 0 – 300
- **Default** – 100

When to Change the Default

Increase this parameter when the desired number of events exceeds the default setting.

PW_MAX_ORA_PID

Description

The maximum number of simultaneous users at a given time of the Post Wait Driver.

- **Range** – 0 – 300
- **Default** – 100

When to Change the Default

Increase this parameter when the desired number of users exceeds the default setting.

RCACHETIME

Description

The RCACHETIME parameter specifies the number of seconds that caching is turned off after a write to a networked file. A value of –1 indicates no caching of shared files.

- **Range** – minus 1 – 10
- **Default** – 10

When to Change the Default

Change the value of this parameter to adjust the length of time caching is turned off.

RCMF

Description

The RCMF parameter is a flag specifying whether to send messages to the remote system console. A value of 0 indicates NOT to send messages; a value of 1 indicates send messages.

- **Range** – 0 – 1
- **Default** – 0

When to Change the Default

Set the value of RCMF to 1 if you want messages sent to the remote system console.

REL

Description

The REL parameter specifies the NCR UNIX system release. It is defined in the */etc/conf/pack.d/kernel/space.c* file, not in the *mtune* file.

- **Range** – N/A
- **Default** – 4.0

When to Change the Default

The value of this parameter is set during installation; you do not need to change it.

RF_MAXKMEM

Description

The RF_MAXKMEM parameter specifies the maximum amount (in bytes) of kernel memory that can be used by RFS users. A value of 0 indicates no limit.

- **Range** – 0 – 50
- **Default** – 0

When to Change the Default

Normally, you should not need to change the value of this parameter.

Decreasing the value limits the amount of kernel memory that RFS users can use.

ROOTFSTYPE

Description

The ROOTFSTYPE parameter specifies the file system type of the *root* file system. This is used by the system to determine the format of the *root* file system.

- **Range** – s5 ufs
- **Default** – s5

When to Change the Default

The value of this parameter is set, during installation, to match the type specified for the *root* file system. You should not change it.

RSTCHOWN

Description

The RSTCHOWN parameter is the restricted file ownership changes flag.

A value of 0 indicates System V Release 3 compatibility mode. In this mode, the owner of a file can change the user ID and group ID of the file to any value, including nonexistent user and group IDs.

A value of 1 for RSTCHOWN indicates FIPS/BSD compatibility mode. In this mode, only root processes (those with a UID of 0) are able to change the ownership of a file. The owner of the file may only change the group ID of the file to one of the groups in which the owner has membership. Root processes may change the group ID of any file to any value.

- **Range** – 0 – 1
- **Default** – 0

When to Change the Default

Change the value of RSTCHOWN to 1 to restrict users from changing the user and group IDs of their files. Be aware that setting RSTCHOWN to 0 adversely affects system security.

RTNPROCS

Description

The RTNPROCS specifies the number of real–time processes that can exist on the system at any one time.

- **Range** – 60 – 60
- **Default** – 60

When to Change the Default

Increase the value of this parameter to permit more simultaneous real–time processes to exist on the system.

SATENABLED

Description

The SATENABLED parameter is a flag to indicate whether the system uses security audit trails. A value of 0 indicates that security audit trails are not enabled.

- **Range** – 0 – 1
- **Default** – 0

When to Change the Default

Do not change the value of this parameter. It is changed automatically by the addition or removal of the **sat** package.

SCORLIM

Description

The SCORLIM soft limit parameter specifies the largest size (in bytes) of a core file that may be created. A value of 0 prevents the creation of core files. The value of SCORLIM can not be set larger than the hard limit parameter HCORLIM.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit.

- **Range** – 0x100000 – 0x7FFFFFFF
- **Default** – 0x2000000

When to Change the Default

Increase the value of this parameter to permit creation of larger core files.

SCPULIM

Description

The SCPULIM soft limit parameter specifies the maximum combined user and system CPU time (in seconds) that a process is allowed. A SIGXCPU signal is sent to any process that exceeds this value. The value of SCPULIM can not be set larger than the hard limit parameter HCPULIM.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 60 – 0x7FFFFFFF
- **Default** – 0x7FFFFFFF

When to Change the Default

The default value indicates unlimited resources. Normally, you should not need to change this value.

SDATLIM

Description

The SDATLIM soft limit parameter specifies the maximum size (in bytes) of a process's data segment. If a process attempts to extend its data segment beyond this limit using **brk**(2), the attempt fails and **errno** is set to ENOMEM. The value of SDATLIM can not be set larger than the hard limit parameter HDATLIM.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x1000000 – 0x7FFFFFFF
- **Default** – 0x1000000

When to Change the Default

Increase the value of this parameter to permit creation of larger data segments for processes.

Regardless of the value of SDATLIM, the SVMMLIM parameter limits the total amount of address space that can be mapped to a process.

SEGMAPSZ

Description

The SEGMAPSZ parameter specifies the size of the bitmap for file I/0. This map determines the number of 4K pages and therefore the amount of memory that can be used for file I/O. If the value is 0, the kernel calculates the value as a function of the amount of physical memory.

- If physical memory is not more than 4MB, the amount of memory available for file I/0 is 80% of (available memory – 2MB).
- If physical memory is greater than 4MB, the amount of memory available for file I/0 is 80% of (available memory – 4MB).
- **Range** – 0 – 4096
- **Default** – 0

When to Change the Default

Normally, you do not need to change the value of this parameter.

SEMAEM

Description

The SEMAEM parameter specifies the maximum adjust–on–exit value for a semaphore in an undo structure. This value is used when a semaphore value becomes greater than or equal to the absolute value of **semop**(2), unless the program has set its own value.

- **Range** – 16384 – 16384
- **Default** – 16384

When to Change the Default

Modify the SEMAEM parameter to increase or decrease the maximum value for an adjust–on–exit value in an undo structure.

This parameter is closely related to the SEMVMX parameter (the maximum value for a semaphore). The value of SEMAEM should not exceed the value of SEMVMX.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMMAP

Description

The SEMMAP parameter specifies the number of entries in the control map used to manage semaphores. This map is used to keep track of free areas in the system pool of semaphores.

- **Range** – 10 – 1000
- **Default** – 25

When to Change the Default

Modify the SEMMAP parameter if the maximum number of semaphores on the system (the SEMMNS parameter) is modified.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMMNI

Description

The SEMMNI parameter specifies the maximum number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time.

Semaphores are created in sets, and there may be more than one semaphore per set. The semaphore identifier permits a process to modify more than one semaphore in a set at the same time.

- **Range** – 10 – 1000
- **Default** – 25

When to Change the Default

Increase the SEMMNI parameter if processes require more than the default number of semaphore sets.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMMNS

Description

The SEMMNS parameter specifies the maximum number of semaphores permitted in the system.

- **Range** – 60 – 1000
- **Default** – 60

When to Change the Default

Increase the value of the SEMMNS parameter if processes require more than the default number of semaphores.

Modify the SEMMAP parameter if the SEMMNS parameter is modified.

NOTE: Changing the value of this parameter has no effect if the sem module is not configured into the kernel.

SEMMNU

Description

The SEMMNU parameter specifies the maximum number of undo structures in the system. An undo structure, which is set up on a per–process basis, keeps track of process operations on semaphores so the operations may be undone if the process terminates abnormally.

Each undo structure has an adjust–on–exit value for each semaphore. When an operation is performed on the semaphore, the direct opposite action is reflected in the adjust–on–exit value in the undo structure. This value is added to the semaphore when the process terminates. The SEMMNU parameter limits the number of processes that can specify the UNDO flag in the **semop**(2) system call to undo their operations on termination.

The undo capability helps to ensure that a process that terminates abnormally does not cause other processes to wait indefinitely for a change to a semaphore.

- **Range** – 30 – 1000
- **Default** – 30

When to Change the Default

Modify the SEMMNU parameter to increase or decrease the number of undo structures permitted on the system.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMMSL

Description

The SEMMSL parameter specifies the maximum number of semaphores per semaphore identifier (set).

- **Range** – 25 – 1000
- **Default** – 25

When to Change the Default

Increase the SEMMSL parameter if processes require more than the default number of semaphores per semaphore set.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMOPM

Description

The SEMOPM parameter specifies the maximum number of semaphore operations per **semop(2)** system call. This parameter permits the system to check or modify the value of more than one semaphore in a set with each **semop(2)** system call.

- **Range** – 10 – 10
- **Default** – 10

When to Change the Default

Modify the SEMOPM parameter to increase or decrease the number of operations permitted per **semop**(2) system call.

This parameter may need to be increased if the SEMMSL parameter (number of semaphores per set) is increased, so that a process can check and/or modify all the semaphores in a set with one system call.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMUME

Description

The SEMUME parameter specifies the maximum number of undo entries per undo structure. Each undo entry represents a semaphore that has been modified with the UNDO flag specified in the **semop(2)** system call.

The amount of memory required for the SEMUME parameter depends on the SEMMNU parameter. See the SEMMNU parameter.

- **Range** – 10 – 10
- **Default** – 10

When to Change the Default

Modify the SEMUME parameter to increase or decrease the number of undo entries per structure.

This parameter is closely related to the SEMOPM parameter (the number of operations per **semop**(2) system call). Normally the value of SEMUME should match the value of SEMOPM.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SEMVMX

Description

The SEMVMX parameter specifies the maximum value for a semaphore. This is used to prevent wraparound. Do not set the value above 32767; this is the maximum field width.

- **Range** – 32767 – 32767
- **Default** – 32767

When to Change the Default

Decrease the SEMVMX parameter to limit the maximum value for a semaphore.

NOTE: Changing the value of this parameter has no effect if the **sem** module is not configured into the kernel.

SFNOLIM

Description

The SFNOLIM soft limit parameter specifies the maximum number of file descriptors (open files) a process may have. The value of SFNOLIM can not be set larger than the HFNOLIM hard limit parameter.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

In NCR UNIX System V Release 3, the NOFILES parameter specified the number of open files per process.

- **Range** – 0x20 – 0x400
- **Default** – 0x40

When to Change the Default

Increase the value of this parameter to permit more file descriptors per process.

SFSZLIM

Description

The SFSZLIM soft limit parameter specifies the maximum size (in bytes) of any single file that can be created by a process. The value of SFSZLIM can not be set larger than the hard limit parameter HFSZLIM.

A SIGXFSX signal is sent to a process that attempts to write a file whose offset is greater than this value. In addition, the write fails with an EFBIG error.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

The file size for a user may also be limited by the ULIMIT shell parameter for the user login. If a user is having difficulty creating large files, check the ULIMIT parameter for that user.

- **Range** – 0x100000 – 0x7FFFFFFF
- **Default** – 0x7FFFFFFF

When to Change the Default

The default value indicates unlimited resources. Normally, you should not need to change the value of this parameter.

SHLBMAX

Description

The SHLBMAX parameter specifies the maximum number of shared libraries that can be attached to a process at one time. This applies only to COFF shared executables.

- **Range** – 2 – 6
- **Default** – 3

When to Change the Default

Increase the value of this parameter if COFF executable processes need to attach more than the default number of shared libraries.

SHM_NAILED_GID[1–9]

Description

Specifies which group of users is able to nail memory.

- **Range** – 0 – 2147483647
- **Default** – 0

When to Change the Default

Change this when you have a group that should have permission to nail memory.

SHMMAX

Description

The SHMMAX parameter specifies the maximum size (in bytes) of a shared memory segment. The value of SHMMAX must be less than the value of SHMALL.

- **Range** – 131072 – 209715200
- **Default** – 524288 (512K)

When to Change the Default

Decrease the value of this parameter to limit the amount of memory an individual shared memory segment can use.

NOTE: Changing the value of this parameter has no effect if the **shm** module is not configured into the kernel.

SHMMIN

Description

The SHMMIN parameter specifies the minimum size (in bytes) of a shared memory segment.

- **Range** – 1 – 1
- **Default** – 1

When to Change the Default

Normally, you do not need to change the value of this parameter. The value of SHMMIN must be less than the value of SHMMAX.

NOTE: Changing the value of this parameter has no effect if the **shm** module is not configured into the kernel.

SHMMNI

Description

The SHMMNI parameter specifies the maximum number of shared memory identifiers permitted on the system at any one time.

- **Range** – 100 – 500
- **Default** – 100

When to Change the Default

Normally, you do not need to change the value of this parameter.

Increase the SHMMNI parameter if processes use so many shared memory segments that this limit is reached. Decrease this parameter to reduce the amount of memory used by shared memory segments.

NOTE: Changing the value of this parameter has no effect if the **shm** module is not configured into the kernel.

SHMSEG

Description

The SHMSEG parameter specifies the maximum number of attached shared memory segments per process. A process must attach the shared memory segment before the data can be accessed.

The maximum number of shared memory segments that can be attached per process depends on the available unused space the process has available. Even if a process has fewer than SHMSEG shared memory segments, it may not be able to attach another because of its limited space.

- **Range** – 6 – 15
- **Default** – 6

When to Change the Default

Normally, you should not need to change the value of this parameter. If the SHMSEG parameter is too large, the amount of memory required to keep track of attached shared memory segments increases.

NOTE: Changing the value of this parameter has no effect if the **shm** module is not configured into the kernel.

SOCK_HIWATER

Description

The SOCK_HIWATER parameter specifies the high water mark for the sockmod module. This value limits the amount of data (number of bytes) that can be placed on sockmod's read or write queue.

- **Range** – 512 – 65535
- **Default** – 2048

When to Change the Default

The value of SOCK_HIWATER should be increased to delay invocation of flow control in sockmod. This can potentially improve performance, especially if bursts of data greater in size than the default value of SOCK_HIWATER are being passed to sockmod. In some cases, modules below sockmod will discard data messages if sockmod is in the flow control mode, resulting in a time-out and retransmission of data.

SOCK_LOWATER

Description

The SOCK_LOWATER parameter specifies the low water mark for the sockmod module. After the high water mark is reached, the number of bytes on sockmod's queue must drop below this value before flow control is turned off.

- **Range** – 128 – 65535
- **Default** – 1024

When to Change the Default

The value of SOCK_LOWATER should be changed in conjunction with SOCK_HIWATER. It's value must always be less than SOCK_HIWATER.

NOTE: The value of SOCK_LOWATER must always be less than SOCK_HIWATER.

SRPC_DOMAIN

Description

The SRPC_DOMAIN parameter specifies the name of the "Secure RPC Domain," the realm in which a uid space is unique. When using secure RPC, each user is assigned a "netname" which is of the form:

```
Operating_System.UserID@Realm
```

If the user ID of each user is the same for machines A, B, and C, the SRPC_DOMAIN parameter should be set to the same name on A, B, and C.

Refer to the *Programmer's Guide: Networking Interfaces* for more information about secure domain.

- **Range** – N/A
- **Default** – NULL

When to Change the Default

Change the value of this parameter if you use secure domain.

SSTKLIM

Description

The SSTKLIM soft limit parameter specifies the maximum size (bytes) of the stack segment for a process. This defines the limit of automatic stack growth by the system. A SIGSEGV signal is sent to a process that attempts to grow the stack beyond this value. Unless the process has arranged to catch this signal on a separate stack (see **signalstack**(2)), this terminates the process.

The value of SSTKLIM can not be set larger than the HSTKLIM hard limit parameter.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x2000 – 0x7FFFFFFF
- **Default** – 0x1000000

When to Change the Default

Modify the value of this parameter to limit or increase the size of the stack segment for a process.

Regardless of the value of SSTKLIM, the SVMMLIM parameter limits the total amount of address space that can be mapped to a process.

STRCTLSZ

Description

The STRCTLSZ parameter specifies the maximum size (in bytes) of the control portion of any STREAMS message. The control portion of a **putmsg** message is not subject to the constraints of the minimum/maximum packet size, so the value entered here is the only way to provide a limit for the control part of a message.

- **Range** – 1024 – 1024
- **Default** – 1024

When to Change the Default

The default value should be more than enough for existing applications.

STRMSGSZ

Description

The STRMSGSZ parameter specifies the maximum size (in bytes) of the data portion of a STREAMS message. This should usually be set just large enough to accommodate the maximum packet size restrictions of the configured STREAMS modules. If it is larger than necessary, a single **write**(2) or **putmsg**(2) can consume an inordinate number of message blocks.

- **Range** – 0 – 32767
- **Default** – 0

When to Change the Default

A value of 0 causes the maximum message size to be the maximum message size of the top module on the stream. If the top module has no maximum message size, the message size is the size of the **write** system call or the size of the message of the **putmsg** system call. A value of 4096 is sufficient for most existing applications.

STRTHRESH

Description

The STRTHRESH parameter specifies the maximum total bytes that streams are allowed to allocate. When the threshold is reached, users without the appropriate privilege are not allowed to open streams, push streams modules, or write to streams devices. Users with appropriate privilege are not affected. The threshold applies to outgoing streams only.

- **Range** – 0 – 0x1000000
- **Default** – 0x200000

When to Change the Default

The value of STRTHRESH should be set to between 25% and 50% of the total system memory. A value of 0 indicates no threshold. A value of 2MB is appropriate for a system with 4MB of memory; a value of 2MB–4MB is appropriate for a system with 8MB of memory; etc.

Increase the value of this parameter if users without the appropriate privilege need more bytes for streams.

