

UNICOS® Configuration Administrator's Guide

Document Number 004-2303-002

Copyright © 1995, 1999 Silicon Graphics, Inc. All Rights Reserved. This document or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Silicon Graphics, Inc.

LIMITED AND RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in the Rights in Data clause at FAR 52.227-14 and/or in similar or successor clauses in the FAR, or in the DOD or NASA FAR Supplement. Unpublished rights reserved under the Copyright Laws of the United States. Contractor/manufacturer is Silicon Graphics, Inc., 2011 N. Shoreline Blvd., Mountain View, CA 94043-1389.

Autotasking, CF77, CRAY, Cray Ada, CraySoft, CRAY Y-MP, CRAY-1, CRInform, CRI/*TurboKiva*, HSX, LibSci, MPP Apprentice, SSD, SUPERCLUSTER, UNICOS, and X-MP EA are federally registered trademarks and Because no workstation is an island, CCI, CCMT, CF90, CFT, CFT2, CFT77, ConCurrent Maintenance Tools, COS, Cray Animation Theater, CRAY APP, CRAY C90, CRAY C90D, Cray C++ Compiling System, CrayDoc, CRAY EL, CRAY J90, CRAY J90se, CrayLink, Cray NQS, Cray/REELlibrarian, CRAY S-MP, CRAY SSD-T90, CRAY SV1, CRAY T90, CRAY T3D, CRAY T3E, CrayTutor, CRAY X-MP, CRAY XMS, CRAY-2, CSIM, CVT, Delivering the power . . ., DGauss, Docview, EMDS, GigaRing, HEXAR, IOS, ND Series Network Disk Array, Network Queuing Environment, Network Queuing Tools, OLNET, RQS, SEGLDR, SMARTE, SUPERLINK, System Maintenance and Remote Testing Environment, Trusted UNICOS, UNICOS MAX, and UNICOS/mk are trademarks of Cray Research, Inc., a wholly owned subsidiary of Silicon Graphics, Inc.

DECnet, VAX, and VAXBI are trademarks of Digital Equipment Corporation. IBM is a trademark and VM is a product of International Business Machines Corporation. Kerberos is a trademark of the Massachusetts Institute of Technology. NFS is a trademark of Sun Microsystems, Inc. NSC is a trademark of Network Systems Corporation. REELlibrarian is a trademark of Sceptre Corporation. SecurID is a trademark of Security Dynamics, Inc. UltraNet is a trademark of Computer Network Technology Corporation.

The UNICOS operating system is derived from UNIX® System V. The UNICOS operating system is also based in part on the Fourth Berkeley Software Distribution (BSD) under license from The Regents of the University of California.

What's New in This Guide

UNICOS® Configuration Administrator's Guide

004–2303–002

This rewrite supports the 10.0.0.4 release of the UNICOS operating system. It includes the following new configuration specification language parameters:

- `harp_entries`
- `harp_rcv`
- `harp_send`

Record of Revision

| <i>Version</i> | <i>Description</i> |
|-----------------------|--|
| 9.0 | May 1995 Original Printing. Documentation to support the UNICOS 9.0 release running on all Cray Research computer systems. This manual contains the contents of and supersedes the information formerly provided in the “Configuring UNICOS” section in the <i>UNICOS System Administration</i> , publication SG-2113 8.0 |
| 9.1 | November 1995 Revision to support the UNICOS 9.1 release. |
| 9.2 | December 1996 Revision to support the UNICOS 9.2 release. |
| 9.3 | May 1997 Revision to support the UNICOS 9.3 release. |
| 10.0 | October 1997 Revision to support the UNICOS 10.0 release. |
| 10.0.0.3 | September 1998 Revision to support the UNICOS 10.0.0.3 release. |
| 10.0.0.4 | January 1999 Revision to support the UNICOS 10.0.0.4 release. |

Contents

| | <i>Page</i> |
|---|-------------|
| Preface | ix |
| UNICOS System Administration Publications | ix |
| Related Publications | x |
| Obtaining Publications | xii |
| Conventions | xii |
| Reader Comments | xiv |
| Introduction [1] | 1 |
| sn.h File [2] | 3 |
| Required Steps | 4 |
| Optional Steps | 5 |
| config.h File [3] | 7 |
| config.mh File [4] | 27 |
| CSL Parameter File [5] | 33 |
| CSL Syntax | 33 |
| Identifiers | 33 |
| Constants | 34 |
| Operators, Separators, and Comments | 35 |
| CSL Usage | 35 |
| The Boot Process | 35 |
| Configuration Verification | 35 |
| 004-2303-002 | iii |

| | <i>Page</i> |
|---|-------------|
| Error Message Syntax | 36 |
| CSL Parameter File Sections | 36 |
| dumpinfo Section | 37 |
| gigaring Section | 39 |
| gr_route Subsection | 39 |
| gr_union Subsection | 40 |
| ios_e Section | 41 |
| IOP Declarations | 42 |
| HISP Declarations | 43 |
| IOP Boot Declarations | 43 |
| Example of ios_e Section | 43 |
| mainframe Section | 44 |
| Common Parameters | 45 |
| Multi-streaming Processors (CRAY SV1 GigaRing Based Systems Only) | 46 |
| Parameters for GigaRing Based Systems Only | 47 |
| Parameters for Model E Based Systems Only | 49 |
| Example of the mainframe Section for a GigaRing Based System | 50 |
| Example of the mainframe Section for a Model E Based System | 50 |
| unicos Section | 50 |
| Example for a GigaRing Based System | 54 |
| Additional Parameters for a CRAY SSD-T90 System | 54 |
| Example for a Model E Based System | 55 |
| filesystem Section | 55 |
| Physical Device Definition | 56 |
| Device Node Definition | 66 |
| CRAY SSD-T90 Description | 68 |
| Root, Swap, and Secondary Data Segment (SDS) Devices | 69 |
| Example of the filesystem Section Containing a RAM File System | 69 |
| Example of the filesystem Section for a GigaRing Based System | 70 |

| | <i>Page</i> |
|---|-------------|
| Example of the <code>filesystem</code> Section for a GigaRing Based System with Disk Devices Configured for Third-party I/O | 71 |
| Example of the <code>filesystem</code> Section for a CRAY SSD-T90 Device | 72 |
| Example of the <code>filesystem</code> Section for Model E Based Systems | 72 |
| Example of the <code>filesystem</code> Section Containing an SSD for Model E Based Systems | 74 |
| <code>network</code> Section | 74 |
| Network Parameters | 75 |
| Customized Network Device Prototypes for Model E Based Systems | 79 |
| <code>device type</code> for Model E Based Systems | 80 |
| Device Types | 82 |
| Device Formats | 83 |
| <code>logical path</code> for Model E Based Systems | 87 |
| <code>direction</code> Argument for Model E Based Systems | 88 |
| <code>device type</code> Statement for Model E Based Systems | 88 |
| <code>np_spec</code> Statement for Model E Based Systems | 88 |
| <code>network</code> Section Example Common to All Systems | 89 |
| <code>network</code> Section Example for GigaRing Based Systems | 90 |
| <code>network</code> Section Example for Model E Based Systems | 90 |
| <code>network</code> Section Example for Model V Based Systems | 92 |
| <code>revision</code> Section | 93 |
| Run-time Configuration Scripts and Files [6] | 95 |
| <code>at.deny</code> and <code>at.allow</code> Files | 96 |
| <code>bcheckrc</code> Script | 96 |
| <code>brc</code> and <code>coredd</code> Scripts | 97 |
| <code>cron.deny</code> and <code>cron.allow</code> Files | 97 |
| <code>cshrc</code> File | 97 |
| <code>daemons</code> File | 98 |
| <code>fstab</code> File | 99 |

| | <i>Page</i> |
|---|-------------|
| gettydefs File | 100 |
| ghippiX.arp File | 101 |
| grX.arp File | 101 |
| group File | 101 |
| hycf.local_network Files | 102 |
| inittab File | 102 |
| interfaces File | 103 |
| issue File | 103 |
| ldchlist File | 104 |
| motd File | 104 |
| netstart Script | 104 |
| netvar.conf File | 105 |
| passwd File | 105 |
| profile File | 106 |
| rc Script and rcoptions File | 106 |
| shutdown Script | 107 |
| umountem Script | 108 |
| Appendix A Reserved Keywords in CSL | 109 |
| Index | 117 |
| Tables | |
| Table 1. Kernel Parameters in config.h (Common) | 8 |
| Table 2. Kernel Parameters in config.h (Common) Not in ICMS | 11 |
| Table 3. Kernel Parameters in config.h (CRAY J90 Systems Only) | 12 |
| Table 4. Dynamic Kernel Memory Parameters | 12 |
| Table 5. Shared Memory Kernel Parameters in config.h | 14 |
| Table 6. Kernel Parameters in config.h for Process Error Thresholds | 15 |
| Table 7. IPC Kernel Parameters in config.h | 15 |

| | <i>Page</i> |
|--|-------------|
| Table 8. MLS Kernel Parameters in config.h | 17 |
| Table 9. TCP/IP Kernel Parameter in config.h | 23 |
| Table 10. NFS Kernel Parameters in config.h | 23 |
| Table 11. CSL Boot-time Equivalents for NFS Kernel Parameters | 25 |
| Table 12. General System Parameters (Common) | 27 |
| Table 13. General System Parameters (GigaRing Based Systems Only) | 28 |
| Table 14. General System Parameters (Model E Based Systems Only) | 29 |
| Table 15. Kernel Generation Parameters | 29 |
| Table 16. Kernel Subsystem Parameters | 31 |
| Table 17. Programming Environment Parameters | 32 |
| Table 18. CPU/MSP Correlation | 46 |
| Table 19. I/O Module 0 Channels | 48 |
| Table 20. Example of Complete Channel Assignments | 49 |
| Table 21. Online Tape Parameters | 51 |
| Table 22. Table Size Parameters | 51 |
| Table 23. Maximum Limits Parameters | 51 |
| Table 24. Disk Parameters (Common) | 52 |
| Table 25. Disk Parameters (GigaRing Based Systems Only) | 53 |
| Table 26. Disk Parameters (Model E Based Systems Only) | 53 |
| Table 27. ION Unit Bit and Range Numbers (GigaRing Based Systems Only) | 54 |
| Table 28. Disk Type and Name | 56 |
| Table 29. Start and Length Units for Physical Storage Device Types | 58 |
| Table 30. Disk Information (GigaRing Based Systems Only) | 59 |
| Table 31. Target Memory Type Values | 68 |
| Table 32. Network Parameter Values (Common) | 75 |
| Table 33. Network Parameter Values (GigaRing Based Systems) | 77 |
| Table 34. Network Parameter Values (Common to Model E and Model V Based Systems) | 78 |
| Table 35. Network Parameter Values (Model V Based Systems) | 78 |

| | <i>Page</i> |
|--|-------------|
| Table 36. Network Parameter Values (Model E Based Systems) | 78 |
| Table 37. Standard Interface Configuration Templates | 81 |
| Table 38. Network Device Types (GigaRing Based Systems) | 83 |
| Table 39. Network Device Type (Model E Based Systems) | 83 |
| Table 40. Network Device Type (Model V Based Systems) | 83 |

This publication provides information about the UNICOS configuration files created when the UNICOS 10.0 operating system is installed and configured.

This manual documents UNICOS release 10.0 running on Cray Research systems. It contains information needed in the administration of various UNICOS features available to all UNICOS systems.



Warning: Starting with the UNICOS 10.0 release, the term *Cray ML-Safe* replaces the term *Trusted UNICOS*, which referred to the system configuration used to achieve the UNICOS 8.0.2 release evaluation. Because of changes to available software, hardware, and system configurations since the UNICOS 8.0.2 system release, the term *Cray ML-Safe* does not imply an evaluated product, but refers to the currently available system configuration that closely resembles that of the evaluated Trusted UNICOS 8.0.2 system.

For the UNICOS 10.0 release, the functionality of the Trusted UNICOS system has been retained, but the `CONFIG_TRUSTED` option, which enforces conformance to the strict B1 configuration, is no longer available.

UNICOS System Administration Publications

Information on the structure and operation of a Cray Research computer system running the UNICOS operating system, as well as information on administering various products that run under the UNICOS operating system, is contained in the following documents:

- *General UNICOS System Administration*, contains information on performing basic administration tasks as well as information about system and security administration using the UNICOS multilevel (MLS) feature. This publication contains chapters documenting file system planning, UNICOS startup and shutdown procedures, file system maintenance, basic administration tools, crash and dump analysis, the UNICOS multilevel security (MLS) feature, and administration of online features.
- *UNICOS Resource Administration*, contains information on the administration of various UNICOS features available to all UNICOS systems. This publication contains chapters documenting accounting, automatic incident reporting (AIR), the fair-share scheduler, file system quotas, file system monitoring, system activity and performance monitoring, and the Unified Resource Manager (URM).

- *UNICOS Configuration Administrator's Guide*, provides information about the UNICOS kernel configuration files and the run-time configuration files and scripts.
- *UNICOS Networking Facilities Administrator's Guide*, contains information on administration of networking facilities supported by the UNICOS operating system. This publication contains chapters documenting TCP/IP for the UNICOS operating system, the UNICOS network file system (NFS) feature, and the network information system (NIS) feature.
- *NQE Administration*, describes how to configure, monitor, and control the Cray Network Queuing Environment (NQE) running on a UNIX system.
- *Kerberos Administrator's Guide*, contains information on administration of the Kerberos feature, a set of programs and libraries that provide distributed authentication over an open network. This publication contains chapters documenting Kerberos implementation, configuration, and troubleshooting.
- *Tape Subsystem Administration*, contains information on administration of UNICOS and UNICOS/mk tape subsystems. This publication contains chapters documenting tape subsystem administration commands, tape configuration, administration issues, and tape troubleshooting.

Related Publications

The following publications contain additional information that may be helpful:

The following man page manuals contain additional information that may be helpful.

Note: For the UNICOS 10.0 release, man page reference manuals are not orderable in printed book form. Instead, they are available as printable PostScript files provided on the same DynaWeb CD as the rest of the supporting documents for this release. Individual man pages are still available online and can be accessed by using the `man(1)` command.

- *UNICOS User Commands Reference Manual*
- *UNICOS System Calls Reference Manual*
- *UNICOS File Formats and Special Files Reference Manual*
- *UNICOS Administrator Commands Reference Manual*
- *UNICOS System Libraries Reference Manual*

The following ready references are available in printed form from the Distribution Center:

- *UNICOS User Commands Ready Reference*
- *UNICOS System Libraries Ready Reference*
- *Asynchronous Transfer Mode (ATM) Administrator's Guide*
- *UNICOS System Calls Ready Reference*
- *UNICOS Administrator Commands Ready Reference*

Design specifications for the UNICOS multilevel security (MLS) feature are based on the trusted computer system evaluation criteria developed by the U. S. Department of Defense (DoD). If you require more information about multilevel security on UNICOS, you may find the following sources helpful:

- DoD Computer Security Center. *A Guide to Understanding Trusted Facility Management* (DoD NCSC-TG-015). Fort George G. Meade, Maryland: 1989.
- DoD Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria* (DoD 5200.28-STD). Fort George G. Meade, Maryland: 1985. (Also known as the *Orange book*.)
- DoD Computer Security Center. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria* (DoD NCSC-TG-005-STD). Fort George G. Meade, Maryland: 1987. (Also known as the *Red book*.)
- DoD Computer Security Center. *Summary of Changes, Memorandum for the Record* (DoD 5200.28-STD). Fort George G. Meade, Maryland: 1986.
- DoD Computer Security Center. *Password Management Guidelines* (CSC-STD-002-85). Fort George G. Meade, Maryland: 1985.
- Wood, Patrick H. and Stephen G. Kochan. *UNIX System Security*. Hasbrouck Heights, N.J.: Hayden Book Company, 1985.

Note: If your site wants to purchase the optional SecurID card used with UNICOS MLS network security, the necessary hardware, software, and user publications can be obtained from Security Dynamics, Inc., 2067 Massachusetts Avenue, Cambridge, MA, 02140, (617) 547-7820.

Obtaining Publications

The *User Publications Catalog* describes the availability and content of all Cray Research hardware and software documents that are available to customers. Customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, call +1 651 683 5907. Silicon Graphics employees may send electronic mail to `orderdisk@sgi.com` (UNIX system users).