SVMMLIM

Description

The SVMMLIM soft limit parameter specifies (in bytes) the maximum amount of address space (data, shared memory, and memory mapped files, stack, etc.) that can be mapped to a process. Attempts to increase a process's address space beyond this value (for example, with **brk**(2), **shmat**(2), or **mmap**(2)) fail with a ENOMEM error.

The SVMMLIM parameter is closely related to the SDATLIM and SSTKLIM parameters.

Any process can use the **setrlimit** (see **getrlimit**(2)) system call to increase its soft limit. A process with an effective user ID of 0 (**root**) can also raise its hard limit.

The value of SVMMLLIM can not be set larger than the HVMMLIM hard limit parameter. A value of 0x7FFFFFFF indicates a resource without limit.

- **Range** – 0x1000000– 0x7FFFFFFF
- **Default** – 0x1000000

When to Change the Default

Increase the value of this parameter to increase the amount of address space that can be mapped to a process.

SYS

Description

The SYS parameter specifies the system name. It is defined in the */etc/conf/pack.d/kernel/space.c* file, not in the *mtune* file. The value is the one you assign during system software installation.

The value of SYS must be less than 8 characters; it can not be blank ("") or "unix".

- **Range** – N/A
- **Default** – UNIX_System_V

When to Change the Default

The value of this parameter is set during installation; you do not need to change it.

TIM_HIWATER

Description

The TIM_HIWATER parameter specifies the high water mark for the timod module. This value limits the amount of data (number of bytes) that can be placed on timod's read or write queue.

- **Range** – 512 – 1048576
- **Default** – 65536

When to Change the Default

The value of TIM_HIWATER should be increased to delay invocation of flow control in timod. This can potentially improve performance, especially if bursts of data greater in size than the default value of TIM_HIWATER are being passed to timod. In some cases, modules below timod will discard data messages if timod is in the flow control mode, resulting in a time-out and retransmission of data.

TIM_LOWATER

Description

The TIM_LOWATER parameter specifies the low water mark for the timod module. After the high water mark is reached, the number of bytes on timod's queue must drop below this value before flow control is turned off.

- **Range** – 512 – 1048576
- **Default** – 16384

When to Change the Default

The value of TIM_LOWATER should be changed in conjunction with TIM_HIWATER.

NOTE: The value of TIM_LOWATER must always be less than TIM_HIWATER.

TIMEZONE

Description

The TIMEZONE parameter specifies the number of minutes that your time differs from Greenwich Mean Time (GMT).

- **Range** – 0 – 1440
- **Default** – 480

When to Change the Default

The TIMEZONE parameter is set during installation, and you should not need to change it unless you move the system to a location in another time zone.

TRACESZ

Description

The TRACESZ parameter specifies the size of the kernel trace buffer .

- **Range** – 4096 – 8192
- **Default** – 8192

When to Change the Default

This parameter is used ONLY during kernel development and debug, and is NOT used during normal system operation.

TRW_HIWATER

Description

The TRW_HIWATER parameter specifies the high water mark for the tirdwr module. This value limits the amount of data (number of bytes) that can be placed on tirdwr's read or write queue.

- **Range** – 512 – 1048576
- **Default** – 65536

When to Change the Default

The value of TRW_HIWATER should be increased to delay invocation of flow control in tirdwr. This can potentially improve performance, especially if bursts of data greater in size than the default value of TRW_HIWATER are being passed to tirdwr. In some cases, modules below tirdwr will discard data messages if tirdwr is in the flow control mode, resulting in a time-out and retransmission of data.

TRW_LOWATER

Description

The TRW_LOWATER parameter specifies the low water mark for the tirdwr module. After the high water mark is reached, the number of bytes on tirdwr's queue must drop below this value before flow control is turned off.

- **Range** – 512 – 1048576
- **Default** – 16384

When to Change the Default

The value of TRW_LOWATER should be changed in conjunction with TRW_HIWATER. It's value must always be less than TRW_HIWATER.

NOTE: The value of TRW_LOWATER must always be less than TRW_HIWATER.

TSMAXUPRI

Description

The range within which users may adjust the user priority of a time–sharing process is (–TSMAXUPRI to +TSMAXUPRI). Configuring a higher value gives users more control over the priority of their processes.

- **Range** – 10 –30
- **Default** – 20

When to Change the Default

Increase the value of this parameter to give users more control over the priority of their processes; decrease the value to give users less control over their processes.

NOTE: Only someone with **root** permissions can change the value of TSMAXUPRI.

TSNPROCS

Description

The TSNPROCS parameter specifies the number of timesharing processes that can exist on the system at one time. (Exist means to be on the run queue.)

- **Range** – 60 – 60
- **Default** – 60

When to Change the Default

Increase the value of this parameter to permit more time–sharing processes to coexist on the run queue.

TUBESIZE

Description

This parameter determines the maximum size of the tube buffer.

- **Range** – 1024 – 4096
- **Default** – 1024

When to Change the Default

Normally, you should not need to change the value of this parameter.

UFSNINODE

Description

The UFSNINODE parameter specifies the number of **ufs** inode table entries, and thereby limits the number of files in **ufs** file systems that can be open at one time. The value of UFSNINODE should be greater than the value specified for **ncsize** in the file */etc/conf/pack.d/kernel/space.c*.

If the root file system is **ufs** and many small files are used, you may need to increase the value of UFSNINODE. If predominantly very large files and/or raw partitions are used, you need fewer inodes and the default should be more than adequate.

- **Range** – 100 – 5000
- **Default** – 500

When to Change the Default

If **sar –v** shows that table overflows are occurring or if **sar –g** shows that **%s5ipf** is greater than 10 percent, you should increase the value of UFSNINODE.

If **sar –v** consistently shows that the inode table is underutilized, you may lower the value of UFSNINODE.

ULIMIT

Description

The ULIMIT shell parameter specifies the maximum size (in bytes) of a single file for each login to which it applies. You may place a ULIMIT entry in the */etc/default/login* file so it will apply to all users, or place an entry in the *.profile* file for particular users.

NOTE: The ULIMIT parameter is a shell parameter, not a kernel parameter. The value of ULIMIT in the *mtune* file is not referenced and does not restrict anything.

In NCR UNIX System V Release 4, the kernel parameters that specify the maximum size of a file are the hard and soft limit parameters HFSZLIM and SFSZLIM.

- **Range** – 2048 – 65536
- **Default** – 65536

When to Change the Default

You do not need to change the value of this parameter in the *mtune* file. You may need to increase the ULIMIT in the */etc/default login* file and/or a particular user's *.profile* file.

VER

Description

The VER parameter specifies the version of the operating system release. It is defined in the */etc/conf/pack.d/kernel/space.c* file, not the *mtune* file.

- **Range** – N/A
- **Default** – 3.0

When to Change the Default

The value of this parameter is set during system software installation; you do not need to change it.

XSDSEGS

Description

The XSDSEGS parameter specifies the number of shared data segments in the system when using XENIX compatibility.

- **Range** – 0 – 100
- **Default** – 25

When to Change the Default

Increase the value of this parameter to permit more shared data segments on the system.

XSDSLOTS

Description

The maximum number of shared data segment attachments allowed in the system is XSDSEGS * XSDSLOTS.

- **Range** – 0 – 5
- **Default** – 3

When to Change the Default

Increase the value of this parameter to permit more shared data segment attachments.

XSEMMAX

Description

The XSEMMAX parameter specifies the maximum number of XENIX semaphores permitted on the system at one time.

- **Range** – 0 – 60
- **Default** – 60

When to Change the Default

Increase the value of this parameter if you need to permit more XENIX semaphores.

Chapter 5

Tuning Basics

This chapter gives you an introduction to tuning. It explains how to determine the need for tuning, the systems resources, and the most common bottlenecks that occur in computer systems.

This chapter contains the following information:

Reasons for Tuning 5–2

Basic Steps 5–3

Knowing When It's Time To Tune 5–6

System Resources 5–8

Potential Bottlenecks 5–11

Basic Tuning Procedure 5–14

Reasons for Tuning

Tuning a system, which is also referred to as tuning the kernel, means to optimize the use of system resources to avoid bottlenecks and poor performance. Tuning is the art of balancing the available resources of the system between all programs running on the system, including the operating system itself. Some resources, like memory and processors, are very expensive and at the same time very valuable system resources. Tuning also implies an increase in system performance at the lowest possible cost.

Tuning a system most often refers to changing the tuning (kernel) parameters. The default values for the system tuning parameters are based on the average system and work well for most environments. It may be necessary to adjust these values depending on the hardware configuration, the software applications running, or the need to optimize the use of one resource at the expense of another. Your system hardware and software configuration may change over time, making it necessary to reevaluate the current tuning configuration to improve the system performance.

Some applications require a very particular tuning environment. In this case, refer to the application–specific documentation for specific tuning information.

Basic Steps

Perform the following basic tasks to improve the system performance without spending a lot of time. These tasks do not require extensive testing:

- Removing Unused Software Packages
- Eliminating Misuse of the System
- Reorganizing the Free Block List

Removing Unused Software Packages

One of the first steps in tuning your system is to examine your software configuration for any software package that is not being used or not necessary for your system configuration. Since memory is a very desirable resource in every system, you should always make sure that the kernel does not take up additional space in memory because of unnecessary drivers and modules linked into it. Additionally, many packages have daemons running in the background using up system time unnecessarily. Use the following procedure to remove packages:

Step 1: To obtain a list of installed software packages, enter the command:

```
# pkginfo | pg
```

Step 2: Once you have identified the packages which are not necessary for your system configuration, remove them, using the following command:

```
# pkgrm package_name
```

Step 3: After you remove a software package, reboot your system. Use the following command to reboot:

```
# cd /
# shutdown -g0 -i6 -y
```

Rebooting is necessary because any drivers that have been installed with a package are linked into the kernel and increase the size of the kernel in memory. Rebooting removes these drivers.

Eliminating Misuse of the System

The system administrator should make sure that all jobs are scheduled appropriately:

- Scheduled jobs when the system is less busy.
- Carefully check and watch jobs executed through **cron**. Check to see if they run at the correct time. Whenever possible, they should be executed when the system load is reduced.
- Examine the **cron** log files for any runaway jobs which could use up either disk space or system time.
- Examine the process status table using the **ps** command to check for any processes using up system time unnecessarily. Then use the **kill** command to stop them.

Reorganizing the Free Block List

For the user file system types (s5, ufs),the free block list can be reorganized so that new files are created with their blocks more contiguously arranged. Reorganizing the free block list does not eliminate the fragmentation of existing files.

The free block list is initialized at file system creation time. When several files are created at the same time, they compete for the same free blocks. The files may be made up of blocks that are out of sequence. When files are removed, their different sequences of blocks are put back into the free block list. New files are the made up of these out of sequence blocks. The free list becomes scattered about the disk with the ongoing activity of allocating and freeing blocks.

To reorganize the free block list for a file system, use the **fsck** utility. The following example shows how to reorganize the free block list in the file system */home*:

```
# fsck -F s5 -s /home
```

NOTE: If you performed **fsck** as part of a normal start–of–the–day operation or as a regular maintenance procedure, the free block list is automatically reorganized in **Phase 6: Salvage Free List** for an s5 file system and in **Phase 5: Check Cylinder Groups** for an ufs file system.

Knowing When It's Time To Tune

After performing the steps in the "Basic Steps" section, you should reevaluate your system for reallocation of system resources when one of the following situations is present:

- The system response time is very slow:
  - The system seems to hang after every command typed in.
  - The system seems to hang for only some of the commands typed in.
- Additional hardware such as memory or peripherals have been installed.
- Additional software has been installed.
- Error messages are displayed repeatedly indicating file system table entries such as the file system inode table are overflowed. The error messages must be displayed consistently and over a longer period of time (days) before it becomes necessary to change kernel parameters. Refer to the *System Message Manual* for more information about how to react to such error messages.
- Results of monitoring the system activity utilities such as **ps**, **sar**, process accounting and system profiling indicate inappropriate system utilization. Refer to the "Monitoring System Activity" chapter for more information.

NOTE: For reliable results of performance monitoring tools, it is best to accumulate data over long periods of time (days) when system activity represents normal usage.

System Resources

Major System Resources

Since UNIX SVR4 is a multiuser, multitasking and interactive operating system, system resources have to be shared among multiple processes or tasks from different users. The main system resources are:

- Memory
- Processor
- I/O devices
- Disk files
- User services

Besides these major resources, every tunable kernel parameter represents a system resource.

Major Competitors for Memory

When tuning a system, the use of memory is always a concern. Memory must be balanced among the following major competitors:

- Kernel
- File I/O
- User processes
- Shared Memory (Note that many applications use shared memory including databases.)

Major System Components

The NCR UNIX SVR4 operating system can be divided into three major components:

- Process Management Subsystem
- File Management Subsystem
- I/O Management Subsystem

Tuning often means to optimize the performance of one subsystem at the expense of another. Before tuning a system, you must understand the individual operations of each of the subsystems as well as of the interaction between the subsystems.

Process Management Subsystem

The process management subsystem acts as the police officer who makes sure that every process in the system gets a fair share of the system resources. The CPU must be scheduled in an orderly fashion among the different processes of the operating system, system peripherals, user applications and other subsystems. The creation and termination of processes also falls under the responsibility of the process management subsystem.

For a command or program to be executed, it must be loaded into memory. The loaded program then becomes a process. Programs are brought in to memory one page at a time as the result of a page fault. In a typical UNIX environment, there are many processes competing for a processor. Since the amount of physical memory is limited, the processes often ask for more memory than is available. In case of competition for the CPU among processes, the process management subsystem must page or swap processes out to disk. The swap area can be thought of as virtual memory.

File Management Subsystem

The file management subsystem handles the accessing and updating of files. It is responsible for recognizing the different file system types and file types, enforcing file access permissions, locating file in main memory and locating file on disk devices.

I/O Management Subsystem

The I/O management subsystem is responsible for the transfer of input and output data from the user or the operating system to a peripheral device. To do the data transfers, the I/O management subsystem uses different buffering methods. For example, data can be kept in cache buffers which are kept in memory to access the same data faster than retrieving it from disk in case it is needed again.

Devices can be physical pieces of hardware such as a disk drive, floppy drive, printer or terminal, or it can be a pseudo–device or virtual device such as */dev/null* or */dev/kmem*. The device driver routines specify the type of transfer. Data can also be transferred with no buffering at all.

Potential Bottlenecks

The following section explains how memory resources are allocated and used in NCR UNIX SVR4.

Major Tuning Issue

The most difficult aspect of tuning is to ensure that the total sum of memory used by the kernel, user processes, file I/O and shared memory does not greatly exceed the total amount of physical memory. These major competitors divide up memory in the following ways:

- Kernel size can grow dynamically.
- Memory use by file I/O is limited (by SEGMAPSZ).
- User processes should have sufficient available memory.
- Shared memory is a fixed value.

The difficulty of memory tuning lies in calculating the size of the kernel in memory because the kernel can grow dynamically. Once this value is known, you can determine whether or not there is enough memory left for user processes. If user processes take up too much memory, you can add more memory or cut down on the number of users. It is very important to allocate enough memory to user processes so they will be kept in memory and not paged out or swapped out entirely to disk.

Balancing Memory Between Subsystems

Central to most tuning is finding the appropriate balance of memory usage between the process management subsystem and I/O management subsystem. Memory that is allocated for shared memory or programs is not available for file I/O; the reverse is also true. By using system tuning parameters, controls and limits can be established that ensure good system performance regardless of the peak load on any subsystem as long as the average load is within the capability of the system.

Recognizing a Memory Starved Process Subsystem

You can recognize a memory starved process management subsystem when the system appears to be locked up since the processes serving the terminals will be swapped out or asleep waiting on resources for long periods. Possible actions include the following:

- Reduce the amount of memory pages allocated for buffers. The page buffer area for disks, also called "caches," is an area where data blocks from the file systems are temporarily stored in pages. Before any data is read, these cache buffer pages are read. If the block is already in the page buffer cache, then the actual I/O from disk does not have to occur. If you allocate too many memory pages for page buffer caches, there will be less memory available for processes. Buffer pages for file I/O are allocated dynamically but the maximum number of buffer pages is determined by the kernel parameter SEGMAPSIZE.

- Increase the swap space. Adding additional swap space on a non–root disk also decreases a possible disk bottleneck. That way, processes can be swapped out to different swap areas on different disks simultaneously.

Recognizing a Memory Starved I/O Subsystem

A memory starved I/O management subsystem will result in certain functions appearing to hang or be extremely slow while other functions perform normally.

Increase the amount of memory allocated for page buffers. The page buffer area for disks, also called "caches," is an area where data blocks from the file systems are temporarily stored in pages. Before any data is read, these cache buffers are read. If the block is already in the buffer cache, then the actual I/O from disk does not have to occur. If you allocate too little memory space for page buffer caches, it will take more time to read data blocks from disk to memory than from page buffer cache to memory.

NOTE: Increasing the amount of memory for page buffers also takes away the memory that would be available for executing processes. If there is not enough memory for processes, the system will have to swap out processes which will also decrease the performance.

Recognizing a Shortage of Physical Memory

If you balanced the memory allocation between the process management and I/O management subsystems, and the system is still running unacceptably slow, the following actions can be taken:

- Redistribute the process load of the system more evenly so that the peak period is less busy.
- Redesign the applications to reduce their memory size, for example by using shared libraries and/or shared text.
- Add more physical memory.

Basic Tuning Procedure

In order to determine the success or failure of changing the system configuration parameters, you should always use the following procedure:

- Analyze the current performance using the utilities as described in the "Monitoring System Activity" chapter.
- Modify only ONE parameter at a time. This way you can see whether or not increasing or decreasing the parameter 's value improves the system performance.
- Analyze the current performance again with the changed parameter in effect.
- Compare the results of your performance analysis before and after the changing of the parameter.
- Repeat this procedure as necessary.

Chapter 6

Memory Utilization

Because processors can only access instructions and data in main memory, memory is one of the most important resources of any computer system. This chapter contains information about the following:

Memory Management 6–2

Demand Paging 6–3

Swapping 6–7

Flushing 6–10

Memory Management

Memory management is a central issue in a multiuser environment where the operating system itself and the user processes may exceed the available memory. The common solution in this situation is to use a part of the disk as secondary memory, or virtual memory. The size of virtual memory can be configured depending on the number of users and processes running on the system.

When executing a process, the CPU can only access the processes code in main memory. NCR UNIX SVR4 implements a scheme of swapping and demand paging to transfer processes in and out of main memory.

Demand Paging

A page is a the smallest amount of memory that a process can allocate. The size of a page is machine dependent. For the NCR System 3000, the page size is always 4KB. Demand paging is a system where the complete process does not need to be in memory to execute, only those pages that are referenced ("demanded") need to be in memory.

Demand Paging Process

With demand paging, usually pages, not entire processes, are swapped out to secondary memory or the swap device. When the system becomes short of main memory pages, the pageout daemon or the page stealer process becomes active. The pageout daemon steals "old" pages from processes and returns them to the free list of pages until a desired level of free pages is reached. An "old" page is a page that has not been read or written to recently. A page will have to be written to disk before it can be freed in case it has been modified.

When another page of the process from virtual memory is needed that is not already there, the UNIX operating system also uses the concept of page faults. A page fault occurs when the required page is not in main memory and the page needs to be read from virtual memory into physical memory.

You should be aware of the following aspects of demand paging:

- When a process is started, the first two pages are read in and all other pages of the process are read into memory as a result of page faults.

NOTE: This happens for every process regardless of how much memory is left. Even if there is enough remaining memory for all the pages of a process, all pages except the first two will be read into memory as a result of a page fault.

- When a process references a page that is not in memory, the process will be put to sleep until that page is moved into main memory.
- When a child process is created, a duplicate of the parents' memory is not made until either the parent or the child attempts a write to that memory page.
- When an application requests a memory allocation, the allocation succeeds but the memory page is not designated until it is referenced by the application. This means that until a page is referenced it does not use the physical memory space but instead resides in the logical address space.

Advantages and Disadvantages

Demand paging has two main advantages:

- It does not waste main memory since only pages of the program which are actually used are copied into main memory.
- It allows large programs to run and it allows for quicker program startup time since only the accessed pages need to be transferred into main memory instead of the entire program.

However, when page faults occur, the system must spend time to locate the additional page and bring it into memory. This adds extra overhead.

Parameters

There are three tuning parameters set up for controlling paging so that the system can absorb the shock of many processes starting at once or to ensure sane operation during peak load periods.

These tuning parameters include the following:

- LOTSFREE
- DESFREE
- MINFREE

LOTSFREE

The system will use all memory up to memory – LOTSFREE for pages. Even if the pages are old (not recently accessed) or dirty (not recently written to disk), they will not be paged out as long as memory –LOTSFREE is not exceeded. Therefore, the **sar –r** will often show freemem as close to memory –LOTSFREE all the time.

If, while the paging daemon is trying to read in a referenced page it finds fewer than LOTSFREE pages of free memory, then the pageout daemon will be called to run four times a second and will swap pages to disk if necessary. If the pageout daemon finds LOTSFREE pages, it will go to sleep. If LOTSFREE has a value of 0, the value is assigned dynamically and is by default set to 1/8th of the pages of memory or 256 pages of size 4K, whichever is less.

DESFREE

This parameter sets the desired level of free pages in the system. If the actions of pageout as a result of exceeding memory – LOTSFREE do not keep up with the demand for memory and memory – DESFREE is exceeded, then the pageout daemon is called to run 100 times a second to swap pages to disk until free memory is greater than LOTSFREE. By default, DESFREE has a value of 0 in the *mtune* parameter configuration file; the value is assigned dynamically to be equal to one half of LOTSFREE, which also equals to 1/16th of the pages of memory.

MINFREE

This parameter specifies the minimum of pages needed to be free in a system. If the pageout actions taken as a result of exceeding memory – DESFREE do not keep up with the demand for memory, then user requests for memory result in suspending user processes until pageout can return free memory to a value greater than LOTSFREE. By default, MINFREE is given a value dynamically; that is, it has a value of 0 in *mtune* file, which causes MINFREE to be a value equal to one half of DESFREE or 1/32 of the pages of memory.

A page will be kept from being swapped out to disk if the process to which the page belongs has the "sticky bit" set or has a real time priority.

Swapping

Swapping of processes occurs only when the page daemon can no longer supply the required free pages for a process. The swapper or the **sched** process is called to copy complete processes from main memory out to disk. This continues until the amount of free memory becomes greater than LOTSFREE. A swapped out process stays on disk until it is ready to run again. Once it is swapped back into main memory, the process is executed according to its priority.

Swapping is a much more serious condition than paging. Unlike with paging, swapping will ignore the "sticky bit" and the Real Time priority of a process.

Parameters

The tuning parameters for swapping include the following:

- GPGSLO
- MINAKMEM
- MINARMEM
- MINASMEM

GPGSLO

This is the main parameter for swapping. Process swapping occurs when the amount of free memory is less than the value of GPGSLO. When the number of free pages are below this value, processes can no longer just be suspended as with a value of free pages less than MINFREE; entire processes must be swapped out from main memory to disk. The default for GPGSLO is 25 pages of size 4K, a value less than MINFREE. This relationship between GPGSLO and MINFREE should always be maintained.

Although occasional swapping is acceptable, consistently swapping processes to disk usually causes a performance degradation that cannot be tolerated. Typically swapping will occur at start of day or when many processes are started within a short period (database startup or comm package startup). It is not unusual for these periods to last 10 to 20 minutes during which users will experience slow downs.

NOTE: Swapping refers to swapping entire processes in and out of main memory while swapping pages refers to the process of paging.

The MINAKMEM, MINARMEM, MINASMEM parameters establish minimum values to avoid deadlock conditions of memory.

MINAKMEM

This parameter defines the minimum available kernel memory required to avoid a deadlock. You do not need to change this value.

MINARMEM

This parameter establishes a minimum value for real memory reserved for user processes to avoid deadlock. If more processes are added to the system and the processes execute very slowly, increasing this value could help increase the performance.

MINASMEM

This parameter establishes a minimum value for memory and swap space used for the operating system itself to avoid deadlock. If more processes are added to the system and an error message is displayed on the console that this value is exceeded, you should increase the value of MINASMEM.

Setting Up Swap Space

The amount of swap space is dependent on the memory size of your system. Refer to the procedure "Configuring Swap Space" in the "Configuring Disks" chapter for recommendations and guidelines about the amount of swap space for a specific memory size. Always remember that the values given are only recommendations, not hard rules.

The only hard rule for swap is that there must be enough swap space to allow all processes currently running in the system to be swapped out.

Flushing

Flushing ensures that available memory that has been written by file I/O will be flushed to disk at least once in every minute. This is accomplished by the **fsflush** routine. The FDFLUSHR parameter controls the rate at which the memory is flushed to disk.

The default value is 1 second. By default, FDFLUSHR examines one sixtieth of memory every second and writes all dirty pages to disk. This process ensures there are no dirty pages in memory which are older than 60 seconds.

The NAUTOUP parameter controls the number of seconds after which a dirty page should be written to disk. The default for this value is 60 seconds. Increasing this value is not recommended since this only causes more I/O to disk and therefore result in performance loss. By decreasing this value, more data is lost in case of a system crash. The loss of data integrity does not outweigh the gain of performance by a decreased value for NAUTOUP.

Chapter 7

Disk Utilization

Since the operating system and the user applications and programs frequently reference data residing on disk, using the disk efficiently and optimizing the physical organization of the data on the disk can improve system performance. This chapter contains the following information:

Factors Affecting Disk Performance 7–2

I/O Balancing 7–3

Buffer Utilization 7–4

Monitoring and Controlling Disk Usage 7–7

Directory Reorganization 7–14

Eliminating File System Fragmentation 7–16

Factors Affecting Disk Performance

Many factors, including the following, affect disk performance:

- I/O balancing
- Buffer utilization
- Application designs
- Directory organization
- File system organization
- File system free space

The remaining sections in this chapter address these aspects of maintaining system performance.

I/O Balancing

Hardware Configuration of Disks

In a multiprocessor environment there are typically multiple disk drives. The more system buses and controllers that are available, the better the disks' I/O loads can be distributed. Performance can be gained when I/O performed simultaneously on multiple disks can be distributed among different hardware channels such as buses and controllers. Each disk controller has two buses leading to different backplanes. Each backplane can hold four full–height disks.

Balancing I/O Between Available Disks

The system's data should be distributed across the disks so that each drive performs an equal share of work and disk "hot spots" can be eliminated. If you have two I/O intensive applications, put each application's data on separate disk drives. Use the **sar –d** utility to identify "hot spots."

Separating Data into Different File Systems

Each disk can be divided into different slices which can contain either file systems or raw data. For very large disks (disks of size 670 MB and larger) it is recommended to separate different types of data out to different slices.

NOTE: Having the data logically separated on different slices or file systems will provide a method to take a file system offline temporarily by unmounting it without making the other data inaccessible. Also if one file system becomes corrupt, any other data on the system will still be accessible.

Buffer Utilization

The following parameters are used in tuning and maintaining buffers:

- BUFHWM
- NBUF
- SEGMAPSZ

Buffer Headers

Buffer headers contain all the information that is needed to do a physical disk I/O. For example, they contain information such as the device and sector where the data is read from and the amount of data that has to be read. The actual file data is stored in pages of virtual memory, and the amount of virtual memory that can be used for this purpose is controlled by SEGMAPSZ.

Reserved for Operating System (kernel): 4 MB

SEGMAPSZ ((memory –4) x 0.8: 48 MB

BUFHWM ((memory –4) x 0.25): 15 MB

BUFHWM is the parameter that specifies the maximum amount of kilobytes of memory that can be used by block I/O . The default value of BUFHWM is 0. This value indicates that BUFHWM is a dynamically allocated system parameter. The kernel automatically sets the value of BUFHWM to 25% of available memory. The available memory is calculated by subtracting the size of the kernel itself from the physical memory. The size of the kernel in a UNIX SVR4 system is approximately 4MB.

General Guidelines for Buffer Tuning

Although there are no hard rules for buffer tuning, you must consider the following:

- Decreasing the buffers will free up more memory for use by the system at the expense of I/O performance.

- Increasing the buffers will decrease the amount of available memory by the system, but will increase the I/O performance.

It is best to opt for more memory performance than disk performance because a system running out of memory will also use disk space for swapping and paging. Avoiding memory shortages will also avoid extra disk I/O.

The following guidelines apply to buffer tuning:

- Keep the number of buffer headers small (use the default of 100) because with NBUF only the number of buffer headers that are allocated at one time is limited. NBUF does not restrict the number of available buffer headers.

- Use the SEGMAPSZ parameter to arbitrate the competition for memory between the file subsystem and the process subsystem, you can . The segment map is a bit map used by the kernel to keep track of the 4K pages available for file I/O, 1 bit per 4K page. By reducing the size of the bit map, the file I/O subsystem is restricted from trying to use all available memory.

  This parameter normally does not need to be changed. However, if you do change it, you should reduce SEGMAPSZ by the number of 4K pages of shared memory that will be in use by applications.

- Disk performance may be more important on a system where users access very large files, therefore using the file systems heavily. However, if the files are much larger than the buffer area, and you are reading the files only once, setting aside a large buffer area is not going to improve performance. It could decrease it, since you will have less memory available.

- Disk performance may be less important on a system with applications performing a lot of computation and little I/O.

- Systems running X–Windows extensively should have plenty of memory so that the page buffer cache should be reasonably small.

Monitoring and Controlling Disk Usage

Keeping disk space uncluttered involves the following:

- Monitoring the amount of disk space used
- Monitoring files and directories that grow
- Identifying and removing inactive files
- Identifying large space users

Why/When

You should monitor disk use for many reasons, including the following:

- If unwatched, the amount of disk space used increases until the allocated space is used up.
- When the allocated space is used up, processes run very slowly or not at all. The system then spends its time sending messages to the system console about being out of file space.
- Users forget about files they no longer use, and the files continue to take up space.
- Some files grow because of normal system use. Controlling their size is an administrative responsibility.
- Some directories, notably */tmp* and */var/tmp*, accumulate files during the day and are cleared when your system goes from single user to multi-user mode. When the system first comes up, these directories need to have enough free blocks to accommodate system needs until shutdown.

Monitoring Amount of Disk Space Used

Watch disk usage during the day to see how close to capacity your system is running.

You can monitor disk space usage at any time with the **df**(1) command. For example, if you enter the command:

```
# df -t
```

a display similar to the following appears on your screen:

```
/          (/dev/root          ):    75024 blocks    25694 files
                             total:  152066 blocks    32256 files
/proc      (/proc              ):        0 blocks      188 files
                             total:        0 blocks    202 files
/dev/fd    (/dev/fd            ):        0 blocks        0 files
                             total:        0 blocks     26 files
/stand     (/dev/dsk/c0t0d0sa ):     3991 blocks     87 files
                             total:    10240 blocks     96 files
```

The –t option displays the number of free blocks and files (line 1) followed by the total number of allocated blocks and files (line 2). Unless you specify a particular file system, information about each mounted file system is displayed.

If the number of free blocks and files is very low and you need more free space, you may need to add a new disk or reslice the current disk(s) to reallocate space.

Monitoring Files and Directories that Grow

A system that is used daily has several files and directories that grow through normal use. Figure 7–1 shows some examples:

File	Use
/var/adm/wtmp	Contains login information. Grows with terminal line difficulties. Use **acctcon**(1M) to determine offending lines.
/var/cron/log	Log of commands **cron** uses. Contains error messages from programs used by */usr/lib/crontab*.
/usr/adm/sulog	Log of who uses the **su** command.
/var/spool	Spooling directory for line printers, **uucp**(1C), etc. Need to compact subdirectories of spool.
/tmp	Temporary directory to hold small miscellaneous files. It is cleared each time your system goes from single user to multi–user. If your system rarely goes single user, check this directory.
/var/tmp	Temporary directory to hold large miscellaneous files. It is cleared each time your system goes from single user to multi–user. If your system rarely goes single user, check this directory.

Figure 7–1: Files and Directories That Grow

Monitoring the growth of these files and directories involves two tasks:

- Controlling Growth
- Maintaining Start–of–day Counts

Controlling Growth

Step 1: Check files that grow by using the **du** command periodically and comparing the output.

How often you should check growing files depends on how active your system is and how critical the disk space problem is.

Step 2: Use a combination of **tail**(1) and **mv**(1) to keep files to a reasonable size. For example:

```
# tail -50 /var/adm/sulog > /var/tmp/sulog

# mv /var/tmp/sulog /var/adm/sulog
```

This sequence puts the last 50 lines of */var/adm/sulog* into a temporary file and moves the temporary file to */var/adm/sulog*, thereby leaving the 50 most recent entries in the file.

Maintaining Start–of–day Counts

Step 1: Use the **df**(1M) command to examine the free space at start–of–day.

```
# df -t
```

Step 2: Examine the output of the **df** command to ensure that the following file systems or directories have sufficient free space to accommodate files that are created during the day:

- */tmp*
- */var/tmp*
- */home*
- Other user file systems

Step 3: If the file systems listed above do not have enough free space, you may have to increase their size or reslice the disk so that you can make them larger.

Identifying and Removing Inactive Files

You should locate and remove files that have not been used recently.

Step 1: Decide on a policy for removing files that have not been used for a period of time.

In deciding what files to remove, consider the following:

- How long should a file remain unused before it becomes a candidate for removal?
- Should users be warned that old files are about to be purged?
- Should the files be removed permanently or archived?

Step 2: Use the **find**(1) command to locate files that have not been used recently.

The following example uses the **find** command to place the names of files that have not been modified in 60 days in the */tmp/deadfiles* file.

```
# find /home -type f -mtime +60 -print \
> /tmp/deadfiles &
```

The "&" causes the command to run in background, a sensible precaution if you expect a substantial amount of output.

Step 3: Remove files, based on the policy you developed in Step 1.

Identifying Large Space Users

You should limit the amount of space users, files and directories are allowed to use.

Step 1: Decide on a policy for limiting disk space for individual users.

Again, the most important questions are not what commands to use but what limits to set. Policy questions include:

- On your system, what constitutes a reasonable amount of disk space for a user to need?
- If a user exceeds the normal amount by 25%, for example, is it possible the user's job requires extraordinary amounts of disk space?
- Is your system, as a whole, running short of space? Do your existing limits need to be reviewed?