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

Conventions

The following conventions are used throughout this document:

| <u>Convention</u> | <u>Meaning</u> | | | | | | | | | | | | | | | | | | |
|-------------------------|--|---|---------------|----|-------------------------------|---|--------------|---|--------------------------------------|---|-------------------------|----|-----------|---|--------------|---|----------------------|----|-------------------------|
| <code>command</code> | This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures. | | | | | | | | | | | | | | | | | | |
| <code>manpage(x)</code> | Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: <table> <tbody> <tr> <td>1</td> <td>User commands</td> </tr> <tr> <td>1B</td> <td>User commands ported from BSD</td> </tr> <tr> <td>2</td> <td>System calls</td> </tr> <tr> <td>3</td> <td>Library routines, macros, and opdefs</td> </tr> <tr> <td>4</td> <td>Devices (special files)</td> </tr> <tr> <td>4P</td> <td>Protocols</td> </tr> <tr> <td>5</td> <td>File formats</td> </tr> <tr> <td>7</td> <td>Miscellaneous topics</td> </tr> <tr> <td>7D</td> <td>DWB-related information</td> </tr> </tbody> </table> | 1 | User commands | 1B | User commands ported from BSD | 2 | System calls | 3 | Library routines, macros, and opdefs | 4 | Devices (special files) | 4P | Protocols | 5 | File formats | 7 | Miscellaneous topics | 7D | DWB-related information |
| 1 | User commands | | | | | | | | | | | | | | | | | | |
| 1B | User commands ported from BSD | | | | | | | | | | | | | | | | | | |
| 2 | System calls | | | | | | | | | | | | | | | | | | |
| 3 | Library routines, macros, and opdefs | | | | | | | | | | | | | | | | | | |
| 4 | Devices (special files) | | | | | | | | | | | | | | | | | | |
| 4P | Protocols | | | | | | | | | | | | | | | | | | |
| 5 | File formats | | | | | | | | | | | | | | | | | | |
| 7 | Miscellaneous topics | | | | | | | | | | | | | | | | | | |
| 7D | DWB-related information | | | | | | | | | | | | | | | | | | |

8 Administrator commands

Some internal routines (for example, the `_assign_asgcmd_info()` routine) do not have man pages associated with them.

variable

Italic typeface denotes variable entries and words or concepts being defined.

user input

This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.

[]

Brackets enclose optional portions of a command or directive line.

...

Ellipses indicate that a preceding element can be repeated.

The following machine naming conventions may be used throughout this document:

Term

Definition

Cray PVP systems

All configurations of Cray parallel vector processing (PVP) systems.

Cray MPP systems

All configurations of the CRAY T3D series. The UNICOS operating system is not supported on CRAY T3E systems. CRAY T3E systems run the UNICOS/mk operating system.

The default shell in the UNICOS and UNICOS/mk operating systems, referred to in Cray Research documentation as the *standard shell*, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2-1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS and UNICOS/mk operating systems also support the optional use of the C shell.

Cray UNICOS Version 10.0 is an X/Open Base 95 branded product.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the title and part number of the document with your comments.

You can contact us in any of the following ways:

- Send electronic mail to the following address:

`techpubs@sgi.com`

- Send a facsimile to the attention of “Technical Publications” at fax number +1 650 932 0801.
- Use the Suggestion Box form on the Technical Publications Library World Wide Web page:

`http://techpubs.sgi.com/library/`

- Call the Technical Publications Group, through the Technical Assistance Center, using one of the following numbers:

For Silicon Graphics IRIX based operating systems: 1 800 800 4SGI

For UNICOS or UNICOS/mk based operating systems or CRAY Origin2000 systems: 1 800 950 2729 (toll free from the United States and Canada) or +1 651 683 5600

- Send mail to the following address:

Technical Publications
Silicon Graphics, Inc.
2011 North Shoreline Boulevard, M/S 535
Mountain View, California 94043-1389

We value your comments and will respond to them promptly.

Introduction [1]

Properly configuring a system is very important and should be done carefully to ensure the efficient performance of your UNICOS computer system. *UNICOS System Configuration Using ICMS*, and the online help files associated with the menu system provide the information you need to configure the UNICOS operating system. This manual provides additional details about the UNICOS kernel configuration files and the run-time configuration files and scripts.

Note: It is strongly recommended that you use the UNICOS installation and configuration menu system (ICMS) to maintain your system configuration.

UNICOS kernel configuration information appears in the following files:

- `sn.h`, which contains parameters that define machine-specific characteristics of your mainframe.
- `config.h`, which contains parameters that define the UNICOS kernel.
- `config.mh`, which contains parameters used to include and exclude kernel subsystems.
- Configuration specification language (CSL) parameter file, which contains statements that specify hardware and software characteristics for your system.

Note: As of the UNICOS 9.2 release, ICMS is not used for installation but is used to maintain configuration.

The default values presented within this document are not necessarily the values a site must use. It is up to the system support staff to determine optimal values for the site. If help is needed, call your local Cray Research service representative.

Note: The documentation that applies to Model E based systems also applies to the CRAY J90 and CRAY J90se Model V based systems unless otherwise stated.

The *Trusted UNICOS* system is a configuration of the UNICOS MLS system that supports processing at multiple security labels and system administration using only nonsuper-user administrative roles. The Trusted UNICOS system consists of the subset of UNICOS software that offers these capabilities. The Trusted UNICOS name does not imply maintenance of the UNICOS 8.0.2 security evaluation.

sn.h File [2]

Note: Typically, the file is stored in `/usr/src/uts/cf.xxxx/sn.h`. It is strongly recommended that you use the UNICOS installation and configuration menu system (ICMS) to maintain your system configuration, rather than manually editing this file. (For more information on the ICMS, see the online help files and the *UNICOS System Configuration Using ICMS*.) If special circumstances require that you set the parameters in `sn.h` manually, use the procedures in this chapter.

The `sn.h` file contains parameters that define machine-specific characteristics of your mainframe. You must change some of these parameters to reflect your system's characteristics.

You will set the parameters by using the following menu:

```
Configure System
  ->Mainframe hardware configuration
```

If you manually edit `sn.h`, you must define the following parameters:

- Mainframe serial number
- Main memory size
- Number of banks of main memory
- Number of bits per chip in main memory

If your system is not fully configured with CPUs, you should also define the following parameters:

- Number of CPUs
- Number of mainframe clusters

Appropriate defaults for the remaining parameters in the `sn.h` file are set automatically, based upon your machine's serial number. You will not need to set these parameters.

2.1 Required Steps

The following steps are required if you edit `sn.h` manually:

1. Set your mainframe serial number by defining the `SN` parameter to be that value.

Example:

```
#define SN          4025
```

2. Set the physical memory size parameter (`MEMORY`) in decimal words to the highest addressable word.

Example:

```
#define MEMORY      512*MEGAWD-1
```

3. Set the `NBANKS` parameter to the number of memory banks in your mainframe:

Example:

```
#define NBANKS      1024
```

4. Set the number of bits per memory chip used in main memory by defining the `CHIPSZ` parameter with one of the following values:

| <u>Bits/chip</u> | <u>CHIPSZ value</u> |
|------------------|---------------------|
| 131072 | M128KCH |
| 262144 | M256KCH |
| 524288 | M512KCH |
| 1048576 | M1MCH |
| 2097152 | M2MCH |
| 4194304 | M4MCH |
| 8388608 | M8MCH |
| 16777216 | M16MCH |

Example:

```
#define CHIPSZ      M1MCH
```

2.2 Optional Steps

The following steps are optional. Appropriate default values are assigned according to the mainframe serial number.

1. Set the mainframe type parameter (`MFTYPE`) to one of the following values.

| <u>Option</u> | <u>Description</u> |
|----------------------|------------------------------------|
| <code>CRAY_TS</code> | CRAY T90 |
| <code>CRAYC90</code> | CRAY C90 |
| <code>CRAYYMP</code> | CRAY SV1, CRAY J90, and CRAY J90se |

Example:

```
#define MFTYPE      CRAYC90
```

2. Set the mainframe subtype (`MFSUBTYP`) parameter to the appropriate option.

| <u>Option</u> | <u>Description</u> |
|---------------------|--|
| <code>C900XX</code> | CRAY C916 |
| <code>C92AXX</code> | CRAY C92A |
| <code>D92AXX</code> | CRAY C92AM |
| <code>C94AXX</code> | CRAY C94A |
| <code>D940XX</code> | CRAY C94M |
| <code>C940XX</code> | CRAY C94 |
| <code>C980XX</code> | CRAY C98 |
| <code>D980XX</code> | CRAY C98M |
| <code>YMPJ90</code> | CRAY Y-MP architecture, CRAY SV1, CRAY J90 and CRAY J90se series |
| <code>T4XXX</code> | CRAY T94 |
| <code>T16XXX</code> | CRAY T916 |
| <code>T32XXX</code> | CRAY T932 |

Example:

```
#define MFSUBTYP   C900XX
```

3. If your system is not fully configured, define the number of CPUs (NCPU) and cluster registers (MAXCLUS).

Example:

```
#define NCPU      8
#define MAXCLUS  9
```

4. Set the clock period to the frequency in hertz (cycles per second) by defining HZ; the default values are based on MFSUBTYP and mainframe serial number.

Example:

```
#define HZ      HZ_416
```


config.h File [3]

Note: It is strongly recommended that you use the UNICOS installation and configuration menu system (ICMS) to maintain your system configuration, rather than manually editing this file. (For more information on the ICMS, see the online help files associated with the tool and *UNICOS System Configuration Using ICMS*.) Typically, this file is stored in `/usr/src/uts/cf.xxxx/config.h`.

This chapter summarizes parameters found in the `/usr/src/uts/cl/cf/config.h` file.

Note: The default values shown reflect the settings used for initial installations; appropriate values for upgrades will be changed automatically as part of the upgrade conversion process. Sites performing an upgrade should only need to change values manually if they are enabling features or changing hardware.

These parameters pertain to general configuration, the UNICOS multilevel security (MLS) feature, and TCP/IP. To set the general configuration and TCP/IP parameters, use the following menu:

```
Configure System
->Kernel Configuration
```

To set the MLS parameters, use the following menu:

```
Configure System
->Multilevel Security (MLS) Configuration
```

In general, a value of 0 disables a parameter and a value of 1 enables it.

Table 1 through Table 3 summarize resource and configuration parameters for the UNICOS kernel. Some parameters are not available in the ICMS and may therefore be edited only by those sites with a source license.

Table 1. Kernel Parameters in `config.h` (Common)

| Parameter | Default value | Description |
|--------------|---------------|---|
| CDLIMIT | 010000000000 | Maximum file size in blocks (see <code>ulimit(2)</code>). |
| DMODE | 1 | Default offline file retrieval mode for sites running data migration. 1 indicates automatic retrieval; 0 indicates manual-only retrieval. |
| EXTDCORE | 0 | Extended core file naming. |
| FLUSHONPANIC | 1 | The capability to flush buffered data to target I/O devices before the system is halted. |
| LDCHCORE | 0 | Memory clicks reserved for ldcache blocks. These are used for ldcache blocks specified as type <code>MEM</code> . This value can be changed at boot time in the CSL parameter file. |
| LINK_MAX | 1000 | Maximum number of directory entries and maximum number of links to one file. |
| MAXASYN | 192 | Maximum number of asynchronous I/O structures allowed per process. <code>MAXASYN</code> has considerable performance impact on applications utilizing asynchronous I/O (for example, <code>listio</code> and <code>aqio</code> system calls). |
| MAXPIPE | 20 | Maximum number of blocks allocated per pipe on the pipe device. |
| MAXRAH | 8 | Maximum number of read-ahead blocks per read operation. |
| NASYN | 800 | Number of asynchronous I/O headers. |
| NBLK_FCTR | 20 | Factor for maximum request size to/from system buffer cache. The default of 20 means that 1/20th (5%) of the system buffer cache can be allocated to a single request. |
| NBUF_FCTR | 20 | Number of cache blocks based on memory size. The default of 20 means that 1/20th (5%) of the memory is assigned to buffer cache. This can be changed at boot time to an absolute number of cache blocks with the <code>NBUF</code> parameter in the CSL parameter file. |

| Parameter | Default value | Description |
|------------|---------------|--|
| NCALL | NPROC | Number of callout table entries. |
| NCLIST | 1000 | Maximum number of <code>clists</code> . These are terminal I/O buffers. |
| NCRBUF | 4 | Maximum number of checkpoint and restart buffers. |
| NC1INODE | NINODE | Number of in-core inodes for NC1FS (file system migration parameter). Must be equal to <code>NINODE</code> . This parameter is not in the ICMS. |
| NC_NAMLEN | 15 | Maximum size of path-name component that will be cached in the directory name look-up cache (DNLC). |
| NC_SIZE | 1024 | Number of entries in directory name look-up cache. This should be less than <code>NINODE</code> . |
| NEXECS | 4 | Maximum number of parallel <code>exec</code> operations with maximum arguments. This parameter is not in ICMS. |
| NFILE | 2100 | Maximum number of files that can be open at one time system-wide. |
| NFLOCKS | 100 | Number of file-lock regions. |
| NHBUF_FCTR | 4 | Number of hash entries in the hash table for system cache, based on number of blocks assigned to system cache. The default of 4 specifies 1 hash list for every 4 system buffers. |
| NINODE | 1500 | Number of unique files that can be open at one time. Also the size of the in-core inode table. |
| NLDCH | 0 or 2000 | Number of SSD logical device cache headers, 0 for CRAY SV1 and CRAY J90 systems and 2000 for all other supported mainframes. This can be changed at boot time in the CSL parameter file. |
| NLDMAP | 250 | Number of file systems open concurrently. |
| NMNT | 250 | Number of slice structures. |
| NMOUNT | 75 | Number of file systems (local, NFS, DFS, and so on) that can be mounted. |

| Parameter | Default value | Description |
|------------------|---------------|---|
| NPBUF | 500 | Number of physical I/O headers. This can be changed at boot time in the CSL parameter file. |
| NPLCHCTL | 0 | Number of partition cache headers. This can be changed at boot time in the CSL parameter file. This parameter is necessary only if you specify the <code>-M CMEM</code> option of the <code>pcache(8)</code> command. |
| NPROC | 650 | Maximum number of processes that may be active simultaneously. |
| NPTY | 128 | Number of pseudo terminals (tty/pty pairs). This is also the maximum number of active terminal sessions. |
| NQUOTA | 1400 or 0 | If your site is running file system quotas, 1400 is the default number of in-core quota entries; otherwise, it is set to 0. |
| NSESS | 300 | Maximum number of sessions open simultaneously. |
| NSIDEBUF | 48 | Number of blocks for the SSD side door buffer. All sites with an SSD should have a side door buffer assigned. |
| NSU_ASYN | 4 | Number of asynchronous I/O headers reserved for the system. |
| NTEXT | 100 | Maximum number of shared-text programs that can be running simultaneously. |
| NUSERS | 200 | Maximum number of users defined for the fair share scheduler. |
| PLCHCORE | 0 | Memory clicks reserved for partition cache blocks. These are used for partition cache blocks specified as type <code>MEM</code> . This can be changed at boot time in the CSL parameter file. |
| TAPE_MAX_PER_DEV | 65536 | Maximum number of bytes per device for buffered I/O. |

| Parameter | Default value | Description |
|------------------|---------------|--|
| TAPE_MAX_CONF_UP | 4 | Maximum number of tapes that may be configured up. |
| XTRASEC | 3 | Number of CPU seconds that a process or session may use following receipt of a SIGCPULIM signal before the SIGKILL signal is sent to terminate the process or session. |

Table 2. Kernel Parameters in config.h (Common) Not in ICMS


| Parameter | Default value | Description |
|--------------------|---|---|
| ACNICE | HZ | The lowest process nice value is saved as <code>pc_acnice</code> after the specified number of seconds of CPU time. |
| FSLG_BUFSIZE | 400 | Number of file system log records. |
| K_OPEN_MAX | 16384 | Maximum number of open files per process. |
| KSTACKSIZE | 1500 or 2000 | Kernel stack size, 2000 if you have the DCE Distributed File Service (DFS) installed, 1500 if you do not. |
| |  | Caution: Do not change this value. This parameter does not appear in ICMS because it should not be changed. |
| MAX_UNLINKED_BYTES | 26214400 | Maximum bytes allowed in unlinked files for checkpoint and restart. |
| NC1MINRAW | 20 | Multiplier for automatic mixed buffer. |
| NC1INODE | NINODE | Number of in-core inodes for NC1FS (file system migration parameter). Must be equal to NINODE. |
| NEXECS | 4 | Maximum number of parallel <code>exec</code> operations with maximum arguments. |
| U_MAXPACK | 16 | Maximum number of outstanding <code>user</code> packets allowed. For CRAY SV1, CRAY J90, and CRAY J90se systems only. |

Table 3. Kernel Parameters in `config.h` (CRAY J90 Systems Only)

| Parameter | Default value | Description |
|-----------|---------------|---|
| NMT | 16 | Number of non-tape daemon tapes. |
| U_MAXDEV | 16 | Number of user devices. |
| U_MAXPACK | 16 | Number of outstanding user packets allowed. |

Table 4 lists the dynamic kernel memory parameters.