Step 2: Use the **du**(1) command to identify users that are increasing disk usage rapidly.

The **du** command produces a summary of block counts (in 512 byte blocks) for files or directories named on the command line. Use **du** after working hours and keep the output in a file for later study.

The following entry displays the block count for all directories in the */home* file system.

```
# du /home
```

Step 3: Use the **find**(1) command to locate specific files that exceed a given size limit.

The following example displays the names of all files (and directories) in the */home* file system that are larger than 10 (512–byte) blocks.

```
# find /home -size +10 -print
```

Step 4: Set limits for files, directories, or users that exceed a reasonable amount of disk space.

Directory Reorganization

Identifying Large Directories

Very large directories are very inefficient and can affect performance. Two directories in particular, */var/mail* and */var/spool/uucp* tend to get very large and should be monitored periodically. If a directory becomes bigger than the number of blocks it can directly address, then the directory searches can cause performance problems. For a s5 file system there are 10 direct addressable blocks. For an ufs file system there are 12 direct addressable blocks. Use the **find** command to search for large directories.

The following example shows how to find all directories of a s5 file system that are larger than 10 logical blocks of 2K:

```
# find / -type d -size +40 -print
```

The following example shows how to find all directories of a ufs file system that are larger than 12 logical blocks of 4K :

```
# find / -type d -size +96 -print
```

Keep in mind that the **find** command thinks in terms of 512–byte blocks.

Check these large directories for unnecessary data. However, deleting files from large directories does not decrease a directory's size. Use the next procedure to make large directories more compact in case they had many files deleted from them.

Shrinking Directories

Removed files leave an empty slot in the directory. For example, if you have a directory with 100 files in it and you remove the first 99 files, the directory still contains 99 empty slots preceding the active slot. Unless a directory is reorganized on the disk level, it will retain its largest size.

The following procedure can be used to remove unused entries from a directory:

Step 1: Rename the directory you want to shrink, for example:

```
# mv /var/mail /var/old.mail
```

Step 2: Recreate the directory that needs to be shrunk, for example:

```
# mkdir /var/mail
```

Step 3: Change your current directory to the renamed directory you created in step 1 and copy the files back. For example:

```
# cd /var/old.mail
# find . -print | cpio -pdmuv /var/mail
```

Step 4: Remove the original directory. For example:

```
# cd ..
# rm -r /var/old.mail
```

You can also use this procedure when you deleted a large number of files from a directory without recreating or copying the same amount of files in that directory.

Eliminating File System Fragmentation

Why/When

If an **s5** file system becomes so fragmented that it affects system performance, use the procedures in this section to eliminate the fragmentation. The procedures in this section are most effective with **s5** file systems; however, they can be used with **ufs** file systems.

The **ufs** file system is designed to greatly reduce fragmentation, compared to the **s5** file system. While you can use these procedures on ufs file systems, the benefits are not as good as on **s5** file systems.

If you have the Journaling File System (**vxfs**) installed on your system, you can reorganize the file system using the command **fsadm**(1M). For more information refer to the *Journaling File System Administrator Guide*.

Warning

Do NOT use these procedures for root file systems; they apply to non-root file systems only.

Methods to Eliminate Fragmentation

There are two methods for eliminating file system fragmentation:

- Using **dcopy**(1M): Optimizes a file system when copying it to another disk partition of equal size. Use this method for s5 file systems only.
- Reorganizing a file system: If a specific file system is used frequently, you may back it up, remove it, can re-create it so that each of its files are created as efficiently as possible. You may use this method for s5 or ufs file systems.

Using dcopy

The **dcopy**(1M) command optimizes a file system and copies it to another file system of the same size (which usually resides on another disk). The optimizations include the following:

- Moving subdirectories to the beginning of directories (for faster pathname searches)
- Removing unused entries in directories
- Spacing the blocks in all existing files to match the software interleave factors
- Spacing the free block list to match the software interleave factors

Using the **dcopy** command provides an average of 10 to 15 percent improvement in overall I/O throughput. The amount of improvement depends on the amount of disorganization that exists before the procedure.

NOTE: The use of **dcopy** is fairly restrictive; the copy must be made to another file system of the same size. You must have another disk partition of the same size, which is unused.

Step 1: Enter single user mode an unmount the file system to be copied.

Step 2: Run **fsck**(1M) on the file system to assure correctness.

Step 3: Copy the file system to a spare disk drive using **volcopy**(1).

Step 4: Copy the file system back to the original drive using **dcopy**(1M).

Step 5: Run **fsck**(1M) on the file system to assure correctness.

Step 6: Enter multiuser mode.

Reorganizing a File System

If the use of the **dcopy**(1M) command is not possible because you do not have an extra free disk or partition, or your file system is not an s5 file system, you may use the following procedure to reorganize the file system. These procedures perform the same optimizations as the **dcopy**(1M) command EXCEPT that subdirectories are NOT moved to the beginning of directories.

Step 1: Enter single user mode.

Step 2: Back up the desired file system using **cpio**(1).

Step 3: Unmount the file system.

Step 4: Recreate the file system using **mkfs**(1M).

Step 5: Restore the file system.

Step 6: Enter multiuser mode.

Chapter 8 CPU Utilization

This chapter describes the following considerations for balancing the system load and utilizing the CPU(s) as effectively as possible:

Workload and CPU Time 8–2

Efficient PATH Variables 8–3

Excessive Pathname Searches 8–5

Workload and CPU Time

Overloading the CPU(s) (processors) can reduce system performance. To achieve the maximum performance, the process load should be evenly distributed over time (as much as possible). The peak load period of usage should not be significantly higher than normal system use. The speed of a CPU cannot be changed. Therefore, the load on the CPU(s) or the amount of work the CPU(s) are requested to do must be changed.

If your system has more than one CPU (processor), the processes will automatically be distributed equally among the processors unless you bind processes to a specific processor using the **pbind** command. Only applications that are not multithreaded should be bound to a specific processor.

Use the following guidelines reduce the work load during peak time:

- The less important processes or jobs should be rescheduled to run during a time of lower load.
- Unnecessary processes should be eliminated from the system or restricted during peak time.
- Schedule jobs when the system is not so busy.
- Check the efficiency of user–defined features, such as *.profile* and the PATH variable.

Efficient PATH Variables

The PATH variable defines a list of colon separated directories that the shell searches for any commands you enter. If the command cannot be located in one of those directories, the shell prints out the error message "command: not found." If your system runs out of processor and disk resources, changes on your PATH variable could help improve the performance.

Use the following guidelines when you are setting up an efficient PATH variable:

- The directory where most commands are retrieved from should be the first one in the PATH variable since the PATH variable is read left to right.
- The PATH variable should contain as few directories as possible.
- The PATH variable should never list a directory twice.
- Long directory names should be avoided. If they cannot be avoided, place them at the end of the PATH variable, so that they get searched last.

The file */etc/default/login* contains the default PATH variable. The PATH variable in the */etc/default/login* file lists the directories with the most common commands. To add other directories to the PATH variable, add a line to the */etc/profile*, for example:

```
PATH=$PATH:/usr/bin/X11:
```

NOTE: The : at the end of the PATH variable causes the current directory to be searched last. If your PATH variable starts with a

:, it can be a security risk. It is a potential security risk because a command could be placed in the current directory with the same name as a system command. This "Trojan horse" command could perform completely different actions from the real command. The user might not even be aware of the command performing something differently or additionally to the real command.

To change the PATH variable for a single user, set up the PATH variable in the *$HOME/.profile* file.

Excessive Pathname Searches

Performing a pathname search means locating the inode of a file. Before the system can read the inode into memory, it must first read the directory blocks that will lead to that inode. For full pathnames, the system must read through all the directory blocks before reading the inode into memory.

If the user application changes to the appropriate directory and uses a relative pathname, the system has to search only those directories below the current directory in the hierarchy. Therefore, the use of relative pathnames can decrease the amount of pathname searching that the system must perform which helps reduce the load during peak times.

Reduce the number of excessive pathname searches by doing the following:

- Avoid very large directories.
- Redesign the applications to sure relative pathnames instead of full pathnames when referencing files.

Glossary

Active System
A system with Basic Networking Utilities and the hardware required to establish communication links (i.e. Auto–Dial Modem).

Address
A number, label, or name that indicates the location of information in the computer's memory.

A.out
The default name of a freshly compiled object file. Historically a.out signifies assembler output.

Archive
(1) A collection of data gathered from several files into one file; (2) especially, such a collection gathered by **ar**(1) for use as a library.

Automatic calling unit
A hardware device used to dial stored telephone numbers. It allows the system to contact another system over phone lines without manual intervention.

Bad block
A section of a storage medium that cannot store data reliably.

Basic Networking Utilities (BNU)
A group of programs and files that permit copy capability between UNIX Systems. Sometimes called UUCP.

Block
The basic unit of buffering in the kernel. See indirect, logical, and physical blocks.

Block device
A device upon which a file system can be mounted; typically a permanent storage device such as a tape or disk drive, so called because data transfers to the device occur by blocks.

Boot
To start the operating system, so called because the kernel must bootstrap itself from secondary storage into an empty machine. No login or process persists across a boot.

Boot block
The first block of a file system that is reserved for a booting program.

Boot program
Loads the operating system into memory.

Buffer
(1) A staging area for input/output where arbitrary–length transactions are collected into convenient units for system operations. (2) To use buffers.

Buffer pool
A region of storage available to the file system for holding blocks. All input/output for block devices goes through the buffer pool so read and write operations may be independent of device blocks.

Cartridge tape
A storage medium that consists of a magnetic tape wound on spools housed in a plastic container.

Character device
A device upon which a file system cannot be mounted such as a terminal or the null device.

Child process
See **fork**.

Controller
A device that directs the transmission of data over the data links of a network.

Crash
If a hardware or software error condition develops that the system can not handle, it takes itself out of service, or crashes. Such conditions occur when the system can not allocate resources, manage processes, respond to requests for system functions, or when the electrical power is unstable.

Cron
A command which creates a daemon that invokes commands at specified dates and times.

Daemon
A background process, often perpetual, that performs a system–wide public function, e.g. **calendar**(1) and **cron**(8).

Device
(1) A special file such as a tape drive or the nulldevice; not a regular file or directory. (2) A physical input/output unit.

Directory hierarchy
The tree of all directories, in which each is reachable from the root via a chain of subdirectories.

Disk
A platter coated with magnetic material on which data can be stored.

Diskette, flex disk
A magnetic storage medium that is smaller and originally more flexible than a hard disk.

Drive
The hardware device that holds magnetic disks, diskettes, and tapes while they are in use.

Dump
A copy of the memory of the system, used for analyzing problems.

Executable file
(1) An object file that is ready to be copied into the address space of a process to run as the code of that process. (2) A file that has execute permission, either an executable file or a shell script.

File system
(1) A collection of files that can be mounted on a block special file; each file of a file system is accessible via some path from the root directory of the file system. (2) The collection of all files on a computer.

Filter
A program that reads from the standard input and writes on the standard output, so called because it can be used as a data–transformer in a pipeline.

Flush
To empty a buffer.

Fork
To split one process into two (the parent process and child process) with separate, but initially identical, text, data, and stack segments.

Free list
In a file system, the list of blocks that are not occupied by data.

Getty
One of a series of processes which may connect the user to the UNIX system. In NCR UNIX System V Release 4, the **ttymon** (and **sac**) processes are normally used instead of **getty**. If used, **getty** is invoked by **init**, and in turn invokes login.

Group
(1) A set of permissions alternative to owner permissions for access to a file. (2) A set of user IDs that may assume the privileges of a group. (3) The group ID of a file.

Group ID
An integer value, usually associated with one or more login names; as the user ID of a process becomes the owner of files created by the process, so the group ID of a process becomes the group of such files.

Hole
A gap in a file caused by seeking while writing. **read**(2) takes data in holes to be zero. A block in a hole occupies no space in its file system. Files with holes in them are known as sparse files.

I–list
The index to a file system listing all the inodes of the file system.

Indirect blocks
Data blocks that are not directly referenced by an inode. The inode has addresses that indirectly reference (by a cascade of pointers) an additional number of blocks to handle extremely large file sizes.

Init
A general process spawner that is invoked as the last step in the **boot** procedure; it regularly checks a table that defines what processes should run at what run level.

Inode
An element of a file system; an inode specifies all properties of a particular file and locates the file's contents, if any.

Inode number, I–number
The position of an inode in the I–list of a file system.

Interface programs
Shell scripts (for example, those furnished with the LP Spooler software) that interface between the user and some part of the overall system (for example, the printer).

Kernel
The UNIX system proper; resident code that implements the system calls.

Kernel address space
A portion of memory used for data and memory addressable only by the kernel.

Link
(1) To add an entry for an existing file to a directory; converse of unlink. (2) By extension, a directory entry.

Link count
The number of directory entries that pertain to an inode; a file ceases to exist when its link count becomes zero and it is not open.

Load device
Designates the physical device from which a program will be loaded into main memory.

Local system
Refers to the system on the near end of a communication link; normally, your system.

Log files
Contain records of transactions that occur on the system; software that spools, for example, generates various log files.

Logical block
A unit of data as it is handled by the software. The system handles data in logical blocks whose size is determined by the file system.

Memory image
A copy of of all the segments of a running or terminated program. The copy may exist in main storage, in the swap area, or in a file.

Mode, file mode
The permissions of a file; colloquially referred to by a 3–digit octal number, e.g. 'a 755 file'. See **chmod**(1).

Mount
To extend the directory hierarchy by associating the root of a file system with a directory entry in an already mounted file system. The converse is unmount, spelled **umount**.

Namelist
Same as symbol table.

Network
The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.

Networking
For computer systems, this means sending data from one system to another over some communications medium (coaxial cable, phone lines, etc.). Common networking services include file transfer, remote login, remote execution.

Node
A terminating point (system) on a network.

Node name
The name for the system(may be up to 255 characters); used as the official name of the machine in a network. The node name resides in the NODE parameter.

Null device
A device that always yields end–of–file on reading and discards all data on writing.

Object file
A file of machine language code and data; object files are produced from source programs by compilers and from other object files and libraries by the link editor. An object file that is ready to run is an executable file.

Operating system
The program for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, the file system, removing this burden from user programs.

Open file
(1) The destination for input or output obtained by opening a file or creating a pipe; a file descriptor. Open files are shared across forks and persist across executes. (2) Loosely, a file that has been opened. However an open file need not exist in a file system, and a file may be the destination of several open files simultaneously.

Other
A set of permissions regulating access to a file by processes with userID different from the owner and groupID different from the group of the file.

Owner
The userid of the process that created a file; the owner has distinctive permissions for a file.

Page
A fixed length, a 4096–byte portion of memory that has a virtual address, and that can be transferred between main and secondary storage.

Paging
The process by which programs are truncated into pages and transferred between main and secondary storage by the virtual handler (or paging daemon).

Parent process
See **fork**.

Partitions
Units of storage space on disk.

Passive system
A system that has Basic Networking Utilities but does not have the hardware required to establish communication links and so never initiates calls.

Permission
A right to access a file in a particular way including read, write, execute (or look up in, if a directory). Permissions are granted separately to owner, group, and others.

Permission bit
A permission, so called because each permission is encoded into one bit in an inode.

Physical block
A unit of data as it is actually stored and manipulated. The system handles data in 512–byte physical blocks.

Physical memory
See **memory**.

Pipe
A direct stream connection between processes, whereby data written on an open file in one process becomes available for reading in another.

Pipeline
A sequence of programs connected by pipes.

Polling
The interrogation of devices by the operating system to avoid contention, determine operation status, or ascertain readiness to send or receive data.

Ports
The point of physical connection between a peripheral device (such as a terminal or a printer) and the device controller, that is part of the computer hardware.

Process
A connected sequence of computation. A process is characterized by a core image with instruction location counter,current directory, a set of open files, control terminal, userid, and groupid.

Process id
An integer that identifies a process.

Process number
Same as process ID.

Profile
(1) An optional shell script, .profile, conventionally used by the shell upon logging in to establish the environment and other working conditions customary to a particular user. (2) To collect a histogram of values of the instruction location counter of a process.

Program
(1) An executable file. (2) A process. (3) All the usual meanings.

Queue
A line or list formed by items in a system waiting for service.

Raw device
A block device, read and write operations to which are not buffered, and are synchronized to natural records of the physical device.

Reboot
Same as boot.

Remote system
Refers to a system on the far end of a communication link; normally, a system that your system calls.

Retension
The process of rewinding the tape in a cartridge tape device to make sure it is at the correct tautness for accurate recording of data.

Root
(1) A distinguished directory that constitutes the origin of the directory hierarchy in a file system. (2) Specifically, the origin for the file system, with the conventional path name "/".
(3) The origin of the directory hierarchy in a file system.

Run level
A software configuration of the system which allows a particular group of processes to exist.

Schedule
To assign resources – main store and CPU time – to processes.

Scheduler
A permanent process, and associated kernel facilities that does swapping.

Search path
In the shell, a list of path names of directories that determines the meaning of a command; the command name is prefixed with members of the search path in turn until a path name of an executable file results; the search path is given by the shell variable PATH.

Section, sector
A 512–byte portion of a track which can be accessed by the magnetic disk heads in the course of a predetermined rotational displacement of the storage device.

Segment
A contiguous range of the address space of a process with consistent storage access capabilities; the four segments are (1) the text segment, occupied by executable code, (2) the data segment, occupied by static data that is specifically initialized, (3) the bss.segment, occupied by static data that is initialed by default to zero values, and (4) the stack segment, occupied by automatic data, see stack. Sometimes (2), (3), and (4) are collectively called data segments.

Semaphore
An IPC facility that allows two or more processes to be synchronized.

Set userid
A special permission for an executable file that causes a process executing it to have the access rights of the owner of the file. The owner's user ID becomes the effective user ID of the process, distinguished from the real user ID under which the process began.

Set userid bit
The associated permission bit.

Shared memory
An IPC facility that permits two or more processes to share the same data space.

Shell
(1) The program **sh**(1), that causes other programs to be executed on command; the shell is usually started on a user's behalf when the user logs in. (2) By analogy, any program started upon logging in.

Shell script
An executable file of commands taken as input to the shell.

Single user
A state of the operating system in which only one user is supported.

Source file
(1) The uncompiled version of a program. (2) Generally, the unprocessed version of a file.

Special file
An inode that designates a device, further categorized as either (1) a block special file describing a block device, or (2) a character special file describing a character device.

Spool
To collect and serialize output from multiple processes competing for a single output service.

Spooler
A **daemon** that spools.

Startup
Same as boot.

Super block
The second block in a file system, that describes the allocation of space in the file system.

Superuser
userid 0 with access to any file regardless of permissions. The superuser can perform certain privileged system calls, e.g. setting the clock.

Swap
To move the memory image of an executing program between main and secondary storage to make room for other processes.

Swap area
The part of secondary storage to which memory images are swapped. The swap area is disjointed from the file system.

Symbolic link
A file that contains the path name of another file or directory. References to the symbolic link become references to the named file or directory.

Symbol table
Information in an object file about the names of data and functions in that file; the symbol table and address relocation information are used by the link editor to compile object files and by debuggers.

System calls
(1) The set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated. (2) The system primitives invoked by user processes for system–dependent functions, such as I/O, process creation, etc.

System console
The directly connected terminal used for communication between the operator and the computer.

System name
The name (up to 255 characters) for the system; resides in the SYS parameter. This is the name given by the manufacturer, indicating the version of your operating system. It is NOT the node name, a name that uniquely identifies your system for communication with other systems.

Text file, ASCII file
A file, the bytes of which are understood to be in ASCII code.

Track
An addressable ring of sections on a disk or flex disk; each disk or flex has a predefined number of concentric tracks, which allows the disk head to properly access sections of data.

Tunable parameters
Variables used to set the sizes and thresholds of the various control structures of the operating system.

Tuning
(1) Modifying the tunable parameters to improve system performance. (2) The reconfiguration of the operating system to incorporate the modifications into an executable version of the system.

UUCP
A group of programs and files that permit UNIX–to–UNIX copy capability between UNIX Systems. If shown in the text with bold type(**uucp**), this is referring specifically to the **uucp**(1) program or login ID. Also, called Basic Networking Utilities (BNU).

UserID
An integer value, usually associated with a login name. The userID of a process becomes the owner of files created by the process and descendent (forked) processes.

Utility, utility program
A standard, generally useful, permanently available program.

Index

A

access
- Maintenance File System: *Vol. 1* 2–11
- to remote systems: *Vol. 2* 9–35

access the system: *Vol. 1* 1–2

activate, processor: *Vol. 1* 1–33

active partition, change: *Vol. 2* 1–24

add
- disk: *Vol. 2* 1–3
- SCSI controller: *Vol. 2* 2–27, 2–34
- swap file: *Vol. 2* 1–27
- uucp logins: *Vol. 2* 9–20

adjust
- printer port characteristics: *Vol. 2* 6–12
- terminfo database: *Vol. 2* 6–14

AFFIN_ON: *Vol. 3* 3–42, 4–2

AFFINDECAY: *Vol. 3* 3–42, 4–3

affinity scheduling parameters: *Vol. 3* 3–42

AGEINTERVAL: *Vol. 3* 3–32

AIOTIMEOUT: *Vol. 3* 3–40, 4–4

alert
- receive for printer fault: *Vol. 2* 5–8
- stop for printer fault: *Vol. 2* 5–9

alias database, sendmail: *Vol. 1* F–15

aliasing, sendmail: *Vol. 1* E–14

alignment pattern: *Vol. 2* 5–14

allow, access to printer: *Vol. 2* 4–46
alternate boot disk: *Vol. 1* 1–17
    boot from: *Vol. 1* 1–24
analyze
    activity data: *Vol. 3* 1–7
    specific process with timex: *Vol. 3* 1–47
    specific processes with time: *Vol. 3* 1–45
analyze system activity, with process accounting: *Vol. 3* 1–35
ARCHITECTURE: *Vol. 3* 3–47, 4–5
ARG_MAX: *Vol. 3* 3–29, 4–6
argument, sendmail: *Vol. 1* F–19
asynchronous I/O
    maximum outstanding calls parameter: *Vol. 3* 4–69
    timeout value parameter: *Vol. 3* 4–4

B

backing up, entire system: *Vol. 1* 6–19
backup
    file system and disk information: *Vol. 1* 6–16
    file systems: *Vol. 1* 6–4
    raw devices: *Vol. 1* 6–14
bad block, mark: *Vol. 2* 1–19
Basic Networking Utilities, (see BNU): *Vol. 2* 8–2
baud rate, printer: *Vol. 2* 4–51
BDFLUSHR: *Vol. 3* 3–29
bfs file system, check: *Vol. 1* 5–78
bfs file system type: *Vol. 1* A–23
bind, display information for processes: *Vol. 1* 1–35
bind a process: *Vol. 1* 1–35
block, mark bad: *Vol. 2* 1–19
BMAX: *Vol. 3* 3–44

BNU
administrative support files: *Vol. 2* 8–12
cleanup undeliverable jobs: *Vol. 2* 9–68
commands: *Vol. 2* 8–5
description of: *Vol. 2* 8–2
maintain: *Vol. 2* 9–64
making physical connection: *Vol. 2* 9–7
set up links: *Vol. 2* 9–5
BNU data base: *Vol. 2* 8–10
BNU link, verify: *Vol. 2* 9–63
BNU log file, check size of: *Vol. 2* 9–68, 9–70
BNU logs: *Vol. 2* 8–16
boot
from alternate disk: *Vol. 1* 1–17
from flex diskettes: *Vol. 1* 2–4
kernel: *Vol. 3* 3–15
boot flex diskettes
copy: *Vol. 1* 2–7
use: *Vol. 1* 2–2
BOOT program, recover: *Vol. 1* 3–29
buffer, number reserved for pageout demon: *Vol. 3* 4–106
buffer utilization: *Vol. 3* 7–4
BUFHWM: *Vol. 3* 3–29, 4–7
build, sendmail configuration file: *Vol. 1* F–39

C

call–out table, maximum entry parameter: *Vol. 3* 4–74
cartridge tape, retension: *Vol. 1* 6–26
cblocks, maximum parameter: *Vol. 3* 4–78
cdfs file system type: *Vol. 1* A–27

change
active partition: *Vol. 2* 1–24
keyboard mapping: *Vol. 1* D–4
line characteristics to 8 bit: *Vol. 1* D–2
menu entry: *Vol. 1* C–18
run levels: *Vol. 1* 1–5
screen mapping: *Vol. 1* D–6
character set: *Vol. 2* 5–24
list: *Vol. 2* 5–26
map: *Vol. 2* 5–26
name: *Vol. 2* 5–25
characteristics, adjust for printer port: *Vol. 2* 6–12
check
file system: *Vol. 1* 5–4
quotas: *Vol. 1* 4–21
size of BNU log files: *Vol. 2* 9–68, 9–70
ufs file systems: *Vol. 1* 5–38
check bfs file system: *Vol. 1* 5–78
cleanup
public area: *Vol. 2* 9–68, 9–69
undeliverable BNU jobs: *Vol. 2* 9–68
CMAX: *Vol. 3* 3–44
CMF: *Vol. 3* 3–39, 4–8
collect activity data, with sadc: *Vol. 3* 1–5
COM2CONS: *Vol. 3* 3–39
combine, BNU log files: *Vol. 2* 9–68, 9–69
commands for BNU: *Vol. 2* 8–5
configuration
files and directories: *Vol. 3* 3–2
parameters: *Vol. 3* 3–25
sendmail: *Vol. 1* E–16

configuration file
build for sendmail: *Vol. 1* F–39
sendmail: *Vol. 1* F–10, F–27
configuration options, sendmail: *Vol. 1* F–50
Configure, modems: *Vol. 2* 3–29
configure
dump area: *Vol. 2* 1–31
LP Print Service: *Vol. 2* 4–3
modem on serial controller: *Vol. 2* 3–25
printers: *Vol. 2* 4–12
second serial port: *Vol. 2* 3–20
serial controller ports for terminals: *Vol. 2* 3–22
serial port for a modem: *Vol. 2* 3–15
serial port for terminal: *Vol. 2* 3–13
serial port for three–wire cable: *Vol. 2* 3–35
control, disk usage: *Vol. 3* 7–7
copy
boot flex diskettes: *Vol. 1* 2–7
maintenance and reference diskettes: *Vol. 1* 2–6
maintenance flex diskette: *Vol. 1* 2–7
copy files, to NCR TOWERs: *Vol. 1* 6–25
copyright, parameter: *Vol. 3* 4–80
core file, size limit parameter: *Vol. 3* 4–122
corruption, file system: *Vol. 1* 5–2
cpu workload: *Vol. 3* 8–2
crash utility: *Vol. 1* 4–6
create
access mechanisms: *Vol. 2* 9–35
file system: *Vol. 1* 4–7
kernel: *Vol. 3* 3–11
menu entry: *Vol. 1* C–18
security mechanisms: *Vol. 2* 9–35
cron: *Vol. 2* 9–64

crontab command: *Vol. 2* 9–65
cu command: *Vol. 2* 9–73
customize
    interface program: *Vol. 2* 6–22
    print service: *Vol. 2* 6–12

D

daemon mode, sendmail: *Vol. 1* F–19
daemons, networking: *Vol. 2* 8–8
data base for BNU: *Vol. 2* 8–10
data segment, maximum size parameter: *Vol. 3* 4–124
daylight savings time, parameter: *Vol. 3* 4–12
deactivate, processor: *Vol. 1* 1–33
debug, sendmail: *Vol. 1* F–20
default
    priority setting: *Vol. 2* 5–3
    set for print queue priority: *Vol. 2* 5–3
define, filter: *Vol. 2* 5–35
delete
    menu: *Vol. 1* C–31
    task entry: *Vol. 1* C–31
delivery mode, sendmail: *Vol. 1* F–24
demand paging: *Vol. 3* 6–3
deny, access to printer: *Vol. 2* 4–46
DESFREE: *Vol. 3* 3–32, 4–9, 6–5
device, protocols: *Vol. 2* 9–27
device name: *Vol. 1* B–5
device numbers, major and minor: *Vol. 1* B–2
devices, used for BNU communication: *Vol. 2* 9–21
Devices file: *Vol. 2* 9–21
df command: *Vol. 1* 4–15; *Vol. 3* 7–8, 7–10

diagnostic diskette: *Vol. 1* 2–2
dial out failure: *Vol. 2* 4–54
Dial–up password protection
    disable: *Vol. 2* 3–34
    enable: *Vol. 2* 3–32
Dialers file: *Vol. 2* 9–30
dialing, international calls: *Vol. 2* 9–34
dialing information, write: *Vol. 2* 9–30
Direct Memory Access (DMA)
    exclusive ownership parameter: *Vol. 3* 4–11
    maximum click number parameter: *Vol. 3* 4–45
    pages reserved for: *Vol. 3* 4–10
directories, reorganizing: *Vol. 3* 7–14
Disable dial–up password protection: *Vol. 2* 3–34
disk
    add: *Vol. 2* 1–3
    display slicing information: *Vol. 2* 1–23
    format: *Vol. 2* 1–6
    monitor and control usage: *Vol. 3* 7–7
    monitor space used: *Vol. 3* 7–8
    monitor usage: *Vol. 3* 7–7
    partition: *Vol. 2* 1–8
    performance: *Vol. 3* 7–2
    replace: *Vol. 2* 1–35
    slice: *Vol. 2* 1–14
disk space, identify large users: *Vol. 3* 7–12
disk usage, monitor and control: *Vol. 3* 7–7
diskadd, use: *Vol. 2* 1–2
diskadd command: *Vol. 2* 1–4
diskette
    diagnostics: *Vol. 1* 2–2
    reference: *Vol. 1* 2–2

display
file system information: *Vol. 1* 4–15
filter: *Vol. 2* 5–44
form: *Vol. 2* 5–18
installed file system types: *Vol. 1* 4–6
process binding information: *Vol. 1* 1–35
queue priority: *Vol. 2* 5–3
slicing information: *Vol. 2* 1–23
status of printers: *Vol. 2* 4–37
status of processors: *Vol. 1* 1–33
dkformat command: *Vol. 2* 1–7
dklayout command: *Vol. 2* 1–13
DMAABLEBUF: *Vol. 3* 3–38, 4–10
DMAEXCL: *Vol. 3* 3–38, 4–11
DO386B1: *Vol. 3* 3–38
DO387CR3: *Vol. 3* 3–38
download fonts: *Vol. 2* 6–11
DSTFLAG: *Vol. 3* 3–38, 4–12
dump area
configure: *Vol. 2* 1–31
examine contents: *Vol. 1* 3–39
save memory to: *Vol. 1* 3–35
save to media: *Vol. 1* 3–37

E

edquota command: *Vol. 1* 4–21
edsysadm command: *Vol. 1* C–6
Enable dial–up password protection: *Vol. 2* 3–32
entire disk, restore: *Vol. 1* 7–10
entire system
backing up: *Vol. 1* 6–19
restoring: *Vol. 1* 7–20

/etc/conf: *Vol. 3* 3–3
/etc/d_passwd: *Vol. 2* 3–32
/etc/dialups: *Vol. 2* 3–32
/etc/netconfig: *Vol. 2* 7–2
/etc/passwd: *Vol. 1* 1–28
    recover: *Vol. 1* 3–13
/etc/shadow: *Vol. 1* 1–28
    recover: *Vol. 1* 3–17
    update: *Vol. 1* 1–29
/etc/uucp: *Vol. 2* 8–10
/etc/vfstab: *Vol. 1* 4–4
Ethernet LAN Module, install: *Vol. 2* 2–23
EVDATA: *Vol. 3* 3–41
EVFNHTS: *Vol. 3* 3–41
EVMAXDPE: *Vol. 3* 3–41
EVMAXETERMS: *Vol. 3* 3–41
EVMAXEV: *Vol. 3* 3–41
EVMAXMEM: *Vol. 3* 3–41
EVMAXTRAPS: *Vol. 3* 3–41
EVTIDHTS: *Vol. 3* 3–41
exit codes, printer interface program: *Vol. 2* 6–24

F

failure, dial out: *Vol. 2* 4–54
fault, printer: *Vol. 2* 5–7
FDFLUSHR: *Vol. 3* 3–29, 4–13
file
    copy to NCR TOWERS: *Vol. 1* 6–25
    identifying inactive: *Vol. 3* 7–11
    removing inactive: *Vol. 3* 7–11
file descriptor, maximum per process: *Vol. 3* 4–135

file mode, sendmail: *Vol. 1* F–25
file system
check and repair: *Vol. 1* 1–12, 5–4
check from maintenance file system: *Vol. 1* 3–40
create: *Vol. 1* 4–7
display information about: *Vol. 1* 4–15
eliminate fragmentation: *Vol. 3* 7–16
identify type of: *Vol. 1* 4–6
integrity: *Vol. 1* 5–2
make: *Vol. 1* 4–7
maximum mounted parameter: *Vol. 3* 4–87
minimizing corruption: *Vol. 1* 5–2
mount: *Vol. 1* 4–11
organization: *Vol. 1* A–2, A–6
unmount: *Vol. 1* 4–13
file system type
bfs: *Vol. 1* A–23
cdfs: *Vol. 1* A–27
list: *Vol. 1* 4–6
parameter for root: *Vol. 3* 4–118
s5: *Vol. 1* A–10
ufs: *Vol. 1* A–17
file system types: *Vol. 1* A–8
file sytem and disk information, backup: *Vol. 1* 6–16
file transfer protocol: *Vol. 2* 9–60
filter: *Vol. 2* 5–32
change: *Vol. 2* 5–44
define: *Vol. 2* 5–35
display: *Vol. 2* 5–44
fast: *Vol. 2* 6–26
PostScript: *Vol. 2* 6–7
remove: *Vol. 2* 5–44
restore defaults: *Vol. 2* 5–45
FLCKREC: *Vol. 3* 3–29, 4–14

font
    downloading: *Vol. 2* 6–11
    PostScript: *Vol. 2* 6–8
force queue, sendmail: *Vol. 1* F–19
form
    display: *Vol. 2* 5–18
    provide to printer: *Vol. 2* 5–12
    restrict user access: *Vol. 2* 5–16
format, disk: *Vol. 2* 1–6
format command: *Vol. 2* 1–7
forward files, sendmail: *Vol. 1* F–18
forwarding, sendmail: *Vol. 1* E–14
free block list, reorganize: *Vol. 3* 5–5
free memory, minimum parameter: *Vol. 3* 4–58
fsck
    automatic checking: *Vol. 1* 1–12, 1–13
    from maiantenance file system: *Vol. 1* 3–40
    manual checking: *Vol. 1* 1–12, 1–14
    on bfs file systems: *Vol. 1* 5–78
    phases on ufs file systems: *Vol. 1* 5–47
fstyp command: *Vol. 1* 4–6