Table 4. Dynamic Kernel Memory Parameters

| Parameter | Default value | Description |
|-----------------|--|---|
| KM_CHM_PADSZ | $(((KSTACKSIZE + KM_WPU - 1) \gg KM_WPU_SHIFT) * (NPROC / 2))$ | Size of the memory padding. The total size (in KM_UNITS) must be large enough for NPROC/2 kernel stack entries. Compilation of <code>lowmem.c</code> will fail if this constraint is not met. |
| KM_EXPAND_UNITS | 128 | The number of units each expansion will add. Because expansion space is acquired from <code>coremap</code> , it must be a multiple of <code>coremap</code> units. That is: $(KM_EXPAND_UNITS * KM_WPU) == \text{multiple of } (MEMKLIK * NWPC)$ |
| KM_NO_THRASH | $(4 * KM_EXPAND_UNITS)$ | The number of units that must be reached before a contraction will take place. This prevents contractions from reclaiming expansion space that may be requested within a short time span. Expansions can be expensive if they require shuffling memory. |


| Parameter | Default value | Description |
|----------------------------|---------------|---|
| | | <p>An expansion may require moving the bitmap, which is stored in dynamic kernel memory, in order to increase its capacity. The frequency of such moves is dependent upon the values of <code>KM_WPU</code> and <code>KM_EXPAND_UNITS</code>.</p> <p>For example, if <code>KM_EXPAND_UNITS</code> is 512, each expansion requires 8 additional words of bitmap memory to manage the space. If <code>KM_WPU</code> is 16, this will cause the bitmap to be reallocated and moved every two expansions. If <code>KM_WPU</code> is 128, the bitmap will need to be reallocated every 8 expansions.</p> <p>In a given configuration of 640 initial units, 128 words per unit, and 128 units acquired with each expansion, the bitmap will not move until the 120th expansion; that is, 640 units divided by 64 bits per word results in 10 words required being initially required for the bitmap. However, since the bitmap is also allocated in an area of memory equal to the value of <code>KM_WPU</code>, 128 words are reserved for the bitmap. This leaves 118 extra words in the bitmap unit. Since each expand of 128 units requires an additional 2 words, 59 expands can be done before the bitmap capacity is exceeded and the bitmap will be grown (and moved if necessary).</p> |
| <code>KM_UNITS</code> | 1408 | <p>Initial dynamic kernel memory units from <code>coremap</code>.</p> <p> Caution: The value of $(\text{KM_UNITS} * \text{KM_WPU})$ should be a multiple of $(\text{MEMKLIK} * \text{NWPC})$ to avoid wasting space acquired from <code>coremap</code>. That is, at boot time, a chunk of memory is acquired from <code>coremap</code> to be managed at a finer granularity.</p> |
| <code>KM_WPU</code> | 128 | Words per unit. This value must be a power of 2. |
| <code>KM_WPU_SHIFTC</code> | 7 | Number of shifts to multiply or divide by the <code>KM_WPU</code> value. |

Table 5 lists the parameters in `config.h` for the CRAY T90 series shared memory feature configuration. These parameters are significant only for the CRAY T90 series.

Table 5. Shared Memory Kernel Parameters in `config.h`

| Parameter | Default value | Description |
|-----------|---------------|---|
| SHMMAX | 128 | Maximum size in clicks of a shared segment. Because memory is allocated in <code>MEMKLIK</code> units, this should be a multiple of <code>MEMKLIK</code> . This value must be greater than or equal to 0. |
| SHMMIN | 1 | Minimum size in clicks of a shared segment. This value must be greater than or equal to and less than the <code>SHMMAX</code> value. |
| SHMMNI | 20 | Maximum number of shared memory identifiers available concurrently in the system (that is, the size of the <code>shmem</code> table). This value must be greater than 0. |
| SHMSEG | 2 | Maximum number of shared segments that a process can have attached simultaneously. This value must be 0, 1, or 2. (The upper bound is a hardware restriction.) |

Table 6 lists the parameters in `config.h` that are used for setting process error thresholds. These values represent the number of allowable errors of each type (program range, operand range, and error exit) per connect to a CPU. The counters are reset on each pass through `resume()`. The default value is 500 for all systems.

If the values are set too high, the counts may never be reached (especially on a busy system), and the failing process will not be aborted. The definition of unreachable will vary according to application and system load. Setting the values to 0 disables this feature.

Table 6. Kernel Parameters in `config.h` for Process Error Thresholds

| Parameter | Default value | Description |
|------------|---------------|---|
| MAXUSRERR | 500 | Maximum number of error exits that may be encountered before the process is aborted. Setting the value to disables this feature. |
| MAXUSRORRE | 500 | Maximum number of operand range errors that may be encountered before the process is aborted. Setting the value to disables this feature. |
| MAXUSRPRE | 500 | Maximum number of program range errors that may be encountered before the process is aborted. Setting the value to 0 disables this feature. |

Table 7 lists the parameters in `config.h` that are used for interprocess (IPC) semaphore defines and message defines.

Table 7. IPC Kernel Parameters in `config.h`

| Parameter | Default value | Description |
|-----------|---------------|---|
| MSGMAX | 2048 | Maximum size of a message in bytes. This value must be greater than 0 and less than the MSGMNB value. |
| MSGMNB | 4096 | Maximum number of bytes that may be in a message queue. This value must be greater than and less than 999999. |
| MSGMNI | 40 | Maximum number of message queues that may be in the system at one time (that is, the size of the <code>msgque</code> table). This value must be greater than 0. |
| MSGSEG | 1024 | Number of message segments. The size of the message area is <code>MSGSEG * MSGSSZ</code> bytes. (This will be rounded up to <code>MEMKLIK</code> .) This value must be greater than or equal to the <code>MSGTQL</code> value. The upper limit is 9999. |

| Parameter | Default value | Description |
|-----------|---------------|---|
| MSGSSZ | 32 | Message segment size. The message area is split up into segments of size MSGSSZ bytes. Each message is allocated the number of segments required to hold the message. This value must be greater than and less than the MSGMAX value, and it must be a multiple of 8 bytes. |
| MSGTQL | 100 | Number of message headers in the system. There is one message header per active message. This value must be greater than or equal to the MSGMNI value. The upper limit is 9999. |
| SEMAEM | 16384 | Maximum adjust-on-exit value that can be set on a <code>semop(2)</code> system call by specifying the <code>SEM_UNDO</code> flag. This value must be greater than 0 and less than the SEMVMX value. |
| SEMMNI | 40 | Maximum number of semaphore sets that may be in the system at one time (that is, the size of the <code>sema</code> table). This value must be greater than and less than 1000. |
| SEMMNS | 100 | Number of semaphores in the system (that is, the size of the <code>sem</code> table). A semaphore set may consist of one or more semaphores. This value must be greater than or equal to the SEMMNI value. |
| SEMMNU | 40 | Number of undo structures in the system (that is, the size of the <code>semu</code> table). One entry is used by each process that registers semaphore operations to be undone at process termination. This value must be greater than and less than the NPROC value. A value close to the SEMMNI value will conserve table space. (The undo structure size in words is $3 * SEMUME + 2$.) |
| SEMMSL | 25 | Maximum number of semaphores per set. This value must be greater than 0 and less than the SEMMNI value. |

| Parameter | Default value | Description |
|-----------|---------------|--|
| SEMOPM | 25 | Maximum number of operations per <code>semop(2)</code> system call. Operations can be performed on the semaphores within one set. This value must be greater than 0 and less than or equal to the <code>SEMMSL</code> value. |
| SEMUME | 20 | Maximum number of undo entries per process. Each undo entry in the undo structure allows for an undo operation on one semaphore in a semaphore set. This value must be greater than 0 and less than or equal to the <code>SEMMSL</code> value. To conserve table space, set <code>SEMUME</code> to a value less than the <code>SEMMSL</code> value. (The undo structure size in words is $3 * SEMUME + 2$.) |
| SEMVMX | 32767 | Maximum value of an individual semaphore. This value must be greater than and less than 2^{32} . |

Table 8 lists the parameters in `config.h` for the UNICOS multilevel security (MLS) feature. These parameters are significant only if your site enables the UNICOS MLS feature. See *General UNICOS System Administration*, for more information on setting these parameters and on the UNICOS MLS feature.

Note: In general, a value of 0 disables a parameter and a value of 1 enables it.

Table 8. MLS Kernel Parameters in `config.h`

| Parameter | Default value | Description |
|------------------------|---------------|--|
| COMPART_ACTIVE_DEFAULT | 0 | Generic MLS defaults. |
| COMPART_VALID_DEFAULT | 0 | Generic MLS defaults. |
| CONSOLE_MSG | 0 | Capability to send a message to <code>/dev/console</code> when any user exceeds <code>MAXLOGS</code> login attempts. |
| DECLASSIFY_DISK | 0 | Deactivates declassification of the disk; activates when the value is not 0. |
| DECLASSIFY_PATTERN | 0 | Declassifies disk write pattern. |

| Parameter | Default value | Description |
|----------------------|---------------|---|
| DELAY_MULT | 0 | Multiplier for failed logins. When set to 1 (enabled), the next login attempt is delayed by (# of failed logins) * (LOGDELAY) seconds. Otherwise, the delay period is not multiplied by any factor, and the next login is delayed LOGDELAY seconds. |
| DEV_ENFORCE_ON | 0 | Capability to enforce strict device labeling. |
| DISABLE_ACCT | 0 | Disables the login account when MAXLOGS is exceeded. Used with DISABLE_TIME. |
| DISABLE_TIME | -1 | Duration in seconds for which the user is disabled when the MAXLOGS is exceeded. If -1 (or any negative value), the user is disabled indefinitely. |
| FORCED_SOCKET | NORMAL | When set to NORMAL, syslogd(8) can use sockets and pipes. |
| FSETID_RESTRICT | 1 | If nonzero, only a privileged process can create setuid / setgid files. |
| LOGDELAY | 0 | Maximum number of seconds to delay between login attempts. |
| MAXLOGS | 0 | Maximum number of failed login attempts allowed before a user is disabled if the DISABLE_ACCT is set to 1. |
| MAXSLEVEL | 0 | Maximum security level allowed on the system. |
| MINSLEVEL | 0 | Minimum security level allowed on the system. |
| MLS_INTEGRITY | NORMAL | Integrity code in secure.c. This setting is not in the ICMS. |
| MLS_OBJ_RANGES | NORMAL | Allow object ranges outside the system label range in mount and setdevs. |
| NFS_REMOTE_RW_OK | 1 | Enables (nonzero) or disables (zero) NFS-mounting of a remote file system in read-write mode. |
| NFS_SECURE_EXPORT_OK | 1 | Enables (nonzero) or disables (zero) exporting a secure file system with NFS. |
| OVERWRITE_COUNT | 3 | Declassifies disk overwrite count. |
| PASS_MAXSIZE | 8 | Maximum size for machine-generated passwords. |

| Parameter | Default value | Description |
|-------------------------|---|---|
| PASS_MINSIZE | 8 | Minimum size for machine-generated passwords. |
| PRIV_SU | 1 | When set to 1, root (UID 0) has privilege. |
| RANDOM_PASS_ON | 0 | When set to yes (1), machine-generated passwords are enabled for all users. Setting to no (0) disables this. |
| SANITIZE_PATTERN | 0 | The pattern used to scrub disks if <code>SECURE_SCRUB</code> is enabled. The default pattern is 0. |
| SECURE_NET_OPTIONS | (<code>NETW_SOCKET_COMPAT</code> <code>NETW_RCMD_COMPAT</code>) | <p>Options for running TCP/IP on a secure system. The value may be any combination of the following:</p> <ul style="list-style-type: none"> • <code>NETW_STRICT_B1</code>, in which strict B1 evaluation rules are applied by the system; therefore, TCP sessions are restricted to a single label, regardless of the type of remote host. • <code>NETW_SOCKET_COMPAT</code>, in which sockets are automatically made multilevel if the creating process has <code>PRIV_SOCKET</code>. • <code>NETW_RCMD_COMPAT</code>, in which traditional <code>hosts.equiv</code> and <code>.rhosts</code> behavior is allowed. <p>If <code>NETW_RCMD_COMPAT</code> is not used, remote logins with <code>rlogin(1)</code> to <code>root</code> are disallowed and all other logins must be to the same user name and require the same user ID on both systems. The default value of represents the combination of <code>NETW_SOCKET_COMPAT</code> and <code>NETW_RCMD_COMPAT</code>. To add the strict B1 evaluation rules, you should add the <code>NETW_STRICT_B1</code> value; that is, (<code>NETW_SOCKET_COMPAT</code> <code>NETW_RCMD_COMPAT</code> <code>NETW_STRICT_B1</code>).</p> |
| SECURE_MAC | 0 | When set to 1, enforces system high/system low MAC. |
| SECURE_OPERATOR_CONSOLE | <code>"/dev/console"</code> | Secure console for use by the MLS operator. This parameter is unused. |

| Parameter | Default value | Description |
|-----------------------|---------------|--|
| SECURE_PIPE | NORMAL | Enables/disables “read down” on pipes. |
| SECURE_SCRUB | NORMAL | Enables (SECURE) or disables (NORMAL) scrubbing of data blocks on file deletion. |
| SECURE_SYSTEM_CONSOLE | " " | Secure console for MLS administration. This parameter is unused. |
| SLG_ACT_NFS | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of NFS activity. |
| SLG_ACT_NQS | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of NQS activity. |
| SLG_ALL_NAMI | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of all mkdir, rmdir, link, and rm calls. |
| SLG_ALL_RM | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of all remove requests. |
| SLG_ALL_VALID | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of all access requests. |
| SLG_BUFSIZE | 163840 | Size of the security log buffer. The default value holds approximately 1000 security log records. Must be a multiple of 4. |
| SLG_CF_NET | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of network configuration changes. |
| SLG_CF_NQS | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of NQS configuration changes. |
| SLG_CF_UNICOS | SLGON | Enables (SLGON) or disables (SLGOFF) logging of UNICOS configuration changes. |
| SLG_DIR | /usr/adm/sl | Full path name of directory where security logs are kept. |
| SLG_DISCV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of discretionary access violations. |
| SLG_FILE | slogfile | File name of active security log. |
| SLG_FILEXFR | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all file transfers. |
| SLG_FPREFIX | s. | Prefix of retired security logs. |

| Parameter | Default value | Description |
|------------------|---------------|--|
| SLG_JEND | SLGON | Enables (SLGON) or disables (SLGOFF) logging of job ends. |
| SLG_JSTART | SLGON | Enables (SLGON) or disables (SLGOFF) logging of job starts. |
| SLG_LINKV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all link violations. |
| SLG_LOG_AUDIT | SLGON | Enables (SLGON) or disables (SLGOFF) logging of audit criteria changes. |
| SLG_LOG_CHDIR | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all <code>chdir(1)</code> requests. |
| SLG_LOG_CRL | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of Cray/REELlibrarian activity. |
| SLG_LOG_DAC | SLGON | Enables (SLGON) or disables (SLGOFF) logging of discretionary access control (DAC) activities. |
| SLG_LOG_IPNET | SLGON | Enables (SLGON) or disables (SLGOFF) logging of Internet Protocol (IP) layer activity. |
| SLG_LOG_OPER | SLGON | Enables (SLGON) or disables (SLGOFF) logging of operator actions. |
| SLG_LOG_PRIV | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of all privileges used in system calls. |
| SLG_LOG_SECSYS | SLGON | Enables (SLGON) or disables (SLGOFF) logging of non-inode security system calls. |
| SLG_LOG_SHUTDOWN | SLGON | Enables (SLGON) or disables (SLGOFF) logging of system shutdown. |
| SLG_LOG_STARTUP | SLGON | Enables (SLGON) or disables (SLGOFF) logging of system startup. |
| SLG_LOG_TAPE | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of tape activity. |
| SLG_LOG_TCHG | SLGON | Enables (SLGON) or disables (SLGOFF) logging of system time changes. |
| SLG_MANDV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of mandatory access requests. |

| Parameter | Default value | Description |
|----------------------|-----------------------------|--|
| SLG_MAXSIZE | 8192000 | Size of security log to retire. When <code>SLG_FILE</code> is larger than this, it is moved to a file prefixed with <code>SLG_FPREFIX</code> . |
| SLG_MKDIRV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all <code>mkdir</code> violations. |
| SLG_NETWV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of network access violations. |
| SLG_PATH_TRACK | SLGON | Enables (SLGON) or disables (SLGOFF) tracking of all path names on accesses. |
| SLG_PHYSIO_ERR | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of all physical I/O errors. |
| SLG_REMOVEV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all <code>rm</code> violations. |
| SLG_RMDIRV | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all <code>rmdir</code> violations. |
| SLG_STATE | SLGOFF | Enables (SLGON) or disables (SLGOFF) security log. |
| SLG_SUID_RQ | SLGON | Enables (SLGON) or disables (SLGOFF) logging of <code>setuid</code> system calls. |
| SLG_SULOG | SLGON | Enables (SLGON) or disables (SLGOFF) logging of all <code>su(1)</code> command attempts. |
| SLG_T_PROC | SLGON | Enables (SLGON) or disables (SLGOFF) logging of trusted process activity. |
| SLG_USER | SLGOFF | Enables (SLGON) or disables (SLGOFF) logging of user names and passwords for failed login attempts. |
| SYSTEM_ADMIN_CONSOLE | <code>"/dev/console"</code> | Default console for MLS administration. The value of this parameter must be <code>/dev/console</code> . |
| SYSVCOMPS | 0 | Bit map that defines valid system compartments. By default, system compartments are turned off. |

Table 9 contains the parameter in `config.h` that relates to TCP/IP. See the *UNICOS Networking Facilities Administrator's Guide*, for more information on setting this parameter.