G

GPGSHI: *Vol. 3* 3–32
GPGSLO: *Vol. 3* 3–32, 4–15, 6–7
GPGSMSK: *Vol. 3* 3–32

H

hard disk, display slicing information: *Vol. 2* 1–23
hard limit parameter: *Vol. 3* 1–6
hardware provider, parameter for name: *Vol. 3* 4–25
hash buffer, maximum parameter: *Vol. 3* 4–84

HCORLIM: *Vol. 3* 3–43, 4–16
HCPULIM: *Vol. 3* 3–43, 4–17
HDATLIM: *Vol. 3* 3–43, 4–18
header declarations, sendmail: *Vol. 1* E–17
help messages, write: *Vol. 1* C–8
HFNOLIM: *Vol. 3* 3–43, 4–19
HFSZLIM: *Vol. 3* 3–43
hold, print request: *Vol. 2* 5–5
host–resident fonts: *Vol. 2* 6–9
HRSZLIM: *Vol. 3* 4–20
HRTIME: *Vol. 3* 3–42, 4–21
HRVTIME: *Vol. 3* 3–42, 4–22
HSTKLIM: *Vol. 3* 3–43, 4–23
HVMMLIM: *Vol. 3* 3–43, 4–24
HW_PROVIDER: *Vol. 3* 3–47, 4–25
HW_SERIAL: *Vol. 3* 3–47, 4–26

I

I/O balancing: *Vol. 3* 7–3
identify
- devices used for BNU communication: *Vol. 2* 9–21
- inactive file: *Vol. 3* 7–11
- systems for BNU communications: *Vol. 2* 9–11
- type of file system: *Vol. 1* 4–6

idle printer: *Vol. 2* 4–55
idtune command: *Vol. 3* 3–26
ILDMAXOPENS: *Vol. 3* 3–45
illegible output, printers: *Vol. 2* 4–50
inactive file, identify and remove: *Vol. 3* 7–11
inclusion, sendmail: *Vol. 1* E–15
init command: *Vol. 1* 1–5

inode, maximum parameter for s5: *Vol. 3* 4–85
/install: *Vol. 1* 2–11
install
  Ethernet LAN Module: *Vol. 2* 2–23
  from remote console: *Vol. 1* 1–47
  LP Print Service: *Vol. 2* 4–11
  Postscript fonts: *Vol. 2* 6–8
  PostScript printer: *Vol. 2* 6–7
  PostScript printers: *Vol. 2* 6–6
  remote installation equipment requirements: *Vol. 1* 1–48
  sendmail: *Vol. 1* F–4
  serial controller: *Vol. 2* 2–18
  setting up remote console: *Vol. 1* 1–49
  software packages from flex: *Vol. 1* 1–30
  software packages from tape: *Vol. 1* 1–30
installation, from remote console: *Vol. 1* 1–47
installation tape, recover files from: *Vol. 1* 3–21
integrity of file system: *Vol. 1* 5–2
interface program: *Vol. 2* 6–18
  customize: *Vol. 2* 6–22
international calls, dial: *Vol. 2* 9–34
item help file: *Vol. 1* C–6

K

kdb symbol table, maximum size parameter: *Vol. 3* 4–31
KDBSYMSIZE: *Vol. 3* 3–38, 4–31
kernel
  create new: *Vol. 3* 3–11
  load/boot: *Vol. 3* 3–15
kernel memory
  maximum used by RFS users: *Vol. 3* 4–117
  minimum available parameter: *Vol. 3* 4–55

keyboard mapping, change: *Vol. 1* D–4
keyword: *Vol. 2* 5–39

L

L5WatchDogPeriod, parameters: *Vol. 3* 4–40
L5WatchDogPFRPeriod, parameters: *Vol. 3* 4–39
level of logging, sendmail: *Vol. 1* F–25
limit
    display for print queue priority: *Vol. 2* 5–3
    for parameters: *Vol. 3* 1–6
    set for priority: *Vol. 2* 5–3
limit load, sendmail: *Vol. 1* F–24
Limits file: *Vol. 2* 9–53
line characteristics, change to 8 bit: *Vol. 1* D–2
link, symbolic: *Vol. 1* A–5
list
    character set: *Vol. 2* 5–26
    file system information: *Vol. 1* 4–15
    installed file system types: *Vol. 1* 4–6
    print wheel: *Vol. 2* 5–25
    software packages: *Vol. 1* 1–30
load
    kernel: *Vol. 3* 3–15
    manage for printer: *Vol. 2* 4–41
log, print requests: *Vol. 2* 5–46
log file, check size of: *Vol. 2* 9–68, 9–70
log files, combine for BNU: *Vol. 2* 9–68, 9–69
log in: *Vol. 1* 1–2
log level, sendmail: *Vol. 1* F–25
logs, for BNU: *Vol. 2* 8–16
LOTSFREE: *Vol. 3* 3–32, 4–42, 6–5
lp package: *Vol. 2* 4–11

LP Print Service
customize: *Vol. 2* 6–12
description: *Vol. 2* 4–2
display filter: *Vol. 2* 5–44
filter: *Vol. 2* 5–32
install: *Vol. 2* 4–11
remove filter: *Vol. 2* 5–44
start: *Vol. 2* 4–39
stop: *Vol. 2* 4–39
lpfilter: *Vol. 2* 5–44
lpforms: *Vol. 2* 5–15, 5–16
lpstat: *Vol. 2* 4–37

M

machine architecture, parameter: *Vol. 3* 4–5
macros, sendmail: *Vol. 1* E–17
mail queue, sendmail: *Vol. 1* F–11
mailer declaration, sendmail: *Vol. 1* E–17
mailer flag, sendmail: *Vol. 1* F–55
maintain
postScript: *Vol. 2* 6–7
PostScript fonts: *Vol. 2* 6–8
PostScript printers: *Vol. 2* 6–6
maintenance diskette, copy: *Vol. 1* 2–6
Maintenance File System, access: *Vol. 1* 2–11
maintenance file system: *Vol. 1* 2–2
check file system: *Vol. 1* 3–40
maintenance flex diskette, copy: *Vol. 1* 2–7
major device numbers: *Vol. 1* B–2
make, file system: *Vol. 1* 4–7

manage
printer faults: *Vol. 2* 5–7
printer load: *Vol. 2* 4–41
queue priorities: *Vol. 2* 5–2
quotas: *Vol. 1* 4–21
map
change keyboard: *Vol. 1* D–4
change screen: *Vol. 1* D–6
character set names: *Vol. 2* 5–26
map table, for PostScript fonts: *Vol. 2* 6–10
mapkey command: *Vol. 1* D–4
mapping, Name–to–Address: *Vol. 2* 7–10
mapscrn command: *Vol. 1* D–6
mark, bad blocks: *Vol. 2* 1–19
MAXAIOS: *Vol. 3* 3–40, 4–43
MAXCLSYSPRI: *Vol. 3* 3–42, 4–44
MAXDMAPAGE: *Vol. 3* 3–38, 4–45
MAXFC: *Vol. 3* 3–32
MAXGDP: *Vol. 3* 3–37, 4–46
maximum click number for DMA, parameter: *Vol. 3* 4–45
MAXMINOR: *Vol. 3* 3–29, 4–47
MAXPMEM: *Vol. 3* 3–29, 4–48
MAXSC: *Vol. 3* 3–32
MAXSEPGCNT: *Vol. 3* 3–33
MAXSERVE: *Vol. 3* 3–37, 4–49
MAXSLICE: *Vol. 3* 3–29, 4–50
MAXUMEM: *Vol. 3* 3–32
MAXUP: *Vol. 3* 3–29, 4–51
MAXWCONSMSG: *Vol. 3* 3–34, 4–52
MAXWERRMSG: *Vol. 3* 3–34, 4–53
MAXWTRCMSG: *Vol. 3* 3–34, 4–54

memory
management: *Vol. 3* 6–2
maximum pages lock in: *Vol. 3* 4–105
maximum physical to use: *Vol. 3* 4–48
minimum available for kernel: *Vol. 3* 4–55
minimum free parameter: *Vol. 3* 4–58
parameter for amount of sufficient free: *Vol. 3* 4–42
parameter for desired free: *Vol. 3* 4–9
save to dump area: *Vol. 1* 3–35
memory dump, customize: *Vol. 1* 1–44
menu, delete: *Vol. 1* C–31
menu entry
change: *Vol. 1* C–18
create: *Vol. 1* C–18
menus
create an entry: *Vol. 1* C–18
delete task entry: *Vol. 1* C–31
message, maximum size parameter: *Vol. 3* 4–62
message collection, sendmail: *Vol. 1* E–15
message delivery, sendmail: *Vol. 1* E–16
message queue
maximum length parameter: *Vol. 3* 4–63
maximum parameter: *Vol. 3* 4–64
message segment
maximum parameter: *Vol. 3* 4–65
size parameter: *Vol. 3* 4–66
MEVDIRENTS: *Vol. 3* 3–41
MEVEXITS: *Vol. 3* 3–41
MEVEXPRS: *Vol. 3* 3–41
MEVEXREFS: *Vol. 3* 3–41
MEVKEVS: *Vol. 3* 3-41
MEVQUEUES: *Vol. 3* 3–41

MEVRETRYS: *Vol. 3* 3–41
MEVSEXPRS: *Vol. 3* 3–41
MEVSIGS: *Vol. 3* 3–41
MEVSTERMS: *Vol. 3* 3–41
MEVSTRDS: *Vol. 3* 3–41
MEVTERMS: *Vol. 3* 3–41
MEVTIDS: *Vol. 3* 3–41
MINAIOS: *Vol. 3* 3–40
MINAKMEM: *Vol. 3* 3–32, 4–55, 6–8
MINARMEM: *Vol. 3* 3–32, 4–56, 6–8
MINASMEM: *Vol. 3* 3–32, 4–57, 6–8
MINFREE: *Vol. 3* 3–32, 4–58, 6–6
MINHIDUSTK: *Vol. 3* 3–32
minor device, number configured for log driver: *Vol. 3* 4–86
minor device numbers: *Vol. 1* B–2
minor number, largest allowed on system: *Vol. 3* 4–47
MINPAGEFREE: *Vol. 3* 3–32, 4–59
MINSERVE: *Vol. 3* 3–37, 4–60
MINUSTKGAP: *Vol. 3* 3–32
mkfs command: *Vol. 1* 4–9
MNR_ON: *Vol. 3* 3–39
mode, sendmail files: *Vol. 1* F–25
modem
    configure on serial controller: *Vol. 2* 3–25
    configure on serial port: *Vol. 2* 3–15
Modems, configure: *Vol. 2* 3–29
module, maximum pushed on stream: *Vol. 3* 4–98
modules: *Vol. 3* 3–17
monitor, disk usage: *Vol. 3* 7–7
mount, file system: *Vol. 1* 4–11

move
print request to head of queue: *Vol. 2* 5–6
request in print queue: *Vol. 2* 5–3
MSGMAP: *Vol. 3* 3–35, 4–61
MSGMAX: *Vol. 3* 3–35, 4–62
MSGMNB: *Vol. 3* 3–35, 4–63
MSGMNI: *Vol. 3* 3–35, 4–64
MSGMSEG: *Vol. 3* 4–65
MSGSEG: *Vol. 3* 3–35
MSGSSZ: *Vol. 3* 3–35, 4–66
MSGTQL: *Vol. 3* 3–35, 4–67
multi–user state: *Vol. 1* 1–5

N

NADVERTISE: *Vol. 3* 3–37, 4–68
NAIOPROC: *Vol. 3* 3–40
NAIOSYS: *Vol. 3* 3–40, 4–69
name
character set: *Vol. 2* 5–25
device: *Vol. 1* B–5
print wheel: *Vol. 2* 5–25
Name–to–Address mapping: *Vol. 2* 7–10
national keyboard, map: *Vol. 1* D–4
NAUTOPUSH: *Vol. 3* 3–34, 4–70
NAUTOUP: *Vol. 3* 3–29, 4–71
NBLK parameters: *Vol. 3* 3–33
NBUF: *Vol. 3* 3–29, 4–72
NCALL: *Vol. 3* 3–29, 4–74
NCDEXTENT, parameters: *Vol. 3* 4–75
NCDFILSYS, parameters: *Vol. 3* 4–76
NCDINODE, parameters: *Vol. 3* 4–77

NCLIST: *Vol. 3* 3–29, 4–78
NCPU: *Vol. 3* 3–29, 4–79
NCPYRIGHT: *Vol. 3* 3–39, 4–80
NCR TOWERS, copy files to: *Vol. 1* 6–25
NDQUOT: *Vol. 3* 3–31, 4–81
NEMAP: *Vol. 3* 3–38, 4–82
NETPATH: *Vol. 2* 7–2
network problems, printer: *Vol. 2* 4–55
network selection: *Vol. 2* 7–2
networking daemons: *Vol. 2* 8–8
NFILE: *Vol. 3* 3–29
NGROUPS_MAX: *Vol. 3* 3–29, 4–83
NHBUF: *Vol. 3* 3–29, 4–84
NINODE: *Vol. 3* 3–31, 4–85
NKDVTTY: *Vol. 3* 3–39
NLOCAL: *Vol. 3* 3–37
NLOG: *Vol. 3* 3–34, 4–86
NMOUNT: *Vol. 3* 3–29, 4–87
NMUXLINK: *Vol. 3* 3–33
NODE: *Vol. 3* 3–46, 4–88
node name
    parameter: *Vol. 3* 4–88
    set for running kernel: *Vol. 2* 9–9
NOFILES: *Vol. 3* 3–29
NOTPGOVERFLOW: *Vol. 3* 3–44, 4–89
NPBUF: *Vol. 3* 3–29
NPROC: *Vol. 3* 3–29, 4–90
NQUEUE: *Vol. 3* 3–33
NRCVD: *Vol. 3* 3–37, 4–91
NRDUSER: *Vol. 3* 3–37, 4–92
NREGION: *Vol. 3* 3–29

NREMOTE: *Vol. 3* 3–37
NRNODE: *Vol. 3* 3–31, 4–93
NS5INODE: *Vol. 3* 3–31
NSCRN: *Vol. 3* 3–38, 4–94
NSNDD: *Vol. 3* 3–37, 4–95
NSRMOUNT: *Vol. 3* 3–37, 4–96
NSTREAM: *Vol. 3* 3–33
NSTREVENT: *Vol. 3* 3–33
NSTRPHASH: *Vol. 3* 3–34, 4–97
NSTRPUSH: *Vol. 3* 3–33, 4–98
NUMSAD: *Vol. 3* 3–34, 4–99
NUMSP: *Vol. 3* 3–34, 4–100
NUMSXT: *Vol. 3* 3–39, 4–101
NUMTIM: *Vol. 3* 3–34, 4–102
NUMTRW: *Vol. 3* 3–34, 4–103
NUMXT: *Vol. 3* 3–39, 4–104

O

offline: *Vol. 1* 1–34
online: *Vol. 1* 1–34
option
    for print service: *Vol. 2* 4–27
    set for sendmail: *Vol. 1* E–18
organization, file system: *Vol. 1* A–2, A–6
osa login: *Vol. 1* 1–2
output illegible, printers: *Vol. 2* 4–50

P

packages, software: *Vol. 1* 1–30
page
maximum locked in memory parameter: *Vol. 3* 4–105
minimum free parameter: *Vol. 3* 4–59
PAGES_UNLOCK: *Vol. 3* 3–32, 4–105
parameter
AFFIN_ON: *Vol. 3* 3–42, 4–2
AFFINDECAY: *Vol. 3* 3–42, 4–3
affinity scheduling: *Vol. 3* 3–42
AGEINTERVAL: *Vol. 3* 3–32
AIOTIMEOUT: *Vol. 3* 3–40, 4–4
ARCHITECTURE: *Vol. 3* 3–47, 4–5
ARG_MAX: *Vol. 3* 3–29, 4–6
BDFLUSHR: *Vol. 3* 3–29
BMAX: *Vol. 3* 3–44
BUFHWM: *Vol. 3* 3–29, 4–7
CMAX: *Vol. 3* 3–44
CMF: *Vol. 3* 3–39, 4–8
COM2CONS: *Vol. 3* 3–39
configuration: *Vol. 3* 3–25
D0387CR3: *Vol. 3* 3–38
DESFREE: *Vol. 3* 3–32, 4–9, 6–5
DMAABLEBUF: *Vol. 3* 3–38, 4–10
DMAEXCL: *Vol. 3* 3–38, 4–11
DO386B1: *Vol. 3* 3–38
DSTFLAG: *Vol. 3* 3–38, 4–12
EVDATA: *Vol. 3* 3–41
EVFNHTS: *Vol. 3* 3–41
EVMAXDPE: *Vol. 3* 3–41
EVMAXETERMS: *Vol. 3* 3–41
EVMAXEV: *Vol. 3* 3–41
EVMAXMEM: *Vol. 3* 3–41
EVMAXTRAPS: *Vol. 3* 3–41

EVTIDHTS: *Vol. 3* 3–41
FDFLUSHR: *Vol. 3* 3–29, 4–13
FLCKREC: *Vol. 3* 3–29, 4–14
GPGSHI: *Vol. 3* 3–32
GPGSLO: *Vol. 3* 3–32, 4–15, 6–7
GPGSMSK: *Vol. 3* 3–32
HCORLIM: *Vol. 3* 3–43, 4–16
HCPULIM: *Vol. 3* 3–43, 4–17
HDATLIM: *Vol. 3* 3–43, 4–18
HFNOLIM: *Vol. 3* 3–43, 4–19
HFSZLIM: *Vol. 3* 3–43, 4–20
HRTIME: *Vol. 3* 3–42, 4–21
HRVTIME: *Vol. 3* 3–42, 4–22
HSTKLIM: *Vol. 3* 3–43, 4–23
HVMMLIM: *Vol. 3* 3–43, 4–24
HW_PROVIDER: *Vol. 3* 3–47, 4–25
HW_SERIAL: *Vol. 3* 3–47, 4–26
ILDMAXOPENS: *Vol. 3* 3–45
KDBSYMSIZE: *Vol. 3* 3–38, 4–31
L5WatchDogPeriod: *Vol. 3* 4–40
L5WatchDogPFRPeriod: *Vol. 3* 4–39
LOTSFREE: *Vol. 3* 3–32, 4–42, 6–5
MAXAIOS: *Vol. 3* 3–40, 4–43
MAXCLSYSPRI: *Vol. 3* 3–42, 4–44
MAXDMAPAGE: *Vol. 3* 3–38, 4–45
MAXFC: *Vol. 3* 3–32
MAXGDP: *Vol. 3* 3–37, 4–46
MAXMINOR: *Vol. 3* 3–29, 4–47
MAXPMEM: *Vol. 3* 3–29, 4–48
MAXSC: *Vol. 3* 3–32
MAXSEPGCNT: *Vol. 3* 3–33
MAXSERVE: *Vol. 3* 3–37, 4–49
MAXSLICE: *Vol. 3* 3–29, 4–50
MAXUMEM: *Vol. 3* 3–32
MAXUP: *Vol. 3* 3–29, 4–51

MAXWCONSMSG: *Vol. 3* 3–34, 4–52
MAXWERRMSG: *Vol. 3* 3–34, 4–53
MAXWTRCMSG: *Vol. 3* 3–34, 4–54
MEVDIRENTS: *Vol. 3* 3–41
MEVEXITS: *Vol. 3* 3–41
MEVEXPRS: *Vol. 3* 3–41
MEVEXREFS: *Vol. 3* 3–41
MEVKEVS: *Vol. 3* 3–41
MEVQUEUES: *Vol. 3* 3–41
MEVRETRYS: *Vol. 3* 3–41
MEVSEXPRS: *Vol. 3* 3–41
MEVSIGS: *Vol. 3* 3–41
MEVSTERMS: *Vol. 3* 3–41
MEVSTRDS: *Vol. 3* 3–41
MEVTERMS: *Vol. 3* 3–41
MEVTIDS: *Vol. 3* 3–41
MINAIOS: *Vol. 3* 3–40
MINAKMEM: *Vol. 3* 3–32, 4–55, 6–8
MINARMEM: *Vol. 3* 3–32, 4–56, 6–8
MINASMEM: *Vol. 3* 3–32, 4–57, 6–8
MINFREE: *Vol. 3* 3–32, 4–58, 6–6
MINHIDUSTK: *Vol. 3* 3–32
MINPAGEFREE: *Vol. 3* 3–32, 4–59
MINSERVE: *Vol. 3* 3–37, 4–60
MINUSTKGAP: *Vol. 3* 3–32
MNR_ON: *Vol. 3* 3–39
MSGMAP: *Vol. 3* 3–35, 4–61
MSGMAX: *Vol. 3* 3–35, 4–62
MSGMNB: *Vol. 3* 3–35, 4–63
MSGMNI: *Vol. 3* 3–35, 4–64
MSGSEG: *Vol. 3* 3–35, 4–65
MSGSSZ: *Vol. 3* 3–35, 4–66
MSGTQL: *Vol. 3* 3–35, 4–67
NADVERTISE: *Vol. 3* 3–37, 4–68
NAIOPROC: *Vol. 3* 3–40

NAIOSYS: *Vol. 3* 3–40, 4–69
NAUTOPUSH: *Vol. 3* 3–34, 4–70
NAUTOUP: *Vol. 3* 3–29, 4–71
NBLK: *Vol. 3* 3–33
NBUF: *Vol. 3* 3–29, 4–72
NCALL: *Vol. 3* 3–29, 4–74
NCDEXTENT: *Vol. 3* 4–75
NCDFILSYS: *Vol. 3* 4–76
NCDINODE: *Vol. 3* 4–77
NCLIST: *Vol. 3* 3–29, 4–78
NCPU: *Vol. 3* 3–29, 4–79
NCPYRIGHT: *Vol. 3* 3–39, 4–80
NDQUOT: *Vol. 3* 3–31, 4–81
NEMAP: *Vol. 3* 3–38, 4–82
NFILE: *Vol. 3* 3–29
NGROUPS_MAX: *Vol. 3* 3–29, 4–83
NHBUF: *Vol. 3* 3–29, 4–84
NINODE: *Vol. 3* 3–31, 4–85
NKDVTTY: *Vol. 3* 3–39
NLOCAL: *Vol. 3* 3–37
NLOG: *Vol. 3* 3–34, 4–86
NMOUNT: *Vol. 3* 3–29, 4–87
NMUXLINK: *Vol. 3* 3–33
NODE: *Vol. 3* 3–46, 4–88
NOFILES: *Vol. 3* 3–29
NOTPGOVERFLOW: *Vol. 3* 3–44, 4–89
NPBUF: *Vol. 3* 3–29
NPROC: *Vol. 3* 3–29, 4–90
NQUEUE: *Vol. 3* 3–33
NRCVD: *Vol. 3* 3–37, 4–91
NRDUSER: *Vol. 3* 3–37, 4–92
NREGION: *Vol. 3* 3–29
NREMOTE: *Vol. 3* 3–37
NRNODE: *Vol. 3* 3–31, 4–93
NS5INODE: *Vol. 3* 3–31

NSCRN: *Vol. 3* 3–38, 4–94
NSNDD: *Vol. 3* 3–37, 4–95
NSRMOUNT: *Vol. 3* 3–37, 4–96
NSTREAM: *Vol. 3* 3–33
NSTREVENT: *Vol. 3* 3–33
NSTRPHASH: *Vol. 3* 3–34, 4–97
NSTRPUSH: *Vol. 3* 3–33, 4–98
NUMSAD: *Vol. 3* 3–34, 4–99
NUMSP: *Vol. 3* 3–34, 4–100
NUMSXT: *Vol. 3* 3–39, 4–101
NUMTIM: *Vol. 3* 3–34, 4–102
NUMTRW: *Vol. 3* 3–34, 4–103
NUMXT: *Vol. 3* 3–39, 4–104
PAGES_UNLOCK: *Vol. 3* 3–32, 4–105
PGOVERFLOW: *Vol. 3* 3–44, 4–106
PIOMAP: *Vol. 3* 3–30, 4–107
PIOMAXSZ: *Vol. 3* 3–30, 4–108
PIOSEGSZ: *Vol. 3* 3–38, 4–109
PRFMAX: *Vol. 3* 3–39, 4–110
PUTBUFSZ: *Vol. 3* 3–30, 4–111
RCACHETIME: *Vol. 3* 3–37
RCHACHETIME: *Vol. 3* 4–114
RCMF: *Vol. 3* 3–39, 4–115
REL: *Vol. 3* 3–46, 4–116
RF_MAXKMEM: *Vol. 3* 3–37, 4–117
RFHEAP: *Vol. 3* 3–37
RFS_VHIGH: *Vol. 3* 3–37
RFS_VLOW: *Vol. 3* 3–37
RIDEOUT: *Vol. 3* 3–39
ROOTFSTYPE: *Vol. 3* 4–118
RSTCHOWN: *Vol. 3* 3–30, 4–119
RTMAXPRI: *Vol. 3* 3–42
RTNPROCS: *Vol. 3* 3–42, 4–120
S52KNBUF: *Vol. 3* 3–31
S52KNHBUF: *Vol. 3* 3–31

SANECNT: *Vol. 3* 3–39
SANITYCLK: *Vol. 3* 3–38
SATENABLED: *Vol. 3* 3–45, 4–121
scheduler: *Vol. 3* 3–42
SCORLIM: *Vol. 3* 3–43, 4–122
SCPULIM: *Vol. 3* 3–43, 4–123
SDATLIM: *Vol. 3* 3–43, 4–124
SEGMAPSZ: *Vol. 3* 3–38, 4–125
SEMAEM: *Vol. 3* 3–35, 4–126
SEMMAP: *Vol. 3* 3–35, 4–127
SEMMNI: *Vol. 3* 3–35, 4–128
SEMMNS: *Vol. 3* 3–35, 4–129
SEMMNU: *Vol. 3* 3–35, 4–130
SEMMSL: *Vol. 3* 3–35, 4–131
SEMOPM: *Vol. 3* 3–35, 4–132
SEMUME: *Vol. 3* 3–35, 4–133
SEMVMX: *Vol. 3* 3–35, 4–134
SFNOLIM: *Vol. 3* 3–43, 4–135
SFSZLIM: *Vol. 3* 3–43, 4–136
SHLBMAX: *Vol. 3* 3–30, 4–137
SHMMAX: *Vol. 3* 3–36, 4–139
SHMMIN: *Vol. 3* 3–36, 4–140
SHMMNI: *Vol. 3* 3–36, 4–141
SHM_NAILED_GID[1–9]: *Vol. 3* 4–138
SHMSEG: *Vol. 3* 3–36, 4–142
SOCK_HIWATER: *Vol. 3* 3–34, 4–143
SOCK_LOWATER: *Vol. 3* 3–34, 4–144
SPTMAP: *Vol. 3* 3–30
SRPC_DOMAIN: *Vol. 3* 3–46, 4–145
SSTKLIM: *Vol. 3* 3–43, 4–146
STRCTLSZ: *Vol. 3* 3–33, 4–147
STRLOFRAC: *Vol. 3* 3–33
STRMEDFRAC: *Vol. 3* 3–33
STRMSGSZ: *Vol. 3* 3–33, 4–148
STRTHRESH: *Vol. 3* 3–33, 4–149

SVMMLIM: *Vol. 3* 3–43, 4–150
SYS: *Vol. 3* 3–46, 4–151
TIM_HIWATER: *Vol. 3* 3–34, 4–152
TIM_LOWATER: *Vol. 3* 3–34, 4–153
timer: *Vol. 3* 3–42
TIMEZONE: *Vol. 3* 3–38, 4–154
TRACESZ: *Vol. 3* 3–30, 4–155
TRW_HIWATER: *Vol. 3* 3–34, 4–156
TRW_LOWATER: *Vol. 3* 3–34, 4–157
TSMAXUPRI: *Vol. 3* 3–42, 4–158
TSNPROCS: *Vol. 3* 3–42, 4–159
UFSINODE: *Vol. 3* 3–31
UFSNINODE: *Vol. 3* 4–161
ULIMIT: *Vol. 3* 3–30, 4–162
USANEON: *Vol. 3* 3–39
VER: *Vol. 3* 3–46, 4–163
VHNDFRAC: *Vol. 3* 3–32
XSDSEGS: *Vol. 3* 3–38, 4–164
XSDSLOTS: *Vol. 3* 3–38, 4–165
XSEMMAX: *Vol. 3* 3–38, 4–166
parity setting, printer: *Vol. 2* 4–51
partition
change active: *Vol. 2* 1–24
disk: *Vol. 2* 1–8
password, recover for root: *Vol. 1* 3–10
password file: *Vol. 1* 1–28
password shadowing: *Vol. 1* 1–28
PATH variables: *Vol. 3* 8–3
pbind: *Vol. 1* 1–35
PBS, power backup system: *Vol. 1* 1–37
Permissions file: *Vol. 2* 9–35
examples: *Vol. 2* 9–45
PGOVERFLOW: *Vol. 3* 3–44, 4–106

phases of fsck, on ufs file systems: *Vol. 1* 5–47
physical connection, for BNU: *Vol. 2* 9–7
physical memory, maximum amount to use: *Vol. 3* 4–48
pinfo: *Vol. 1* 1–33
PIOMAP: *Vol. 3* 3–30, 4–107
PIOMAXSZ: *Vol. 3* 3–30, 4–108
PIOSEGSZ: *Vol. 3* 3–38, 4–109
pkgadd: *Vol. 2* 4–11
pmadm(1M): *Vol. 2* 3–7
port monitor
    administrative file: *Vol. 2* 3–5
    ttymon: *Vol. 2* 3–8
PostScript, maintain: *Vol. 2* 6–6
PostScript fonts
    host–resident: *Vol. 2* 6–9
    map table: *Vol. 2* 6–10
    printer–resident: *Vol. 2* 6–9
PostScript printer: *Vol. 2* 6–2
    install: *Vol. 2* 6–6, 6–7
power backup systems, PBS: *Vol. 1* 1–37
powerfail, strategy: *Vol. 1* 1–37
PRFMAX: *Vol. 3* 3–39, 4–110
print, sendmail mail queue: *Vol. 1* F–11
print request
    hold: *Vol. 2* 5–5
    move in queue: *Vol. 2* 5–3
    move to head of queue: *Vol. 2* 5–6
print service
    customize: *Vol. 2* 6–12
    install: *Vol. 2* 4–11
    start: *Vol. 2* 4–39
    stop: *Vol. 2* 4–39

print service options: *Vol. 2* 4–27
print wheel: *Vol. 2* 5–24
list: *Vol. 2* 5–25
name: *Vol. 2* 5–25
printer
adjust port characteristics: *Vol. 2* 6–12
character set: *Vol. 2* 5–24
configure: *Vol. 2* 4–12
default priority: *Vol. 2* 5–3
dial out failures: *Vol. 2* 4–54
display filter: *Vol. 2* 5–44
display status of: *Vol. 2* 4–37
faults: *Vol. 2* 5–7
filter: *Vol. 2* 5–32
filter defaults: *Vol. 2* 5–45
form: *Vol. 2* 5–12
idle: *Vol. 2* 4–55
interface program: *Vol. 2* 6–18
managing load: *Vol. 2* 4–41
networking problems: *Vol. 2* 4–55
print wheel: *Vol. 2* 5–24
queue priorities: *Vol. 2* 5–2
remove filter: *Vol. 2* 5–44
troubleshooting: *Vol. 2* 4–49
printer–resident fonts: *Vol. 2* 6–9
priorities, print queue: *Vol. 2* 5–2
priority
default: *Vol. 2* 5–3
display limits and defaults: *Vol. 2* 5–3
sendmail queue: *Vol. 1* F–23
set limits: *Vol. 2* 5–3

process
analyze with time: *Vol. 3* 1–45
analyze with timex: *Vol. 3* 1–47
bind: *Vol. 1* 1–35
maximum address space parameter: *Vol. 3* 4–150
maximum created parameter: *Vol. 3* 4–90
maximum group parameter: *Vol. 3* 4–83
maximum size of stack segment: *Vol. 3* 4–146
size of data segment parameter: *Vol. 3* 4–124
process accounting: *Vol. 3* 1–35
shutdown: *Vol. 3* 1–36
process binding, display information: *Vol. 1* 1–35
process table, number of entries to allocate: *Vol. 3* 4–90
processor
activate: *Vol. 1* 1–33
deactivate: *Vol. 1* 1–33
processors, display status: *Vol. 1* 1–33
profile system: *Vol. 3* 1–49
profiler command: *Vol. 3* 1–49
protocol
file transfer: *Vol. 2* 9–60
for devices: *Vol. 2* 9–27
prtvtoc command: *Vol. 2* 1–23
public area, clean up: *Vol. 2* 9–68, 9–69
PUTBUFSZ: *Vol. 3* 3–30, 4–111

Q

queue
move request: *Vol. 2* 5–3
sendmail: *Vol. 1* F–11
queue interval, sendmail: *Vol. 1* F–19

queue priorities: *Vol. 2* 5–2
sendmail: *Vol. 1* F–23
queued message, sendmail: *Vol. 1* E–16
quot command: *Vol. 1* 4–22
quota structure, parameter for ufs: *Vol. 3* 4–81
quotacheck command: *Vol. 1* 4–21
quotaoff command: *Vol. 1* 4–22
quotaon command: *Vol. 1* 4–22
quotas: *Vol. 1* 4–20
manage: *Vol. 1* 4–21
set: *Vol. 1* 4–20
turn off: *Vol. 1* 4–22
turn on: *Vol. 1* 4–22