Table 9. TCP/IP Kernel Parameter in config.h

| Parameter | Default value | Description |
|--------------|---------------|--|
| TCP_NMBSPACE | 3800 | Controls the number of MBUFs (TCP/IP managed memory buffers). For information about appropriate values, see <i>UNICOS Networking Facilities Administrator's Guide</i> . This pool of buffers is allocated at boot time in module <code>/usr/src/uts/tcp/kern/uipc_mbuf</code> and cannot be increased while the system is running. The ICMS places this value in both the <code>config.h</code> file and the CSL parameter file. The value in the CSL parameter file overrides the value in <code>config.h</code> . |

Table 10 shows the NFS kernel parameters in `config.h`.

Table 10. NFS Kernel Parameters in config.h

| Parameter | Default value | Description |
|-----------------|---------------|---|
| NFS3_ASYNC_MAX | 64 | The amount of data (in 4096-byte blocks) that will be written per file asynchronously. After this value is exceeded, all writes will be synchronous. (2000 * 4096 = 8,192,000 bytes.) |
| NFS3_ASYNC_TIME | 120 | The amount of time (in seconds) that data will be held in the NFS asynchronous write cache on the client. |
| NFS_DUPTIMEOUT | 3 | Time interval in seconds during which duplicates will not be replayed. |
| NFS_MAXDATA | 32768 | Maximum user data read or written by way of NFS. |
| NFS_MAXDUPREQS | 1200 | Size of the duplicate request cache. This value must be large enough so that the entry will still be there when the first retry comes in. |

| Parameter | Default value | Description |
|---------------------|---------------|--|
| NFS_NUM_RNODES | 256 | Number of NFS nodes that may be read from (known as <i>rnodes</i>) at any one time in the kernel. Each active NFS file or directory requires an <i>rnode</i> . Files may have multiple requests outstanding at any one time. If the number of active NFS files exceeds the NFS_NUM_RNODES value, the <i>rnodes</i> are shared among active files. |
| NFS_PRINTINTER | 0 | Time interval between two messages on the console window when the server is not responding. |
| CNFS_STATIC_CLIENTS | 8 | Number of CNFS static client handles. |
| NFS_STATIC_CLIENTS | 8 | Number of NFS static client handles. |
| CNFS_TEMP_CLIENTS | 8 | Number of CNFS temporary client handles. |
| NFS_TEMP_CLIENTS | 8 | Number of NFS temporary client handles. |

Some of the NFS kernel parameters have CSL boot-time equivalents, as shown in Table 11. These boot-time equivalents override the values that were set in the `config.h` file when the kernel was built. The CSL tags and their values will override the built-in kernel values, and change the performance of the UNICOS operating system.

Table 11. CSL Boot-time Equivalents for NFS Kernel Parameters

| CSL tag | config.h |
|---------------------|---------------------|
| nfs3_async_max | NFS3_ASYNC_MAX |
| nfs3_async_time | NFS3_ASYNC_TIME |
| nfs_duptimeout | NFS_DUPTIMEOUT |
| nfs_maxdata | NFS_MAXDATA |
| nfs_maxdupreqs | NFS_MAXDUPREQS |
| nfs_num_rnodes | NFS_NUM_RNODES |
| nfs_printinter | NFS_PRINTINTER |
| cnfs_static_clients | CNFS_STATIC_CLIENTS |
| nfs_static_clients | NFS_STATIC_CLIENTS |
| cnfs_temp_clients | CNFS_TEMP_CLIENTS |
| nfs_temp_clients | NFS_TEMP_CLIENTS |

config.mh File [4]

Note: It is strongly recommended that you use the install tool to maintain your system configuration, rather than manually editing this file. For more information on the UNICOS installation and configuration menu system (ICMS), refer to the online help files and to the *UNICOS System Configuration Using ICMS*.

The `config.mh` file contains values that are used to set general system parameters, kernel generation parameters, kernel subsystem parameters, and programming environment parameters. Default values are either literals or 1 (which enables a subsystem) or 0 (which disables a subsystem). The following tables describe the parameters and their default values, grouped by type and listed in alphabetical order.

Note: In general, a value of 0 disables a parameter, and a value of 1 enables it.

These parameters are set with the following ICMS menus:

```
Configure System
  ->Major Software Configuration

Configure System
  ->Major Hardware Configuration

Build/Install System
  ->Build options

Configure System
  ->IOS Configuration (Model E based systems only)
```

Table 12. General System Parameters (Common)

| Parameter | Default value | Description |
|----------------|---------------|---|
| CONFIG_DIAGDIR | /ce | Directory where the online diagnostics are kept. This must be a full path name. |
| CONFIG_ID | UNICOS | Identification of the operating system. |

| Parameter | Default value | Description |
|----------------|--------------------|--|
| CONFIG_NODE | <i>node_name</i> | Node name of the system, by which it is known to a communications network. See <code>uname(1)</code> . |
| CONFIG_SN | <i>snnumber</i> | Serial number of the system to generate. |
| CONFIG_SYS | <i>system_name</i> | Name of the system. This information is used by the <code>uname(1)</code> command. Typically, <i>system_name</i> is the same as the serial number. |
| CONFIG_TARGET | (null) | Target of the system to generate. If null, the existing default is used. See <code>target(1)</code> . |
| CONFIG_TMPDIR | <code>/tmp</code> | Name of the desired temporary-file directory. If the <code>TMPDIR</code> environment variable is already defined, this value is not used. See <code>tmpnam(3)</code> . |
| CONFIG_VERSION | <i>version</i> | Version name of the system. If unspecified, it defaults to the value of the environment variable <code>LOGNAME</code> . See <code>uname(1)</code> . |

Table 13. General System Parameters (GigaRing Based Systems Only)

| Parameter | Default value | Description |
|--------------|---------------|---|
| CONFIG_ID | UNICOS | Identification of the operating system. |
| CONFIG_IOS_F | 1 | GigaRing support. |
| CONFIG_MK | 0 | UNICOS/mk operating system (used for CRAY T3E systems only). The default is UNICOS. |

Table 14. General System Parameters (Model E Based Systems Only)

| Parameter | Default value | Description |
|----------------|---------------|---|
| CONFIG_ID | UNICOS | Identification of the operating system. |
| CONFIG_IOS_F | 0 | GigaRing support. |
| CONFIG_IOSA_SN | 0 | IOS-E serial numbers. |
| CONFIG_IOSB_SN | 0 | IOS-E serial numbers. |
| CONFIG_MK | 0 | UNICOS/mk operating system (used for CRAY T3E systems only). The default is UNICOS. |
| CONFIG_NIOS | 1 | Number of IOS-E machines (0, 1, or 2). |

Table 15. Kernel Generation Parameters

| Parameter | Default value | Description |
|--------------------|--------------------------------|---|
| CONFIG_CAM_CPP_LOC | CONFIG_GENBIN/./reqs/cc/mppcpp | The environment variable CAM_CPP_LOCATION is set to the value of CONFIG_CAM_CPP_LOC. The cray-t3e assembler uses this variable. |
| CONFIG_CPP | CONFIG_GENBIN/./reqs/cc/cpp | Sets the nmake(1) CPP variable to the generation cpp. |
| CONFIG_CPSAVE | 0 | The cpset(8) -o option capability. Most released software does not use this option, in which case this parameter has no effect. |
| CONFIG_GCC | CONFIG_GENBIN/cc | Sets the C default to the generation cc compiler. |
| CONFIG_GEN_SEGDIR | CONFIG_GENBIN/./lib/segdirs | The environment variable GEN_SEGDIR is set to the value of CONFIG_GEN_SEGDIR. The segldr(1) command uses this variable. |

| Parameter | Default value | Description |
|----------------------|--------------------------------------|--|
| CONFIG_GENBIN | /usr/gen/bin | Full path of the directory that contains the generation software. |
| CONFIG_GENCMS | CONFIG_GCC CONFIG_CPP | A list of products that must exist as executables. The existence of the listed products is verified when any part of UNICOS is regenerated using <code>nmake(1)</code> . |
| CONFIG_GENPROD_RULES | 0 | Repeatable relocatables capability. |
| CONFIG_MPP_CPP | CONFIG_GENBIN/../../reqs/cc/mppcpp | Sets the <code>nmake(1)</code> CPP variable to the generation <code>mppcpp</code> for the <code>cray-t3e</code> target. |
| CONFIG_PACKAGE | 0 | Certain <code>nmake(1)</code> targets are disabled when this is enabled. Used for packaging purposes only. 1 indicates that this is a packaging build. |
| CONFIG_PATH | CONFIG_GENBIN:/bin:/usr/bin:/usr/ucb | Generation software directory paths. The environment variable <code>PATH</code> is set to the value of <code>CONFIG_PATH</code> . |
| CONFIG_RLS_MAJOR | <i>integer</i> | UNICOS major release number. |
| CONFIG_RLS_MINOR | <i>integer</i> | UNICOS minor release number. |
| CONFIG_RLS_REVISION | <i>integer</i> | UNICOS revision release number. |
| CONFIG_SUPPORT_DIR | (null) | Full path of the directory that contains the support software. |
| CONFIG_TRGBIN | /usr/gen/trg | Full path of the directory that contains the targeting software. |
| CONFIG_UMASK | 022 | The <code>umask(1)</code> setting to be used during the build. |
| CONFIG_XLIBS | 0 | Build and install cross-targeted libraries. This is not available on CRAY T90 IEEE mainframes. |
| CONFIG_XLIBTARGET | <i>target</i> | Desired set of cross-targeted libraries, such as <code>cray-c90</code> for use on a CRAY T90 mainframe. This is not available on CRAY T90 IEEE mainframes. |

| Parameter | Default value | Description |
|--------------------|---------------|---|
| CONFIG_MIXED | 0 | Build and install cross-targeted libraries for a mixed mode (Cray floating-point and IEEE) CRAY T90 CPU system. |
| CONFIG_MIXEDTARGET | <i>target</i> | Desired set of cross-targeted libraries for a mixed mode system, such as <i>cray-ts,ieee</i> for use on a Cray floating-point CRAY T90 mainframe. |

Table 16. Kernel Subsystem Parameters

| Parameter | Default value | Description |
|-----------------|---------------|--|
| CONFIG_BBG | 0 | Bus Based Gateway (BBG). |
| CONFIG_BMM | 0 | Bit matrix multiply functional unit (<i>uts</i> kernel). |
| CONFIG_CRL | 0 | Cray/REELlibrarian. |
| CONFIG_CVT | 1 | Cray Visualization Toolkit (CVT). If this parameter is set to 1, CONFIG_X11 must also be set to 1. |
| CONFIG_DFS | 1 | Distributed Computing Environment (DCE) distributed file system (DFS). |
| CONFIG_DM | 0 | Cray Data Migration Facility (DMF). |
| CONFIG_ELS | 0 | CRAY SV1, CRAY J90, and CRAY J90se support. |
| CONFIG_FQUOTAS | 1 | File quotas. |
| CONFIG_HPI3 | 0 | IPI-3/HIPPI packet driver capability in the kernel. |
| CONFIG_HSX | 0 | HSX device driver (<i>uts</i> kernel). |
| CONFIG_IPI3 | 1 | IPI-3/IPI capability in the kernel. |
| CONFIG_KERBEROS | 0 | Kerberos support. |
| CONFIG_MPP | 1 | CRAY T3D system is attached. |
| CONFIG_NETMON | 1 | Network monitor. |

| Parameter | Default value | Description |
|----------------|---------------|--|
| CONFIG_NETTOLS | 0 | Network testing tools. |
| CONFIG_NFS | 1 | Network File System (NFS). |
| CONFIG_NFS3 | 1 | NFS version 3 Protocol (NFS3). CONFIG_NFS must be configured if this is set to 1. |
| CONFIG_NFSKRB | 0 | NFS with Kerberos authentication. If this parameter is set to 1, CONFIG_NFS and CONFIG_KERBEROS must also be set to 1. |
| CONFIG_OWS | 1 | Operator workstation. This parameter is not used and is provided for compatibility purposes. |
| CONFIG_RPC | 1 | Remote process control system (RPC). |
| CONFIG_TAPE | 1 | Online tape capability in the kernel. |
| CONFIG_TCP | 1 | TCP/IP network system. This parameter must always be set to 1. |
| CONFIG_TRUSTED | 0 | Trusted UNICOS. |
| CONFIG_X11 | 1 | X11 Window Management System. |
| CONFIG_YP | 0 | Yellow pages. |

Table 17. Programming Environment Parameters

| Parameter | Default value | Description |
|-------------------|-------------------------|--|
| CONFIG_CRAYLIBS | CONFIG_GENBIN/./lib | The environment variable GEN_CRAYLIBS is set to the value of CONFIG_CRAYLIBS. The compilers and assembler use this variable. |
| CONFIG_LD_STD_DIR | CONFIG_GENBIN/./lib/cld | The environment variable GEN_LD_STD_DIR is set to the value of CONFIG_LD_STD_DIR. The cld script uses this variable. |

CSL Parameter File [5]

Note: It is strongly recommended that you use the install tool to maintain your system configuration, rather than manually editing this file. For more information on the UNICOS installation and configuration menu system (ICMS), see the online help files and the *UNICOS System Configuration Using ICMS*.

This chapter describes the configuration specification language (CSL) parameter file, which contains statements that specify hardware and software characteristics for your system.

Note: As of the UNICOS 9.2 release, ICMS is not used on UNICOS systems for installation or loading of the UNICOS operating system.

When the configuration is activated, a copy of the CSL parameter file is stored in `/etc/config/param`. For all Cray Research systems, the administrator must manually transfer the parameter file to the workstation.

5.1 CSL Syntax

There are three classes of tokens that make up CSL:

- Identifiers
- Constants
- Operators, separators, and comments

White space (horizontal tabs, new lines, carriage returns, and spaces) separates individual tokens.

5.1.1 Identifiers

An *identifier* is a sequence of digits and letters that specify either special keywords (such as `CONFIG`) or specific objects (such as a physical device). The digits and characters can be enclosed by double quotes. The underscore (`_`) and dash (`-`) are interpreted as letters. Uppercase and lowercase letters are interpreted differently; that is, CSL identifiers are case-sensitive. There are no restrictions on the first character of an identifier.

For example, each of the following is a valid identifier:

```
tmp
STI
abc_d
29-A1-31
Dump
0x1246
"507"
_xyz
```

There are two classes of identifiers:

- *Keyword identifiers* have special meaning in CSL and cannot be used to name other things. For a list of reserved keywords, see Appendix A, page 109.
- *Object identifiers* name specific objects. Objects are divided into three classes:
 - Physical devices
 - Logical devices
 - Slices

Each class of object has its own name space. That is, each object in a given class must have a unique name, but objects in different classes can share the same name.

5.1.2 Constants

All constants are positive integers. An integer consists of 1 digit or a sequence of digits. If the first digit of a constant is (zero), the constant is interpreted as octal; otherwise, the constant is assumed to be decimal. The use of digits 8 or 9 in an octal constant causes an error.

The following are all examples of valid constants:

```
012345      12      44673      0003455
```

5.1.3 Operators, Separators, and Comments

The allowable operators and separators in CSL are as follows:

```
{          }          ;          '          ,
```

The meanings of these operators and separators depend on the context in which they are used.

To intersperse comments between objects, begin and end the comment text as follows:

```
/* This is the comment text. */
```

5.2 CSL Usage

This section provides general information on CSL usage.

CSL statements are located in the CSL parameter file. CSL statements are used to extend or override the default configuration.

All statements in the CSL parameter file must be terminated by a semicolon.

5.2.1 The Boot Process

UNICOS processes CSL statements in order of appearance in the CSL parameter file at boot time.

When processing is finished, a copy of the CSL file is placed in the /CONFIGURATION file.

5.2.2 Configuration Verification

You can verify configurations by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Verify the disk configuration
```

You can also use the user-level program `econfig(8)`, which shows error messages for any errors in the configuration.

The `econfig` program accepts only valid CSL statements as input. It is recommended that you verify the CSL statements by using the `econfig`

command before booting a new configuration to prevent receiving errors during CSL processing.

5.2.3 Error Message Syntax

If, while processing CSL statements, UNICOS detects a condition warranting your attention, a message describing the condition is written to the system console. Conditions warranting a message are divided into three levels of severity:

- Informational messages, which do not affect the boot process.
- Error messages, which inform you of a problem in statement syntax or configuration consistency that prevents the system from booting. Directive processing continues, but the system panics when statement processing completes.
- Panic messages, which inform you of a problem that prevents the completion of statement processing. The system panics immediately.

The syntax of a condition message is as follows:

severity: message_text, location

severity Values are INFO, ERROR, and PANIC.

message_text The message proper.

location Problem location. The *location* is expressed as a line number in the CSL parameter file.

5.3 CSL Parameter File Sections

The statements in the CSL parameter file are organized in the file by functionality:

| <u>Section</u> | <u>Description</u> |
|----------------|---|
| dumpinfo | Model E based system dump parameters |
| gigaring | GigaRing configuration parameters |
| ios_e | Model E and Model V configuration parameters |
| mainframe | Physical mainframe characteristics parameters |

| | |
|------------|---|
| unicos | UNICOS kernel parameters |
| filesystem | Physical storage devices and file system layout parameters |
| network | Network parameters and device parameters |
| mpp | Massively parallel processing (MPP) parameters (Model E based systems only) |
| revision | Revision identifier parameters |
| t90_config | CRAY T90 configuration parameters |



Caution: There are specific naming requirements for naming dump devices and dump locations. For information about these requirements, see *General UNICOS System Administration*.

Note: The default parameter file contains sections that are I/O-specific and therefore will not be used by all systems. You should remove the unused sections from your mainframe's parameter file to avoid potential problems:

- For GigaRing based systems, remove the `dumpinfo` and `ios_e` sections.
- For Model E based systems, remove the `gigaring` section.

5.3.1 dumpinfo Section



Caution: There are specific naming requirements for naming dump devices and dump locations. For information about these requirements, see *General UNICOS System Administration*.

Note: This section does not apply to GigaRing based systems or to Model V based systems.

The `dumpinfo` section of the CSL parameter file defines the types and ranges of memory to dump when invoking the OWS `dumpsys(8)` command.

The `dumpinfo` section is specified in the CSL parameter file using the following syntax:

```
dumpinfo {  
    range_declaration_1;  
    range_declaration_2;  
    .  
    .  
    .  
}
```

A range declaration is specified using the following syntax:

```
type range is start to stop units ;
```

| | |
|--------------|--|
| <i>type</i> | Either <code>memory</code> or <code>SSD</code> , indicating central memory or SSD memory to be dumped. |
| <i>start</i> | Integer indicating the beginning location of a memory section to be dumped. |
| <i>stop</i> | Integer indicating the ending location of a memory section to be dumped. |
| <i>units</i> | Units in which the start and stop values are to be computed: words, Mwords, or Gwords. |

Up to four declarations of each *type* are allowed in the `dumpinfo` section.

The following example shows a typical `dumpinfo` section of a CSL parameter file:

```
dumpinfo {  
    memory range is 0 to 16 Mwords;  
}
```


Note: If the first memory range specification is not large enough to contain the entire kernel space, the mainframe dump routine will override the site specification and try to dump the full kernel space when a system dump is performed. This can lead to a truncated dump if the dump device is not large enough to contain the expanded dump. A truncated dump will not include executing exchange packages, CPU registers, or user areas. This reduces the usefulness of the dump. To override this behavior, and to enforce the site-specified values, insert a memory range specification of 0 to 0 as the first range. Sites that have LDCHCORE, PLCHCORE, and/or a random access memory (RAM) disk configured commonly experience this, because the LDCHCORE, PLCHCORE, and RAM disk space is included within the kernel space.

For example:

```
dumpinfo {
    memory range is 0 to 0 Mwords;
    memory range is 0 to 16 Mwords;
}
```

5.3.2 gigaring Section

Note: This section does not apply to Model E or Model V based systems.

GigaRing configuration in UNICOS is done in two places:

- Internal routing and path selection is done in the `gigaring` section of the parameter file.
- Physical channel designation is done in the `mainframe` section of the parameter file.

The `gigaring` section consists of two subsections:

- `gr_route`
- `gr_union`

5.3.2.1 `gr_route` Subsection

The `gr_route` subsection provides a means of internal routing and path selection known as *source routing*. Source routing chooses the channel used to route traffic to a given ring. Where more than one mainframe GigaRing channel exists on a ring, messages are routed in a round-robin fashion.

Routing information is declared in the `gr_route` subsection. For example:

```
gigaring {
    gr_route {
        ring 06 {
            channel 024;
        }
        ring 07 {
            channel 034;
            channel 044;
        }
    }
}
```

There can be up to 8 routes per ring. By default, the first route declared is designated as the primary route. Valid ring numbers are decimal integers in the range 0 through 127. Valid node numbers are decimal integers in the range 1 through 63.

5.3.2.2 `gr_union` Subsection

A *GigaRing union* is a logical representation of a device that has more than one ring and node designation. For example, a CRAY SSD-T90 with four GigaRing connections (and therefore four ring and node addresses) can be represented by one `gr_union` ring and node address. This applies only to devices for which the mainframe acts as the master for the direct memory access (DMA) operation; in the current implementation, this is only the CRAY SSD-T90.

By convention, ring 0200 is reserved for GigaRing union devices. There is a maximum of 16 nodes (node numbers through 017) reserved for these devices. Devices that are configured as GigaRing union devices allow their device drivers to query the low-level master DMA driver for physical ring and node addresses. The device driver can then route the DMA requests by targeting one physical ring/node address. This is known as *destination routing*. DMA requests can be scheduled by targeting the least busy channel. The retrying of requests in error can be targeted to another destination.

The GigaRing union device allows for ease of configuration and backward compatibility with UNICOS device node methodology.

An example of a `gr_union` declaration is as follows:

```
gigaring {
    gr_union {
        ring 0200, node 01 {
            ring 06, node 04;
            ring 06, node 05;
            ring 07, node 04;
            ring 07, node 05;
        }
    }
}
```

In this example, a GigaRing union logical device designated as ring 0200, node 01, will be created and will consist of four physical destinations.

5.3.3 `ios_e` Section

Note: This section does not apply to GigaRing based systems.

The `ios_e` section defines the topology of the IOS-E hardware. Specify the characteristics using the following menu selection:

```
Configure System
->IOS configuration
```

This section includes the following information:

- Number of clusters that constitute the IOS-E
- Number and type of I/O processors (IOPs) in each cluster
- High-speed (HISP) channel information (channel, target, and operating mode)
- Full OWS path names for each IOP binary

The `ios_e` section is specified in the CSL parameter file by the following statement:

```
ios_e { list of cluster declarations }
```

A cluster declaration is specified by the following statement:

```
cluster ordinal { list of cluster statements }
```

The following types of IOS cluster statements exist:

- IOP declarations
- HISP declarations
- IOP boot declarations

5.3.3.1 IOP Declarations

IOP declarations are specified by the following statement:

```
iop;
```

The following are valid IOPs:

- muxiop or miop
- eiop 0
- eiop 1
- eiop 2
- eiop 3
- eiop 4

Note: eiop 4, muxiop, and miop are equivalent declarations except for Model V based systems. For Model V based systems, eiop indicates the VME controller type and may range from 0 through 50. See the *UNICOS Basic Administration Guide for CRAY J90, CRAY J90se, and CRAY SV1 Model V based Systems*.

5.3.3.2 HISP Declarations

Note: This section does not apply to Model V based systems.

HISP declarations must be correct for the mainframe type on which they are declared, or system problems may occur. HISP declarations are specified by the following statement:

```
channel ordinal is hisp ordinal to chan_target , mode chan_type ;
```

HISP channel targets are as follows:

- mainframe
- SSD

The only channel type is c200d200 (or C90), which stands for 200 Mbyte control / 200 Mbyte data.

5.3.3.3 IOP Boot Declarations

Note: This section does not apply to Model V based systems.

IOP boot declarations are specified by the following statement:

```
boot iop with "pathname" ;
```

If the *pathname* is fully resolved (to the root directory), it is used. If a partial path name is specified, it is appended to the OWS value for ROOTDIR, as specified in the OWS configuration file.

5.3.3.4 Example of ios_e Section

The following example shows the *ios_e* section of a CSL parameter file:

```
ios_e {
    cluster 0 {
        muxiop; eiop 0; eiop 1; eiop 2; eiop 3;
        channel 010 is hisp 0 to mainframe, mode C90;
        channel 014 is hisp 1 to          SSD, mode c200d200;
        boot muxiop with "/home/ows1601/cri/os/ios/iopmux";
        boot eiop 0 with "/home/ows1601/cri/os/ios/eiop.comm";
        boot eiop 1 with "/home/ows1601/cri/os/ios/eiop.bmx";
    }
}
```

```
        boot eiop 2 with "/home/ows1601/cri/os/ios/eiop.dca2";
        boot eiop 3 with "/home/ows1601/cri/os/ios/eiop.dca1";
    }
    cluster 1 {
        muxiop; eiop 0; eiop 1; eiop 2; eiop 3;
        channel 010 is hisp 0 to mainframe, mode C90;
        channel 014 is hisp 1 to          SSD, mode c200d200;
        boot muxiop with "/home/ows1601/cri/os/ios/iopmux";
        boot eiop 0 with "/home/ows1601/cri/os/ios/eiop.dca1";
        boot eiop 1 with "/home/ows1601/cri/os/ios/eiop.dca1";
        boot eiop 2 with "/home/ows1601/cri/os/ios/eiop.dca2";
        boot eiop 3 with "/home/ows1601/cri/os/ios/eiop.hippi";
    }
}
```

5.3.4 mainframe Section

The mainframe section defines the following hardware parameters:

- Number of multi-streaming processors (MSPs) (CRAY SV1 GigaRing based systems only)
- Number of CPUs
- Number of mainframe cluster registers
- Size of the mainframe memory
- Spare chip configuration (CRAY C90 series only)
- Channel information
 - Physical channel for a GigaRing environment
 - Low-speed channel information (channel and target) for Model E based systems
 - VHISP channel information for Model E based systems

Set these parameters by using the following menu selection for Model E based systems:

```
Configure System
->Mainframe hardware configuration
```

The mainframe section is specified in the CSL parameter file by the following statement:

```
mainframe { list of hardware definitions }
```

5.3.4.1 Common Parameters

The following parameters are common to GigaRing based systems and Model E based systems:

- Number of CPUs
- Number of mainframe cluster registers
- Size of memory

5.3.4.1.1 Number of CPUs

The number of CPUs is specified by the following statement:

```
value cpus ;
```

value is the physical number of CPUs in the machine. Specifying a smaller value will cause UNICOS to use only that many CPUs, starting at CPU 0.

5.3.4.1.2 Number of Mainframe Cluster Registers

The number of mainframe cluster registers is specified by the following statement:

```
value clusters ;
```

value is the number of clusters. If this is not specified, it defaults to 1 greater than the *cpus* value.

5.3.4.1.3 Size of Memory

The mainframe memory size is specified by the following statement:

```
value units memory[ , mem_halving] ;
```

units may be either `words` or `Mwords`. Typically, *value* is set to the physical amount of memory in the machine, but it can be set to a smaller value. This may be useful when experiencing memory problems.

The *mem_halving* parameter is for the CRAY C90 series only. It specifies the hardware memory configuration. Values can be as follows:

- `lower half`
- `upper half`
- `both halves`

Your customer engineer can tell you the memory configuration for your machine.

5.3.4.2 Multi-streaming Processors (CRAY SV1 GigaRing Based Systems Only)

The number of multi-streaming processors (MSPs) is specified by the following statement:

```
value msp;
```

value specifies the number of MSPs in the machine. *value* is an integer in the range from 0 through 6.

The maximum number of MSPs configurable is constrained by the number of physical CPUs, as shown in Table 18.

Table 18. CPU/MSP Correlation

| Number of CPUs | Maximum Number of MSPs |
|----------------|------------------------|
| 4 | 0 |
| 8 | 1 |

| Number of CPUs | Maximum Number of MSPs |
|----------------|------------------------|
| 12 | 2 |
| 16 | 3 |
| 20 | 4 |
| 24 | 5 |
| 28 | 6 |
| 32 | 6 |

5.3.4.3 Parameters for GigaRing Based Systems Only

The physical channel configuration declares a physical channel to be a GigaRing channel. It creates a GigaRing port by assigning a ring number and node number to a given mainframe channel number.

```
channel ordinal is gigaring to ring ring_number, node node_number ;
```

ordinal is the channel number. *ring_number* must be an integer in the range through 127. *node_number* must be an integer in the range 1 through 63. For every channel entry in the `gr_route` and `gr_union` sections, there must be a corresponding channel entry in the `mainframe` section.

At UNICOS boot time, a `cnode` structure is declared to represent each GigaRing port. The ring and node numbers will be read and verified from the GigaRing node, or, in the case of a direct connect, be set according to the ring and node numbers specified.

The following channel numbers are valid for CRAY J90, CRAY J90se, and CRAY SV1 GigaRing based systems:

```
024      064
034      074
044      0104
054      0114
```

The following channel numbers are valid for CRAY T90 GigaRing based systems (all even octal integers from 0100 through 0176):

| | | | |
|------|------|------|------|
| 0100 | 0120 | 0140 | 0160 |
| 0102 | 0122 | 0142 | 0162 |
| 0104 | 0124 | 0144 | 0164 |
| 0106 | 0126 | 0146 | 0166 |
| 0110 | 0130 | 0150 | 0170 |
| 0112 | 0132 | 0152 | 0172 |
| 0114 | 0134 | 0154 | 0174 |
| 0116 | 0136 | 0156 | 0176 |

Table 19 shows the channels for I/O module 0.

Table 19. I/O Module 0 Channels

| Erred packet queue 0 | Erred packet queue 1 | Incoming packet queue 0 | Incoming packet queue 1 | Outgoing packet queue | DMA TIB/TCB buffer | Ring number |
|----------------------|----------------------|-------------------------|-------------------------|-----------------------|--------------------|-------------|
| 0100 | 0101 | 0200 | 0201 | 0300 | 0301 | GR 0 |
| 0102 | 0103 | 0202 | 0203 | 0302 | 0303 | GR 1 |
| 0104 | 0105 | 0204 | 0205 | 0304 | 0305 | GR 2 |
| 0106 | 0107 | 0206 | 0207 | 0306 | 0307 | GR 3 |
| 0110 | 0111 | 0210 | 0211 | 0310 | 0311 | GR 4 |
| 0112 | 0113 | 0212 | 0213 | 0312 | 0313 | GR 5 |
| 0114 | 0115 | 0214 | 0215 | 0314 | 0315 | GR 6 |
| 0116 | 0117 | 0216 | 0217 | 0316 | 0317 | GR 7 |

Table 20 shows the channel number ranges for a CRAY T932 configuration of four IO2 modules.

Table 20. Example of Complete Channel Assignments

| Erred packet queue 0 | Erred packet queue 1 | Incoming packet queue 0 | Incoming packet queue 1 | Outgoing packet queue | DMA TIB/TCB buffer | IO2 module number |
|----------------------|----------------------|-------------------------|-------------------------|-----------------------|--------------------|-------------------|
| 0100-0116 | 0101-0117 | 0200-0216 | 0201-0217 | 0300-0316 | 0301-0317 | IO2 - 0 |
| 0120-0136 | 0121-0137 | 0220-0236 | 0221-0237 | 0320-0336 | 0321-0337 | IO2 - 1 |
| 0140-0136 | 0141-0157 | 0240-0256 | 0241-0257 | 0340-0356 | 0341-0357 | IO2 - 2 |
| 0160-0176 | 0161-0177 | 0260-0276 | 0261-0277 | 0360-0376 | 0361-0377 | IO2 - 3 |

5.3.4.4 Parameters for Model E Based Systems Only

The following parameters are for Model E based systems only:

- Channel declarations
- Spare chip configuration (CRAY C90 only)

5.3.4.4.1 Channel Declarations

Channel declarations are either for a low-speed channel pair or a VHISP channel. A channel declaration is specified by the following statement:

```
channel ordinal is channel_type to channel_target ;
```

ordinal is the channel number, for which the preferred format is octal. *channel_type* is either `lowspeed` or `VHISP`. A low-speed *channel_target* is the cluster ordinal or `pseudo TCP`. The VHISP *channel_target* is `SSD`. You must specify `pseudo TCP` for TCP/IP to be operational.

5.3.4.4.2 Spare Chip Configuration for CRAY C90 Series

For the CRAY C90 series, specify the path name to the spare chip configuration file on the OWS by using the following statement:

```
configure C90 spares with "pathname" ;
```

If the *pathname* is fully resolved (to the root directory), it is used. If a partial path name is specified, it is appended to the OWS's value for `ROOTDIR`, as specified in the OWS configuration file. Your customer engineer can tell you if your CRAY C90 system has the memory chip sparing capability.

5.3.4.5 Example of the `mainframe` Section for a GigaRing Based System

The following shows an example of the `mainframe` section of the CSL parameter file for a GigaRing based system:

```
mainframe {  
    16 cpus;  
    2 msp;  
    1024 Mwords memory;  
    channel 024 is gigaring to ring 6, node 1;  
    channel 034 is gigaring to ring 7, node 5;  
    channel 044 is gigaring to ring 7, node 6;  
}
```

5.3.4.6 Example of the `mainframe` Section for a Model E Based System

The following shows an example of the `mainframe` section of the CSL parameter file in a Model E based system:

```
mainframe {  
    2 cpus;  
    32 Mwords memory;  
    channel 16 is lowspeed to cluster 0;  
    channel 18 is lowspeed to cluster 1;  
    channel 1 is vhisps 0 to SSD;  
}
```

5.3.5 `unicos` Section

The `unicos` section sets certain tunable parameters. Set these parameters by using the following menu selection:

```
Configure System  
->UNICOS Kernel Configuration
```

The `unicos` section is specified in the CSL parameter file by the following statement:

UNICOS { *list of tunable parameters* } ;

A UNICOS tunable parameter is specified by the following statement:

value parameter ;

All systems have the same tunable parameters for online tapes, table sizes, and maximum limits. Table 21 through Table 23 show these parameters.

Table 21. Online Tape Parameters

| Parameter | Description |
|------------------|--|
| TAPE_MAX_CONF_UP | Maximum number of tape devices that can be configured up at the same time. |
| TAPE_MAX_DEV | Maximum number of tape devices. |
| TAPE_MAX_PER_DEV | Maximum number of bytes allocated per tape device. |

Table 22. Table Size Parameters

| Parameter | Description |
|-----------|---|
| LDCHCORE | Memory clicks reserved for ldcache blocks assigned as type MEM. |
| NLDCH | Number of ldcache headers. |
| NPBUF | Number of physical I/O buffers. |

Table 23. Maximum Limits Parameters

| Parameter | Description |
|---------------|---|
| FAULT_REPONSE | Enable/disable the fault tolerance feature. |
| GUESTMAX | Maximum number of guest systems. |

| Parameter | Description |
|-----------|---|
| NBUF | Number of buffer headers. |
| NGRT | Maximum number of guest resource table entries. |

GigaRing based systems and Model E based systems have common and unique disk parameters. Table 24 through Table 26 show these parameters.

Table 24. Disk Parameters (Common)

| Parameter | Description |
|-----------|--|
| LDDMAX | Maximum number of logical disk devices (LDDs). This value also limits the minor number for this type. The maximum minor number is LDDMAX -1. |
| MDDSLMAX | Maximum number of mirrored disk device (MDD) slices. This value also limits the minor number for this type. The maximum minor number is MDDSLMAX -1. |
| PDDMAX | Maximum number of physical disk devices (PDDs). |
| PDDSLMAX | Maximum number of PDD slices. This value also limits the minor number for this type. The maximum minor number is PDDSLMAX -1. |
| RDDSLMAX | Maximum number of RAM disk device (RDD) slices. This value also limits the minor number for this type. The maximum minor number is RDDSLMAX -1. |
| SDDSLMAX | Maximum number of striped disk device (SDD) slices. This value also limits the minor number for this type. The maximum minor number is SDDSLMAX -1. |

Table 25. Disk Parameters (GigaRing Based Systems Only)

| Parameter | Description |
|-----------|---|
| SSDTMAX | Maximum number of SSD-T90 devices. The value must be an integer in the range 0 through 8. 0 is the default. |
| SSDTSLMAX | Maximum number of slices that can be allocated to the SSD-T90 device. This value must be an integer in the range 0 through $2^{15} - 1$, depending upon the value set for SSDTMAX. The default is 32. This value also limits the minor number for this type. The maximum minor number is SSDTSLMAX -1. |
| XDDMAX | Maximum number of physical devices. The default is 32. |
| XDDSLMAX | Maximum number of physical slices. The default is 256. |

Table 26. Disk Parameters (Model E Based Systems Only)

| Parameter | Description |
|-----------|--|
| HDDMAX | Maximum number of HIPPI disk devices (HDDs). |
| HDDSLMAX | Maximum number of HDD slices. This value also limits the minor number for this type. The maximum minor number is HDDSLMAX 1. |
| SSDDSLMAX | Maximum number of SSD driver (SSDD) slices. This value also limits the minor number for this type. The maximum minor number is SSDDSLMAX -1. |

Table 27 shows the valid unit bit and range numbers for IONs.

Table 27. ION Unit Bit and Range Numbers (GigaRing Based Systems Only)

| ION type | Bits | Range |
|----------|--|-------|
| FCN | Loop ID 0-7 | 0-127 |
| HPN | Facility bits 0-8 | 0-127 |
| IPN | Unit bits on a daisy chain 0-2 | 0-7 |
| MPN | Device ID 0-7 Logical unit number 0-7 | 0-14 |
| RAID | RAID partition bits 9-15 | 0-127 |

5.3.5.1 Example for a GigaRing Based System

The following is an example `unicos` section for a GigaRing based system:

```
unicos {
    2480    NBUF;                /* System buffers */
    100     NLDCH;              /* Ldcache headers */
    2000    LDCHCORE;          /* Ldcache memory */
    150     LDDMAX;             /* Max. number of LDD devices */
    150     PDDMAX;             /* Max. number of PDD devices */
    256     PDDSLMAX;           /* Max. number of DISK slices */
    32      XDDMAX              /* Max. number of xdisk devices */
    256     XDDSLMAX           /* Max. number of xdisk slices */
    8       SDDSLMAX;           /* Max. number of SDD slices */
    8       MDDSLMAX;           /* Max. number of MDD slices */
    4       RDDSLMAX;           /* Max. number of RAM slices */
    30      TAPE_MAX_CONF_UP;    /* Max. number of tapes configured */
    65536   TAPE_MAX_PER_DEV;    /* Max. tape buffer size */
}
```

5.3.5.2 Additional Parameters for a CRAY SSD-T90 System

The following is an example of additional parameters required for a CRAY SSD-T90 system:

```
unicos {
    2       SSDTMAX;
    16      SSDTSLMAX;
}
```


5.3.5.3 Example for a Model E Based System

The following is an example `unicos` section for a Model E based system:

```
unicos {
    2480    NBUF;                /* System buffers */
    100     NLDCH;              /* Ldcache headers */
    2000    LDCHCORE;          /* Ldcache memory */
    150     LDDMAX;             /* Max. number of LDD devices */
    150     PDDMAX;             /* Max. number of PDD devices */
    256     PDDSLMAX;          /* Max. number of DISK slices */
    8       SDDSLMAX;          /* Max. number of SDD slices */
    8       MDDSLMAX;          /* Max. number of MDD slices */
    4       RDDSLMAX;          /* Max. number of RAM slices */
    4       SSDDSLMAX;         /* Max. number of SSD slices */
    30      TAPE_MAX_CONF_UP;   /* Max. number of tapes configured */
    65536   TAPE_MAX_PER_DEV;  /* Max. tape buffer size */
}
```

5.3.6 filesystem Section

The `filesystem` section includes the following:

- Description of the physical devices in the system:
 - `xdisks`, `disks`, `RAM`, and `SSDT` (CRAY SSD-T90) for GigaRing based systems
 - `disks`, `RAM`, and `SSD` for Model E based systems
- Description of the device nodes in the system:
 - `XDD`, `QDD`, `PDD`, `RDD`, `MDD`, `HDD`, and `SDD` for GigaRing based systems
 - `PDD`, `LDD`, `RDD`, `MDD`, `HDD`, and `SDD` for Model E based systems
- Identification of the root, swap, SDS, and dump devices
- Description of the CRAY SSD-T90 (GigaRing based systems only)

Set these parameters by using the following menu selection:

```
Configure System
->Disk configuration
```

For information on striping file systems, see *General UNICOS System Administration*.

The beginning of the `filesystem` section is indicated by the following line in the CSL parameter file:

```
filesystem {
```

The following sections describe each portion of the `filesystem` section of the parameter file. For more detailed information on physical device specification, see *General UNICOS System Administration*.

Set these parameters by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Physical Devices
```

5.3.6.1 Physical Device Definition

The following sections describe the definition of physical storage devices for GigaRing based systems and Model E based systems. Table 28 summarizes the disk types and names.

Table 28. Disk Type and Name

| Disk type | Disk name |
|-----------|--|
| IPI-2 | DD3, DD4, DD5I, DDAS2, DDES DI, DDIMEM, RAM, RD-1, DA60, DA62, DA301, DA302, DD40, DD41, DD42, DD49, DD50, DD60, DD61, DD62, DD301, DD302, HD16, HD32, HD64, RAM, SSD, |
| SCSI | DD314, DD318, DD501, DD5S, DD6S, DD7S, DD_U |
| Fibre | DD000, DD304, DD308, DD309, DD310 |
| HIPPI | HD16, HD32, HD64 |
| Special | RAM, SSD, SS DT |

5.3.6.1.1 Slice Definitions for Physical Storage Devices

Each physical storage device can be segmented into one or more pieces; each piece is a *slice*. For each slice, a *minor device number*, a starting device address, and the slice length must be specified. The starting device address and the slice length can be specified in units of sectors or blocks. A *block* is defined to be 4096 bytes (512 64-bit words). A *sector* is an integer multiple of some number of blocks that varies with different physical storage devices. The ratio of the device's sector size to the size of a block is defined as one *iounit*. For example, a physical disk device may be formatted with a sector size of 16,384 bytes, giving it an *iounit* value of 4.

Generally, the start and length of a disk slice is specified in units of *sectors*. If the unit value is *blocks*, the start and length values must be integer multiples of the *iounit* value. The *iounit* value for a physical disk storage device is determined by the device *type* designator in the physical disk declaration.

Generally, the start and length of an *xdisk* slice are specified in units of *sectors*. If *blocks*, the start and length values must be integer multiples of the *iounit* value. The *iounit* value for a physical disk storage device is determined by the *iounit* designator in the physical *xdisk* declaration. In the case of an *xdisk* where no *iounit* is declared, the *iounit* value defaults to 1.

The start and length of a slice of a RAM disk are specified in *blocks*.

The start and length of an *ssd* slice can be specified in units of either *blocks* or *sectors*. If *sectors*, the *iounit* value of the SSD is determined by the optional *type* designator in the physical SSD declaration. If a *type* is not specified, the *iounit* value defaults to 1.

The start and length of an *ssdt* slice can be either *blocks* or *sectors*. If *sectors*, are used, the *iounit* of the CRAY SSD-T90 device is determined by the optional *iounit* designator in the physical CRAY SSD-T90 device declaration. If an *iounit* designator is not specified, the *iounit* value defaults to 1.

The following example shows the two forms for slice configuration:

```

slice_type slice_name {
    minor minor_number;
    sector starting_sector_number;
    length length_in_sectors sectors;
}

```

```

slice_type slice_name {
    minor minor_number;
    block starting_block_number
    length length_in_blocks blocks;
}
    
```

Device minor numbers must be unique among all slices of a given storage device type. They must be greater than 0, and less than the maximum number of slices specified in the `unicos` section of the parameter file.

Table 29 shows the preferred and optional start and length units for the various physical storage device types, as well as the parameter in the `unicos` section that determines the maximum value allowable for the minor device number.

Table 29. Start and Length Units for Physical Storage Device Types

| Physical storage device type | Preferred units | Optional units | Maximum minor number -1 |
|------------------------------|-----------------|----------------|-------------------------|
| disk | sectors | blocks | PDDSLMAX |
| xdisk | sectors | blocks | XDDSLMAX |
| hippi_disk | sectors | blocks | HDDSLMAX |
| RAM | blocks | (none) | RDDSLMAX |
| SSD | blocks | sectors | SSDDSLMAX |
| SSD-T90 | blocks | sectors | SSDTSLMAX |

5.3.6.1.2 Physical Device Definition for GigaRing Based Systems

The following types of physical storage devices are available for GigaRing based systems:

- Random access memory (RAM)
- Physical storage devices (disk and xdisk)
- Solid-state storage device for a CRAY T90 system (CRAY SSD-T90)

Table 30 summarizes disk information for GigaRing based systems.

Table 30. Disk Information (GigaRing Based Systems Only)

| Disk type | ION | PCA type | Driver | Node residence | Major number | CSL type | mkspice value |
|-----------|-----|----------|--------|----------------|--------------|----------|---------------|
| IPI-2 | IPN | SPN | qdd | /dev/pdd | dev_qdd | disk | YES |
| SCSI | MPN | MPN | xdd | /dev/xdd | dev_xdd | xdisk | NO |
| Fibre | FCN | SPN | xdd | /dev/xdd | dev_xdd | xdisk | NO |
| HIPPI | HPN | MPN | xdd | /dev/xdd | dev_xdd | xdisk | NO |

The RAM device definition has the following format (which is identical to that in a Model E based system):

```
RAM ram_name
  { length length_number units ;
    pdd pdd_slice_name
      { minor minor_number
        block starting_block_number
        length length_in_blocks blocks
      }
  }
```

- ram_name* Name of the RAM, which must be unique among all devices.
- length_number* Size of the RAM, specified in *units*.
- units* One of the following: blocks, words, or Mwords.
- pdd_slice_name* Name of the slice.
- minor_number* Minor number of the slice, which must be unique across the device type.
- starting_block_number* Block number where the slice starts.
- length_in_blocks* Length of the slice in blocks.

The physical storage device definition has the following format in a GigaRing based system:

```

disk device_name {
    type type;
    iopath {
        ring ring_number;
        node node_number;
        channel channel_number;
    }
    unit disk_unit_number;
    pdd pdd_slice_name {
        minor minor_number;
        sector starting_sector_number;
        length length_in_sectors sectors;
    }
}

```

| | |
|-------------------------------|---|
| <i>device_name</i> | Name of the physical storage device, which must be unique among all devices. |
| <i>type</i> | Type of the physical storage device. |
| <i>ring_number</i> | Number of the I/O path ring. |
| <i>node_number</i> | Number of the I/O path node. |
| <i>channel_number</i> | Number of the I/O path channel. The channel specified is the channel in the peripheral channel adapter (PCA). You must include a leading 0 to specify the channel number in octal form. |
| <i>disk_unit_number</i> | Number of the disk. For disk devices that can be daisy chained, the unit number specifies the physical unit number of the device. It is recommended that start and length for disk devices be expressed in sectors. |
| <i>pdd_slice_name</i> | Name of the slice, which must be unique among all slices for all devices. |
| <i>minor_number</i> | Minor number of the slice, which must be unique across the device type. |
| <i>starting_sector_number</i> | Starting sector number of the slice. |
| <i>length_in_sectors</i> | Length of the slice in sectors. |

The SSDT definition applies only to GigaRing based systems. It has the following format:

```

ssdt ssdt_name {
    iounit iounit_value;
    tmttype tmttype_number;
    [lunit lunit_number;]
    iopath {
        ring ring_number;
        node node_number;
    }
    [piopaths piopaths_bitmask];
    xdd xdd_name {
        minor minor_number;
        block starting_block_number;
        length length_in_blocks blocks;
    }
}

```

| | |
|-----------------------|--|
| <i>ssdt_name</i> | Name of the CRAY SSD-T90, which must be unique among all devices. |
| <i>iounit_value</i> | Block size in 4096-byte units. |
| <i>tmttype_number</i> | Target memory type, which determines the memory address and configuration characteristics for a given CRAY SSD-T90. This value must be an integer in the range through 255. The default is 1. |
| <i>lunit_number</i> | (Optional) The logical unit number of the CRAY SSD-T90; this is only used if there is more than one CRAY SSD-T90 device connected to the system, or if you want to split a single CRAY SSD-T90 into more than one logical device. Normally, a system has only one CRAY SSD-T90 and <i>lunit</i> defaults to 0. |
| <i>ring_number</i> | GigaRing ring number for the CRAY SSD-T90 device. (This can be a logical ring/node address in the form of a <code>gr_union</code> device; see Section 5.3.2.2, page 40.) |
| <i>node_number</i> | GigaRing node number for the CRAY SSD-T90 device. (This can be a logical ring/node address in the form of a <code>gr_union</code> device; see Section 5.3.2.2, page 40.) |

piopaths_bitmask

(Optional) A bit mask representing the number of physical paths (GigaRing channels) used to split a single request across multiple CRAY SSD-T90 connections to a CRAY T90 mainframe. Each bit in the bit mask is a possible path:

- 03 represents 2 paths, which is standard for 512-Mword CRAY SSD-T90 devices
- 017 represents 4 paths, which is standard for 1024-Mword CRAY SSD-T90 devices

Using this parameter can increase the bandwidth of individual large requests but will result in higher system overhead and may decrease overall CRAY SSD-T90 throughput. The `piopaths` parameter is typically used for specific applications (such as NASTRAN) that need the additional bandwidth and cannot use asynchronous requests.