R

raw devices
backup: *Vol. 1* 6–14
restore: *Vol. 1* 7–8
raw I/O, size of segment parameter: *Vol. 3* 4–109
RCACHETIME: *Vol. 3* 3–37, 4–114
RCMF: *Vol. 3* 3–39, 4–115
real-time processes, parameter: *Vol. 3* 4–120
receive, printer fault alert: *Vol. 2* 5–8
receive descriptor, maximum user entires parameter: *Vol. 3* 4–92
receive descriptors, number configured parameter: *Vol. 3* 4–91
record locking regions, parameter for maximum number: *Vol. 3* 4–14

recover
/etc/passwd: *Vol. 1* 3–13
/etc/shadow: *Vol. 1* 3–17
BOOT program: *Vol. 1* 3–29
files from installation tape: *Vol. 1* 3–21
files in the /stand file system: *Vol. 1* 3–2
root password: *Vol. 1* 3–10
sector 0: *Vol. 1* 3–25
unix: *Vol. 1* 3–6
recover diskette, copy: *Vol. 1* 2–6
reference diskette: *Vol. 1* 2–2
REL: *Vol. 3* 3–46, 4–116
release, parameter: *Vol. 3* 4–116
remote installation
overview: *Vol. 1* 1–47
setting up remote console: *Vol. 1* 1–49
setting up the equipment: *Vol. 1* 1–48
remote mounts, maximum parameter: *Vol. 3* 4–96
remote.unknown file: *Vol. 2* 9–57
remove
filter: *Vol. 2* 5–44
inactive file: *Vol. 3* 7–11
software packages: *Vol. 1* 1–31
repair, file system: *Vol. 1* 5–4
replace, failed disk: *Vol. 2* 1–35
repquota command: *Vol. 1* 4–22
request, hold: *Vol. 2* 5–5
request log: *Vol. 2* 5–46
restore
corrupted root file system: *Vol. 1* 7–21
entire disk: *Vol. 1* 7–10
file systems: *Vol. 1* 7–2
raw devices: *Vol. 1* 7–8

restoring, entire system: *Vol. 1* 7–20
restricted file ownership flag: *Vol. 3* 4–119
retension a cartridge tape: *Vol. 1* 6–26
RF_MAXKMEM: *Vol. 3* 3–37, 4–117
RFFHEAP: *Vol. 3* 3–37
RFS_VHIGH: *Vol. 3* 3–37
RFS_VLOW: *Vol. 3* 3–37
RIDEOUT: *Vol. 3* 3–39
root, recover password: *Vol. 1* 3–10
root file system
    parameter: *Vol. 3* 4–118
    restore corrupted: *Vol. 1* 7–21
root login: *Vol. 1* 1–2
root logins, from terminals: *Vol. 1* 1–3
ROOTFSTYPE: *Vol. 3* 4–118
RSTCHOWN: *Vol. 3* 3–30, 4–119
RTMAXPRI: *Vol. 3* 3–42
RTNPROCS: *Vol. 3* 3–42, 4–120
run level, change: *Vol. 1* 1–5
runacct command: *Vol. 3* 1–35

S

s5 file system type: *Vol. 1* A–10
S52KNBUF: *Vol. 3* 3–31
S52KNHBUF: *Vol. 3* 3–31
SAC. *See* service access controller
sacadm(1M): *Vol. 2* 3–5
sadc: *Vol. 3* 1–5
    output: *Vol. 3* 1–6
SAF. *See* service access facility
SANECNT: *Vol. 3* 3–39

SANITYCLK: *Vol. 3* 3–38
sar, analyze activity data: *Vol. 3* 1–7
SATENABLED: *Vol. 3* 3–45, 4–121
save dump area to media: *Vol. 1* 3–37
save memory to dump area: *Vol. 1* 3–35
scheduler parameters: *Vol. 3* 3–42
SCORLIM: *Vol. 3* 3–43, 4–122
SCPULIM: *Vol. 3* 3–43, 4–123
screen mapping, change: *Vol. 1* D–6
SCSI controller, add: *Vol. 2* 2–27, 2–34
SDATLIM: *Vol. 3* 3–43, 4–124
sector 0, recover: *Vol. 1* 3–25
Secure RPC Domain, parameter: *Vol. 3* 4–145
security audit trail, parameter: *Vol. 3* 4–121
SEGMAPSZ: *Vol. 3* 3–38, 4–125
SEMAEM: *Vol. 3* 3–35, 4–126
semaphore
    adjust–on–exit value: *Vol. 3* 4–126
    maximum in system: *Vol. 3* 4–129
    maximum operations per semop: *Vol. 3* 4–132
    maximum per semaphore identifier: *Vol. 3* 4–131
    maximum value parameter: *Vol. 3* 4–134
    number entries in control map: *Vol. 3* 4–127
SEMMAP: *Vol. 3* 3–35, 4–127
SEMMNI: *Vol. 3* 3–35, 4–128
SEMMNS: *Vol. 3* 3–35, 4–129
SEMMNU: *Vol. 3* 3–35, 4–130
SEMMSL: *Vol. 3* 3–35, 4–131
SEMOPM: *Vol. 3* 3–35, 4–132
semphore, maximum identifiers in kernel: *Vol. 3* 4–128
SEMUME: *Vol. 3* 3–35, 4–133

SEMVMX: *Vol. 3* 3–35, 4–134
send descriptors, maximum parameter: *Vol. 3* 4–95
sendmail
alias database: *Vol. 1* F–15
aliasing: *Vol. 1* E–14
alternate configuration file: *Vol. 1* F–20
arguments: *Vol. 1* E–13
arguments: *Vol. 1* F–19
building a configuration file: *Vol. 1* F–39
command line flags: *Vol. 1* F–48
configuration: *Vol. 1* E–16
configuration file: *Vol. 1* F–10, F–27
configuration options: *Vol. 1* F–50
daemon mode: *Vol. 1* F–19
debugging: *Vol. 1* F–20
delivery mode: *Vol. 1* F–24
design goals: *Vol. 1* E–5
file mode: *Vol. 1* F–25
forcing the queue: *Vol. 1* F–19
forwarding: *Vol. 1* E–14
forwarding files: *Vol. 1* F–18
header declarations: *Vol. 1* E–17
inclusion: *Vol. 1* E–15
installing: *Vol. 1* F–4
interfaces: *Vol. 1* E–8
limiting load: *Vol. 1* F–24
logging level: *Vol. 1* F–25
macros: *Vol. 1* E–17
mail queue: *Vol. 1* F–11
mailer declarations: *Vol. 1* E–17
mailer flags: *Vol. 1* F–55
message collection: *Vol. 1* E–15
message delivery: *Vol. 1* E–16
queue interval: *Vol. 1* F–19

queue priorities: *Vol. 1* F–23
queued messages: *Vol. 1* E–16
set options: *Vol. 1* E–18
system log: *Vol. 1* F–10
tuning: *Vol. 1* F–21
serial controller
configure modem on: *Vol. 2* 3–25
configure terminal on: *Vol. 2* 3–22
install: *Vol. 2* 2–18
serial number, parameter for: *Vol. 3* 4–26
serial port
configure a second: *Vol. 2* 3–20
configure modem on: *Vol. 2* 3–15
configure terminal on: *Vol. 2* 3–13
server
maximum handling remote requests: *Vol. 3* 4–49
minimum running on system: *Vol. 3* 4–60
service access controller: *Vol. 2* 3–3
administrative file: *Vol. 2* 3–4
service access facility: *Vol. 2* 3–2
set
node name for running kernel: *Vol. 2* 9–9
prioritiy limits: *Vol. 2* 5–3
quotas: *Vol. 1* 4–20
set up
link for BNU: *Vol. 2* 9–5
virtual terminals: *Vol. 1* 1–3
setuname command: *Vol. 2* 9–10
setup
equipment for remote installation: *Vol. 1* 1–48
remote console: *Vol. 1* 1–49
SFNOLIM: *Vol. 3* 3–43, 4–135
SFSZLIM: *Vol. 3* 3–43, 4–136

shadow file: *Vol. 1* 1–28
update: *Vol. 1* 1–29
shadowing, password: *Vol. 1* 1–28
shared data segment
attachments allowed: *Vol. 3* 4–165
number in system: *Vol. 3* 4–164
shared library, maximum attached to process: *Vol. 3* 4–137
shared memory
maximum attached segments parameter: *Vol. 3* 4–142
maximum identifiers parameter: *Vol. 3* 4–141
maximum segment size parameter: *Vol. 3* 4–139
minimum segment size parameter: *Vol. 3* 4–140
shell layer, sessions in progress parameter: *Vol. 3* 4–101
SHLBMAX: *Vol. 3* 3–30, 4–137
SHM_NAILED_GID, parameters: *Vol. 3* 4–138
SHMMAX: *Vol. 3* 3–36, 4–139
SHMMIN: *Vol. 3* 3–36, 4–140
SHMMNI: *Vol. 3* 3–36, 4–141
SHMSEG: *Vol. 3* 3–36, 4–142
shutdown: *Vol. 1* 1–5
shutdown system: *Vol. 1* 1–7
single user state: *Vol. 1* 1–5
size, check for BNU log files: *Vol. 2* 9–68, 9–70
slice
disk: *Vol. 2* 1–14
display information for hard disk: *Vol. 2* 1–23
SOCK_HIWATER: *Vol. 3* 4–143
SOCK_LOWATER: *Vol. 3* 4–144
sockmod module
high water mark: *Vol. 3* 4–143
low water mark: *Vol. 3* 4–144
sockmod module parameters: *Vol. 3* 3–34

soft limit parameter: *Vol. 3* 1–6
software packages: *Vol. 1* 1–30
    install from flex: *Vol. 1* 1–30
    install from tape: *Vol. 1* 1–30
    list: *Vol. 1* 1–30
    remove: *Vol. 1* 1–31
SPTMAP: *Vol. 3* 3–30
SRPC_DOMAIN: *Vol. 3* 3–46, 4–145
SSTKLIM: *Vol. 3* 3–43, 4–146
stack segment, maximum size parameter: *Vol. 3* 4–146
/stand
    compact files: *Vol. 1* 4–18
    recover files in: *Vol. 1* 3–2
start, LP Print Service: *Vol. 2* 4–39
start–of–day count: *Vol. 3* 7–10
startup system: *Vol. 1* 1–7
status
    display for printers: *Vol. 2* 4–37
    display for processors: *Vol. 1* 1–33
stop
    LP print service: *Vol. 2* 4–39
    printer fault alert: *Vol. 2* 5–9
strategy, powerfail: *Vol. 1* 1–37
STRCTLSZ: *Vol. 3* 3–33, 4–147
STRLOFRAC: *Vol. 3* 3–33
STRMEDFRAC: *Vol. 3* 3–33
STRMSGSZ: *Vol. 3* 3–33, 4–148
STRTHRESH: *Vol. 3* 3–33, 4–149
stty command: *Vol. 2* 4–31
su login: *Vol. 1* 1–2
support package: *Vol. 1* 1–13
SVMMLIM: *Vol. 3* 3–43, 4–150

swap command: *Vol. 2* 1–27
swap file, add: *Vol. 2* 1–27
swap sapce, configure: *Vol. 2* 1–25
symbolic links: *Vol. 1* A–5
SYS: *Vol. 3* 3–46, 4–151
sysadm, menu hierarchy: *Vol. 1* C–2
sysadm login: *Vol. 1* 1–2
system activity
    analyze with process accounting: *Vol. 3* 1–35
    analyze with sar: *Vol. 3* 1–7
system log, sendmail: *Vol. 1* F–10
system message headers, maximum parameter: *Vol. 3* 4–67
system name, parameter: *Vol. 3* 4–151
system profiling: *Vol. 3* 1–49
system release, parameter: *Vol. 3* 4–116
system shutdown: *Vol. 1* 1–7
system startup: *Vol. 1* 1–7
system state, change: *Vol. 1* 1–5
Systems file: *Vol. 2* 9–11

T

task entry, delete: *Vol. 1* C–31
template: *Vol. 2* 5–38
terminal
    configure on serial controller: *Vol. 2* 3–22
    configure on serial port: *Vol. 2* 3–13
    maximum xt supported parameter: *Vol. 3* 4–104
    virtual: *Vol. 1* 1–3
terminal line settings: *Vol. 2* 3–11
terminfo databse, adjust: *Vol. 2* 6–14
three–wire cable, configure serial port: *Vol. 2* 3–35

TIM_HIWATER: *Vol. 3* 4–152

TIM_LOWATER: *Vol. 3* 4–153

time command: *Vol. 3* 1–45

time slice, maximum value parameter: *Vol. 3* 4–50

time–sharing process, user priority parameter: *Vol. 3* 4–158

timer parameters: *Vol. 3* 3–42

timesharing process, number permitted: *Vol. 3* 4–159

timex command: *Vol. 3* 1–47

TIMEZONE: *Vol. 3* 3–38, 4–154

timod, maximum modes parameter: *Vol. 3* 4–102

timod module

- high water mark: *Vol. 3* 4–152
- low water mark: *Vol. 3* 4–153

timod module parameters: *Vol. 3* 3–34

timod module paramters: *Vol. 3* 3–34

tirdwr module

- high water mark: *Vol. 3* 4–156
- low water mark: *Vol. 3* 4–157

tirdwr module parameters: *Vol. 3* 3–34

tirdwr module paramters: *Vol. 3* 3–34

TRACESZ: *Vol. 3* 3–30, 4–155

troubleshooting, printer: *Vol. 2* 4–49

troubleshooting BNU

- administrative problems: *Vol. 2* 9–72
- faulty ACUs and modems: *Vol. 2* 9–72
- no login: *Vol. 2* 9–71
- out of space: *Vol. 2* 9–71

troubleshooting printers
character set or font wrong: *Vol. 2* 4–53
dialout failures: *Vol. 2* 4–54
idle printers: *Vol. 2* 4–55
illegible output: *Vol. 2* 4–50
network problems: *Vol. 2* 4–55
no output: *Vol. 2* 4–49
spacing: *Vol. 2* 4–52
troubleshooting with cu: *Vol. 2* 9–73
troubleshooting with uutry: *Vol. 2* 9–74
TRW_HIWATER: *Vol. 3* 4–156
TRW_LOWATER: *Vol. 3* 4–157
TSMAXUPRI: *Vol. 3* 3–42, 4–158
TSNPROCS: *Vol. 3* 3–42, 4–159
ttydefs: *Vol. 2* 3–11
ttymon: *Vol. 2* 3–8
log files: *Vol. 2* 3–10
terminal line settings: *Vol. 2* 3–11
tuning
basic steps: *Vol. 3* 5–3
bottlenecks: *Vol. 3* 5–10
reasons for: *Vol. 3* 5–2
sendmail: *Vol. 1* F–21
turn off, quotas: *Vol. 1* 4–22
turn on quotas: *Vol. 1* 4–22
type, identify for file system: *Vol. 1* 4–6
types, file system: *Vol. 1* A–8

U

ufs file system
check: *Vol. 1* 5–38
phases of fsck: *Vol. 1* 5–47

ufs file system type: *Vol. 1* A–17
UFSINODE: *Vol. 3* 3–31
UFSNINODE: *Vol. 3* 4–161
ULIMIT: *Vol. 3* 3–30, 4–162
umount command: *Vol. 1* 4–14
undeliverable BNU jobs, clean up: *Vol. 2* 9–68
undo structure
    maximum entries parameter: *Vol. 3* 4–133
    maximum parameter: *Vol. 3* 4–130
unix, recover: *Vol. 1* 3–6
unmount, file system: *Vol. 1* 4–13
usage of disk, monitor and control: *Vol. 3* 7–7
USANEON: *Vol. 3* 3–39
user
    access to forms: *Vol. 2* 5–16
    large space: *Vol. 3* 7–12
user diagnostics diskette: *Vol. 1* 2–6
uucheck command: *Vol. 2* 9–77
uucp logins, add: *Vol. 2* 9–20
uudemon.admin: *Vol. 2* 9–65
uudemon.cleanup: *Vol. 2* 9–65
uudemon.hour: *Vol. 2* 9–66
uudemon.poll: *Vol. 2* 9–66
uulog command: *Vol. 2* 9–77
uuname command: *Vol. 2* 9–77
uustat command: *Vol. 2* 9–68
uutry command: *Vol. 2* 9–74

V

VER: *Vol. 3* 3–46, 4–163
verify, BNU link: *Vol. 2* 9–63
version of OS release, parameter: *Vol. 3* 4–163
vfstab file: *Vol. 1* 4–4
VHNDFRAC: *Vol. 3* 3–32
virtual circuits, number open on network: *Vol. 3* 4–46
virtual terminal, number permitted parameter: *Vol. 3* 4–94
virtual terminals: *Vol. 1* 1–3
    accessing: *Vol. 1* 1–3
    closing: *Vol. 1* 1–4
    setting up: *Vol. 1* 1–3

W

write
    dialing information: *Vol. 2* 9–30
    help messages: *Vol. 1* C–8
    interface program: *Vol. 2* 6–19

X

XENIX semaphore, maximum parameter: *Vol. 3* 4–166
XSDSEGS: *Vol. 3* 3–38, 4–164
XSDSLOTS: *Vol. 3* 3–38, 4–165
XSEMMAX: *Vol. 3* 3–38, 4–166
xt terminals, maximum supported parameter: *Vol. 3* 4–104