By default, individual user requests are scheduled in a round-robin fashion across the connection and no parallel I/O is done. There are no `piopath` values, which gives the best system throughput for multiple user requests.

xdd_name

Name of the slice, which must be unique among all slices for all devices.

minor_number

Minor number of the slice, which must be unique across the device type.

starting_block_number

Starting block number of the slice.

length_in_blocks

Length of the slice in blocks.

The `xdisk` definition applies only to GigaRing based systems. It has the following format:

```
xdisk xdisk_name {
    iounit iounit_value;
    iopath {
        ring ring_number;
        node node_number;
        channel channel_number;
    }
    unit disk_unit_number
    xdd xdd_slice_name {
        minor minor_number;
        sector starting_sector_number;
        length length_in_sectors;
    }
}
```

| | |
|-------------------------------|--|
| <i>xdisk_name</i> | Name of the physical storage device, which must be unique among all devices. |
| <i>iounit_value</i> | Sector size in 4096-byte I/O units. |
| <i>ring_number</i> | GigaRing ring number of the peripheral channel adapter (PCA). |
| <i>node_number</i> | GigaRing ring node number of the PCA. |
| <i>channel_number</i> | Number of the I/O path channel. The channel specified is the channel in the PCA. You must include a leading 0 to specify the channel number in octal form. |
| <i>disk_unit_number</i> | Device unit number. |
| <i>xdd_slice_name</i> | Name of the slice, which must be unique among all slices for all devices. |
| <i>minor_number</i> | Minor number of the slice, which must be unique across the device type. |
| <i>starting_sector_number</i> | Starting sector number of the slice. |
| <i>length_in_sectors</i> | Length of the slice in sectors. |

5.3.6.1.3 Physical Device Definition for Model E Based Systems

The following types of physical devices are available for Model E based systems:

- Random access memory (RAM)
- Physical storage devices
- Solid-state storage device (SSD)

The RAM device definition has the following format (which is identical to that in a GigaRing based system):

```
RAM ram_name
  { length length_number units ;
  pdd pdd_slice_name
    { minor minor_number
      block starting_block_number
      length length_in_blocks blocks
    }
  }
```

| | |
|------------------------------|---|
| <i>ram_name</i> | Name of the RAM, which must be unique among all devices. |
| <i>length_number</i> | Size of the RAM, specified in <i>units</i> . |
| <i>units</i> | One of the following: blocks, words, or Mwords. |
| <i>pdd_slice_name</i> | Name of the slice. |
| <i>minor_number</i> | Minor number of the slice, which must be unique across the device type. |
| <i>starting_block_number</i> | Block number where the slice starts. |
| <i>length_in_blocks</i> | Length of the slice in blocks. |

The physical storage device definition has the following format for a Model E based system:

```

disk device_name {
    type type;
    iopath {
        cluster cluster_number;
        eiop eiop_number;
        channel channel_number;
    }
    unit disk_unit_number;
    pdd pdd_slice_name {
        minor minor_number;
        sector starting_sector_number;
        length length_in_sectors sector;
    }
}

```

| | |
|-------------------------|--|
| <i>device_name</i> | Name of the physical storage device, which must be unique among all devices. |
| <i>type</i> | Type of the physical storage device. |
| <i>cluster_number</i> | Number of the I/O cluster. Note: For Model V based systems, <i>cluster</i> specifies the VME IOS, and <i>eiop</i> specifies the controller within the VME IOS. For more information, see the <i>UNICOS Basic Administration Guide for CRAY J90, CRAY J90se, and CRAY SV1 Model V based Systems</i> . |
| <i>eiop_number</i> | Number of the EIOP I/O processor. |
| <i>channel_number</i> | Number of the I/O path channel. You must include a leading 0 to specify the channel number in octal form. |
| <i>disk_unit_number</i> | Number of the disk. For disk devices that can be daisy chained, the unit number specifies the physical unit number of the device. It is recommended that start and length for disk devices be expressed in sectors. |
| <i>pdd_slice_name</i> | Name of the slice. |

| | |
|-------------------------------|---|
| <i>minor_number</i> | Minor number of the slice, which must be unique across the device type. |
| <i>starting_sector_number</i> | Starting sector number of the slice. |
| <i>length_in_sectors</i> | Length of the slice in sectors. |

The solid-state storage device (SSD) definition has the following format:

```

SSD ssd_name
  { length length_number units ;
    pdd pdd_slice_name {
      minor minor_number;
      block starting_block_number;
      length length_in_blocks blocks;
    }
  }
    
```

| | |
|------------------------------|---|
| <i>ssd_name</i> | Name of the SSD, which must be unique among all devices. |
| <i>length_number</i> | Size of the SSD. |
| <i>units</i> | One of the following: blocks, Mwords, or sectors. |
| <i>pdd_slice_name</i> | Name of the slice, which must be unique among all slices for all devices. |
| <i>minor_number</i> | Minor number of the slice, which must be unique across the device type. |
| <i>starting_block_number</i> | Starting block number of the slice. |
| <i>length_in_blocks</i> | Length of the slice in blocks. |

5.3.6.2 Device Node Definition

The following device nodes can be defined in the `filesystem` section of the CSL parameter file:

| <u>Device type</u> | <u>Description</u> |
|--------------------|---|
| <code>pdd</code> | Physical device for use with the CSL type disk. |
| <code>ldd</code> | Logical device. |
| <code>sdd</code> | Striped device. |

| | |
|-----|---|
| mdd | Mirrored device. |
| qdd | Physical device for GigaRing based systems; used to divide xdisk entries in the filesystem section. |
| xdd | Physical disk device for GigaRing based systems for use with the CSL type xdisk. |

The only limitation is that any slice used in a node definition must have been defined in a physical storage device definition. For more information on mirrored and striped devices, see *General UNICOS System Administration*.

Set the QDD and PDD parameters by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Physical Device Slices (/dev/pdd entries)
```

Set the LDD parameters by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Logical Devices (/dev/dsk entries)
```

Set the SDD parameters by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Striped Devices (/dev/sdd entries)
```

Set the MDD parameters by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Mirrored Devices (/dev/mdd entries)
```

Set the XDD parameters by using the following menu selection:

```
Configure System
  ->Disk Configuration
    ->Physical Device Slices on GigaRing Systems
      (/dev/xdd entries)
```

Each node definition has the following syntax:

```
node_type name {  
    minor number;  
    device slice;
```

The node type and device are one of the device types defined in the previous list. The name is site-configurable. The minor number is required and must be unique across the device type. The slice is a name of a slice (or slices or other device node definition) previously defined in your CSL parameter file.

5.3.6.3 CRAY SSD-T90 Description

The CRAY SSD-T90 configuration reflects the explicit nature of the GigaRing, as opposed to the implicit VHISP form used for Model E based systems. A CRAY SSD-T90 device can have either a physical GigaRing *iopath* or a GigaRing union *iopath*.

Note: The CRAY SSD-T90 has an *iopath* designator (without a channel number) in its declaration.

The *lunit* (logical unit) parameter is the device ordinal that will be assigned to this physical CRAY SSD-T90. With one CRAY SSD-T90, the *lunit* parameter is optional and defaults to 0. If more than one CRAY SSD-T90 is designated, *lunit* is required, must be unique among SSDs, and must be smaller than the maximum number of CRAY SSD-T90 devices. The maximum number of CRAY SSD-T90 devices is designated by the *SSDTMAX* parameter in the *unicos* section of the parameter file. See Section 5.3.6.1, page 56, for the format.

The *tmttype* parameter is the target memory type associated with the CRAY SSD-T90. The target memory type determines memory address and configuration characteristics for a given CRAY SSD-T90. Table 31 shows the *tmttype* values.

Table 31. Target Memory Type Values

| <i>tmttype</i> value | Description |
|----------------------|-----------------------|
| 8 | An 8-processor system |
| 9 | A 16-processor system |

5.3.6.4 Root, Swap, and Secondary Data Segment (SDS) Devices

The statements for the root, swap, and SDS devices have the following syntax in the `filesystem` section of the CSL parameter file for GigaRing based systems:

```
rootdev is ldd name;
swapdev is ldd name;
sdsdev is xdd name;
dmpdev is xdd name;
```

`sdsdev` must be an SSD or CRAY SSD-T90 slice; `dmpdev` must be an XDD definition; the others must be an LDD definition.

These statements have the following syntax for Model E based systems:

```
rootdev is ldd name;
swapdev is ldd name;
sdsdev is pdd name;
dmpdev is ldd name;
```

Set these parameters by using the following menu selection:

```
Configure System
->Disk Configuration
->Special System Device Definitions
```

5.3.6.5 Example of the `filesystem` Section Containing a RAM File System

The following example shows the `filesystem` section of a working CSL parameter file containing a RAM file system:

Note: Additional CSL tags are required in the `unicos` section. See Section 5.3.5, page 50.

```
filesystem {
  RAM ramdev {length 10240 blocks;
    pdd ram {minor 3; block 0; length 10240 blocks;}
  }
}
```

5.3.6.6 Example of the `filesystem` Section for a GigaRing Based System

The following example shows the `filesystem` section of a working CSL parameter file for a GigaRing based system:

Note: Additional CSL tags are required in the `unicos` section. See Section 5.3.5, page 50.

```
filesystem {
  /* Physical device configuration */
  xdisk d04026.3 { iounit 1;
    iopath { ring 04; node 02; channel 06; } unit 03;
    pdd 04026.3_usr_i { minor 231; sector 0;          length 444864 sectors; }
    pdd 04026.3_ccn   { minor 232; sector 444864; length 222432 sectors; }
  }
  xdisk d03020.0 { iounit 1;
    iopath { ring 03; node 02; channel 0; } unit 0;
    xdd mpn.s400 { minor 17; sector 0;          length 102000 sectors; }
    xdd mpn.roote { minor 18; sector 102000; length 250000 sectors; }
    xdd mpn.usre  { minor 19; sector 352000; length 250000 sectors; }
  }
  /* HPN Device */
  xdisk d257.200.78.223 { iounit 22;
    iopath { ring 02; node 05; channel 07; }
    unit 40136; ifield 0337;
    xdd hpn.1 { minor 23; block 0; length 250000 blocks; }
  }
  /* Logical device configuration */
  ldd usr_i { minor 86; xdd 04026.3_usr_i; }
  ldd ccn   { minor 40; xdd 04026.3_ccn ; }
  ldd root_e { minor 17; xdd mpn.roote ; }
  ldd usr_e { minor 30; xdd mpn.usre ; }
  ldd swap  { minor 2; xdd mpn.s400 ; }
  ldd hpn   { minor 10; xdd hpn.1 ; }
  rootdev is ldd root_e;
  swapdev is ldd swap;
}
```


5.3.6.7 Example of the filesystem Section for a GigaRing Based System with Disk Devices Configured for Third-party I/O

The following example shows the filesystem section of a working CSL parameter file for a GigaRing based system with disk devices configured for third-party I/O.

Note: Additional CSL tags are required in the unicos section. See Section 5.3.5, page 50.

```
filesystem {
  /* Physical device configuration */
  xdisk d04026.3 { iounit 1;
    iopath { ring 04; node 02; channel 06; } unit 03;
    pdd 04026.3_usr_i { minor 231; sector 0; length 444864 sectors; }
    pdd 04026.3_ccn { minor 232; sector 444864; length 222432 sectors; }
  }
  xdisk d03020.0 { iounit 1;
    iopath { ring 03; node 02; channel 0; } unit 0;
    xdd mpn.s400 { minor 17; sector 0; length 102000 sectors; }
    xdd mpn.roote { minor 18; sector 102000; length 250000 sectors; }
    xdd mpn.usre { minor 19; sector 352000; length 250000 sectors; }
    xdd mpn.dump { minor 136; sector 807360; length 100000 sectors; }
  }
  /* HPN Device */
  xdisk d257.200.78.223 { iounit 22;
    iopath { ring 02; node 05; channel 07; }
    unit 40136; ifield 0337;
    xdd hpn.1 { minor 23; block 0; length 250000 blocks; }
  }
  /* Logical device configuration */
  ldd usr_i { minor 86; xdd 04026.3_usr_i ; }
  ldd ccn { minor 40; xdd 04026.3_ccn ; }
  ldd root_e { minor 17; xdd mpn.roote ; }
  ldd usr_e { minor 30; xdd mpn.usre ; }
  ldd swap { minor 2; xdd mpn.s400 ; }
  ldd hpn { minor 10; xdd hpn.1 ; }
  rootdev is ldd root_e;
  swapdev is ldd swap;
  dmpdev is xdd mpn.dump;
}
```

5.3.6.8 Example of the filesystem Section for a CRAY SSD-T90 Device

The following example shows the filesystem section of a working CSL parameter file for a CRAY SSD-T90 device.

Note: Additional CSL tags are required in the unicos section. See Section 5.3.5, page 50.

```
filesystem {
    ssdt SSDT00 { iounit 1; tmttype 1; lunit 0;
        iopath { ring 0200; node 01; }
        length 512 Mwords;
        xdd ssd_blk0 { minor 2; block 0; length 131072 blocks; }
        xdd ssd_blk1 { minor 3; block 131072; length 131072 blocks; }
    }
    sdsdev is xdd ssd_blk1;
}
```

5.3.6.9 Example of the filesystem Section for Model E Based Systems

The following example shows the filesystem section of a working CSL parameter file for Model E based systems:

```
filesystem {
    disk d0230.0 {type DD62; iopath{cluster 0; eiop 2; channel 030;} unit 0;
        pdd root.0 {minor 10; sector 0; length 166824 sectors;}
        pdd usr.1 {minor 11; sector 166824; length 166824 sectors;}
        pdd core {minor 12; sector 333648; length 333648 sectors;}
    }
    disk d0232.0 {type DD62; iopath{cluster 0; eiop 2; channel 032;} unit 0;
        pdd root.1 {minor 15; sector 0; length 166824 sectors;}
        pdd usr.0 {minor 16; sector 166824; length 166824 sectors;}
        pdd scratch {minor 17; sector 333648; length 333648 sectors;}
    }
    disk d0234.0 {type DD62; iopath{cluster 0; eiop 2; channel 034;} unit 0;
        pdd src {minor 20; sector 0; length 667296 sectors;}
    }
    disk d1230.0 {type DD62; iopath{cluster 1; eiop 2; channel 030;} unit 0;
        pdd mfs0 {minor 60; sector 0; length 166824 sectors;}
        pdd scr1 {minor 61; sector 166824; length 500472 sectors;}
    }
    disk d1232.0 {type DD62; iopath{cluster 1; eiop 2; channel 032;} unit 0;
        pdd mfs1 {minor 62; sector 0; length 166824 sectors;}
        pdd scr2 {minor 63; sector 166824; length 333648 sectors;}
    }
}
```

```
    pdd dump          {minor 64; sector 500472; length 166824 sectors;}
  }
  disk dl234.0 {type DD62; iopath{cluster 1; eiop 2; channel 034;} unit 0;
    pdd stripe0      {minor 70; sector 0; length 667296 sectors;}
  }
  disk dl236.0 {type DD62; iopath{cluster 1; eiop 2; channel 036;} unit 0;
    pdd stripe1      {minor 71; sector 0; length 667296 sectors;}
  }
  sdd strfs {minor 1; pdd stripe0;
             pdd stripe1;}
  mdd mfs   {minor 1; pdd mfs0;
             pdd mfs1;}
  ldd root0 { minor 10; pdd root.0 ;}
  ldd usr   { minor 11; pdd usr.0 ;}
  ldd src   { minor 12; pdd src ;}
  ldd core  { minor 13; pdd core ;}
  ldd dump  { minor 14; pdd dump ;}
  ldd bkroot { minor 15; pdd root.1 ;}
  ldd bkusr  { minor 16; pdd usr.1 ;}
  ldd tmp    { minor 21; pdd tmp0 ;
             pdd tmp1 ;}
  ldd usrtmp { minor 27; pdd usrtmp ;}
  ldd scratch { minor 35; pdd scratch ;
              pdd scr1 ;
              pdd scr2 ;}
  ldd mir.fs { minor 30; mdd mfs ;}
  ldd str.fs { minor 40; sdd strfs ;}
  rootdev is ldd root0;
  swapdev is ldd swap;
  dmpdev is ldd dump;
}
```

5.3.6.10 Example of the `filesystem` Section Containing an SSD for Model E Based Systems

The following example shows the `filesystem` section of a working CSL parameter file containing an SSD:

Note: Additional CSL tags are required in the `unicos` section. See Section 5.3.5, page 50.

```
filesystem {
  SSD ssddev    {length 512 Mwords;
    pdd ssd_0a   {minor  2; block      0; length  524288 blocks;}
    pdd ssd_0b   {minor  3; block  524288; length  524288 blocks;}
  }
  ldd swap      { minor  23; pdd ssd_0a      ;}
  swapdev is ldd swap;
  sdsdev  is pdd ssd_0b;
}
```

5.3.7 network Section

The `network` section defines network devices and network parameters. You can configure them by using the following menu:

```
Configure System
  ->UNICOS Kernel Configuration
    ->Communication Channel Configuration
```

The `network` section includes the following information:

- Descriptions of network parameters
- Customized network device prototypes (Model E based systems only)
- Descriptions of each specific network device using standard templates or customized prototypes

The `network` section is specified in the CSL parameter file in the following manner:

```
network {
  number network_parameter_statement;
  custom_network_device_specification_statement;
  physical_network_device_statement;
}
```

The three statements are repeated as necessary to describe the network device configuration.

The following sections describe the three statement types that compose the network section.

5.3.7.1 Network Parameters

You can set the network parameters by using the following menu selections:

```
Configure System
  ->UNICOS Kernel Configuration
    ->Network Parameters
```

and

```
Configure System
  ->Network Configuration
    ->TCP/IP Configuration
      ->TCP Kernel Parameters Configuration
```

The network parameter statement has the following format:

| |
|--|
| <i>number network_parameter_statement;</i> |
|--|

The *number* is a valid CSL number for the network statement. *network_parameter_statement* argument can have the values described in Table 32 through Table 36, page 78.

Table 32. Network Parameter Values (Common)

| Parameters | Description |
|----------------|---|
| atmarp_entries | Size of the asynchronous transfer mode (ATM) address resolution protocol (ARP) table. |
| atmarp_recv | Amount of socket space used for receive for ATM ARP traffic. Must be a power of 2. |
| atmarp_send | Amount of socket space used for send for ATM ARP traffic. Must be a power of 2. |

| Parameters | Description |
|----------------------------------|--|
| <code>cnfs_static_clients</code> | Maximum number of active Cray NFS static clients. |
| <code>cnfs_temp_clients</code> | Maximum number of active Cray NFS temporary clients. |
| <code>harp_entries</code> | Size of the High Performance Parallel Interface (HIPPI) address resolution protocol (ARP) table. |
| <code>harp_recv</code> | Amount of socket space used for receive for HIPPI ARP traffic. Must be a power of 2. |
| <code>harp_send</code> | Amount of socket space used for send for HIPPI ARP traffic. Must be a power of 2. |
| <code>hidirmode</code> | Sets permissions or file mode for the following directories: <code>/dev/ghippi#</code> (GigaRing based system) and <code>/dev/hippi</code> (Model E based system). |
| <code>hifilemode</code> | Sets permissions or file mode for the following files: <code>/dev/ghippi#/*</code> (GigaRing based system) and <code>/dev/hippi/*</code> (Model E based system). |
| <code>nfs_duptimeout</code> | Time interval in seconds during which duplicate requests received by the NFS server will be dropped. |
| <code>nfs_maxdata</code> | Maximum amount of data that can be transferred in an NFS request (the NFS data buffer size). |
| <code>nfs_maxdupreqs</code> | Size of the NFS server's duplicate request cache. |
| <code>nfs_num_rnodes</code> | Size of the NFS client's NFS file-system-dependent node table (rnode table for NFS). |
| <code>nfs_printinter</code> | Time in seconds between server not responding message to a down server. |
| <code>nfs_static_clients</code> | Number of static client handles reserved for NFS client activity. |
| <code>nfs_temp_clients</code> | Number of dynamically allocated client handles that can be used for NFS client activity when all of the static client handles are in use. |
| <code>nfs_wcredmax</code> | Maximum number of credential structures. |

| Parameters | Description |
|------------------------------|---|
| <code>nfs3_async_max</code> | Maximum amount of data (in 4096-byte blocks) that will be written per file asynchronously. After this value is exceeded, all writes will be synchronous. The default is 2000 (2000 * 4096 = 8,192,000 bytes). |
| <code>nfs3_async_time</code> | The amount of time (in seconds) that data will be held in the NFS async write cache on the client. |
| <code>tcp_numbspace</code> | Number of clicks of memory set aside for TCP/IP managed memory buffers (MBUFs). |

Table 33. Network Parameter Values (GigaRing Based Systems)

| Parameters | Description |
|-------------------------|---|
| <code>ithreshold</code> | (Optional) Specifies the multiplier (an integer in the range 1 through 4) used to configure the input response table size. The default is 1. This parameter is ignored for <code>gr</code> interfaces. |
| <code>othreshold</code> | (Optional) Specifies the multiplier (an integer in the range 1 through 4) used to configure the output response table size. The default is 1. This parameter is ignored for <code>gr</code> interfaces. |
| <code>maxinputs</code> | Maximum number of asynchronous read I/O requests that can be issued to the I/O node. This must be an integer value in the range 1 through 256. The default is 16. |
| <code>maxoutputs</code> | Maximum number of write I/O requests that can be issued to the ION. This must be an integer value in the range 1 through 256. The default is 16. |
| <code>maxusers</code> | Maximum number of applications that can share the HIPPI device. This parameter applies only to HIPPI devices; it must be set to 1 for other interfaces. For HIPPI devices, the value must be an integer in the range 1 through 8. The default is 2. |

Table 34. Network Parameter Values (Common to Model E and Model V Based Systems)

| Parameters | Description |
|------------|--|
| fdmaxdevs | Maximum number of fiber distributed data interface (FDDI) FCA-1 channel adapters. |
| himaxdevs | Maximum number of channels that you can configure for HIPPI. |
| himaxpaths | Maximum numbers of logical paths per channel that you can configure for HIPPI. |
| npmaxdevs | Maximum number of channels that can be configured for low-speed channel communication devices. |

Table 35. Network Parameter Values (Model V Based Systems)

| Parameters | Description |
|------------|-------------------------------------|
| atmmxdevs | Maximum number of ATM devices. |
| enmaxdevs | Maximum number of Ethernet devices. |

Table 36. Network Parameter Values (Model E Based Systems)

| Parameters | Description |
|------------|--|
| fddirmode | Sets permissions of file mode for <code>/dev/fddi*</code> directories. |
| fdfilemode | Sets permission of file mode for <code>/dev/fddi*/*</code> files. |
| fdmaxpaths | Maximum number of logical paths per physical FDDI channel. |
| npdirmode | Sets the permissions for subdirectories of <code>/dev/comm</code> . |
| npfilemode | Sets the permissions for <code>/dev/comm/CHAN/lpXX</code> files. |

| Parameters | Description |
|------------|---|
| npito | Input time-out. |
| npmaxpaths | Total number of logical paths for low-speed channel communication devices. |
| npmaxppd | Maximum number of paths per device for low-speed channel communications devices. |
| npoto | Output time-out. |
| nprthresh | Reads threshold for low-speed channels (number of reads that can be queued to the IOS at one time). |
| nprto | Default read time-out (per path). |
| npwthresh | Writes threshold for low-speed channels (number of writes that can be queued to the IOS at one time). |

5.3.7.2 Customized Network Device Prototypes for Model E Based Systems

The customized network device prototype lets you define characteristics for low-speed devices unique to your site. The network device prototype statement has the following syntax:

```
np_spec interface_type {
    { device type device_type;
      channel mode channel_mode;
      driver type driver_type;
      driver mode driver_mode_number;
      device function function_number;
      direction timeout timeout_number;
    }
}
```

| | |
|-----------------------|--|
| <i>interface_type</i> | Type of the interface as defined in Table 37, page 81. |
| <i>device_type</i> | Type of the device as defined in Section 5.3.7.3. |
| <i>channel_mode</i> | One of the following values: 12MB |

| | |
|---------------------------|--|
| | 12MB_LP |
| | 6MB |
| <i>driver_type</i> | One of the following values: |
| | FY |
| | LCP |
| | MP |
| | NSC_MP |
| | PB |
| | RAW |
| <i>driver_mode_number</i> | Number of the driver mode. |
| <i>function_number</i> | Number of the function. |
| <i>direction</i> | Either input or output (only used with HIPPI devices). |
| <i>timeout_number</i> | Time-out value. |

5.3.7.3 device type for Model E Based Systems

The `device type` statement, which defines the network device type in IOS Model E and Model V systems, has the following format:

```
device type device;
```

The following are valid device types:

| <u>Device type</u> | <u>Description</u> |
|--------------------|--------------------|
| A130 | NSC A130 devices |
| CNT | CNT devices |
| DIAG | Diagnostic devices |
| DX4130 | FDDI connection |
| DX8130 | FDDI connection |
| EN643 | IP router |
| FEI_3 | FEI-3 devices |
| FEI_3FY | FEI-3FY devices |

| | |
|--------|-------------------------------------|
| FEI_4 | FEI-4 devices |
| FEI_CN | FEI-CN devices |
| FEI_DS | FEI-1 data-streaming IBM connection |
| FEI_UC | FEI-1 user driver channel |
| FEI_VA | VAX FEI-1, port A |
| FEI_VB | VAX FEI-1, port B |
| FEI_VM | Cray devices |
| N130 | NSC N130 devices |
| VME | VME-bus devices |

Cray Research provides you with configuration templates for a number of standard interfaces, as shown in Table 37.

Table 37. Standard Interface Configuration Templates

| Interface type | Device type | Channel mode | Driver type | Driver mode | Device func | Input time-out | Output time-out | Read time-out |
|----------------|-------------|--------------|-------------|-------------|-------------|----------------|-----------------|---------------|
| S_DIAG6 | DIAG | 6MB | RAW | 0 | 0 | 100 | 100 | 100 |
| S_DIAG12 | DIAG | 12MB | RAW | 0 | 0 | 100 | 100 | 100 |
| S_DIAG12L | DIAG | 12MB_LP | RAW | 0 | 0 | 100 | 100 | 100 |
| S_FEICN | FEI_CN | 6MB | LCP | 0 | 0 | 100 | 100 | 600 |
| S_FEIDS | FEI_DS | 6MB | LCP | 0 | 0 | 100 | 100 | 600 |
| S_FEIUC | FEI_UC | 6MB | LCP | 0 | 0 | 100 | 100 | 600 |
| S_FEIVA | FEI_VA | 6MB | LCP | 1 | 0 | 100 | 100 | 600 |
| S_FEIVB | FEI_VB | 6MB | LCP | 2 | 0 | 100 | 100 | 600 |
| S_FEIVM | FEI_VM | 6MB | LCP | 0 | 0 | 100 | 100 | 600 |

| Interface type | Device type | Channel mode | Driver type | Driver mode | Device func | Input time-out | Output time-out | Read time-out |
|----------------|-------------|--------------|-------------|-------------|-------------|----------------|-----------------|---------------|
| S_FEI3 | FEI_3 | 6MB | MP | 0 | 0 | 100 | 100 | 600 |
| S_FEI312 | FEI_3 | 12MB | MP | 0 | 0 | 100 | 100 | 600 |
| S_VAXBI | VAX_BI | 12MB | MP | 1 | 0 | 100 | 100 | 600 |
| S_N130X | N130 | 12MB | PB | 0 | 0 | 100 | 100 | 600 |
| S_EN643X | EN643 | 12MB | PB | 0 | 0 | 100 | 100 | 600 |
| S_DX4130X | DX4130 | 12MB | PB | 0 | 0 | 100 | 100 | 600 |
| S_DX8130X | DX8130 | 12MB | PB | 0 | 0 | 100 | 100 | 600 |
| S_FEI3FY | FEI_3FY | 6MB | FY | 010 | 0 | 100 | 100 | 600 |
| S_FEI4 | FEI_4 | 6MB | FY | 010 | 0 | 100 | 100 | 600 |
| S_A130X | A130 | 12MB | NSC_MP | 0 | 0 | 100 | 100 | 600 |

If your site has a network device that does not match these definitions, you must customize a device definition by using the customized network device prototypes, as described in this section. The menu that you would use to do so is as follows:

```
Communication Channel Configuration
->Custom network device specification
```

5.3.7.4 Device Types

The following tables describe the types of devices.

Note: For more information about ATM, see the *Asynchronous Transfer Mode (ATM) Administrator's Guide*.

Table 38. Network Device Types (GigaRing Based Systems)

| Device type | Description |
|-------------|--|
| gfddi | GigaRing FDDI device. |
| gatm | GigaRing asynchronous transfer mode (ATM) device. |
| gether | GigaRing Ethernet device. |
| ghippi | GigaRing HIPPI device. |
| gr | GigaRing TCP/IP device (provides direct mainframe-to-mainframe communication). |

Table 39. Network Device Type (Model E Based Systems)

| Device type | Description |
|-------------|---|
| fddev | IOS-E FDDI device. |
| hi | High-speed HIPPI device. |
| npdev | Low-speed channel; if this device is specified, then the <i>number</i> argument is the ordinal of the network device. |

Table 40. Network Device Type (Model V Based Systems)

| Device type | Description |
|-------------|------------------|
| atmdev | ATM device. |
| endev | Ethernet device. |
| fddev | FDDI device. |

5.3.7.5 Device Formats

The following sections describe device formats.

5.3.7.5.1 Device Formats for GigaRing Based Systems

The following formats apply to GigaRing based systems:

```

gatm number {
    iopath {
        iopath_information
    }
    maxusers number;
    maxinputs number;
    maxoutputs number;
    [ithreshold number;]
    [othreshold number;]
}

```

```

gether number {
    iopath {
        iopath_information
    }
    maxusers number;
    maxinputs number;
    maxoutputs number;
    [ithreshold number;]
    [othreshold number;]
}

```

```

gfddi number {
    iopath {
        iopath_information
    }
    maxusers number;
    maxinputs number;
    maxoutputs number;
    [ithreshold number;]
    [othreshold number;]
}

```

```
ghippi number {
    iopath {
        iopath_information
    }
    maxusers number;
    maxinputs number;
    maxoutputs number;
    [ithreshold number];
    [othreshold number];
}
```

```
gr number {
    iopath {
        ring ring_number;
        node node_number;
    }
}
```

In the `gr` device format, *ring_number* identifies the GigaRing that TCP/IP will use to communicate between mainframes, and *node_number* identifies this mainframe's assigned node ID.

5.3.7.5.2 Device Format for Model E Based Systems

The following formats apply to Model E based systems only:

```
fddev number {
    padcnt number;
    treq number;
    maxwrt number;
    maxrd number;
    iopath {
        iopath_information
    }
}
```

Note: There are two formats for `hidev`: one for input and one for output.

```
hidev number {  
    iopath {  
        iopath_information  
    }  
    logical path number {  
        flags number;  
        I_field number;  
        ULP_id number;  
    }  
    flags number;  
    input;  
    device type type;  
}
```

```
hidev number {  
    iopath {  
        iopath_information  
    }  
    logical path number {  
        flags number;  
        I_field number;  
        ULP_id number;  
    }  
    flags number;  
    output;  
    device type type;  
}
```

```
npdev number {  
    iopath {  
        iopath_information  
    }  
    np_spec name;  
}
```

5.3.7.5.3 Device Formats for Model V Based Systems

The following formats apply to Model V based systems only:


```

atmdev number {
    iopath {
        iopath_information
    }
}

```

```

endev number {
    iopath {
        iopath_information
    }
}

```

```

fddev number {
    iopath {
        iopath_information
    }
}

```

5.3.7.6 logical path for Model E Based Systems

The logical path statement has the following format:

```

logical path number {
    identifier number
}

```

If you have a high-speed HIPPI network device, you could use the *identifier* argument, which has the following values:

| <u>Identifier</u> | <u>Description</u> |
|-------------------|---|
| UL_id | Upper-level protocol identifier; it is a value between 0 and 255 that identifies the protocol or user of a HIPPI packet. The value is the first byte of each HIPPI packet and is used with <code>input</code> to direct incoming packets to the appropriate process (TCP/IP uses a value of 004). |
| I_field | A data value that is passed over the HIPPI channel when a connection is made. In HXCF_DISC mode, the I-field value precedes |

| | |
|-------|---|
| flags | <p>each packet. The <code>I-field</code> value selects the output port of an NSC P_8 or PS_32 switch.</p> <p>Flags related to the specific HIPPI logical device. The only flag available at this time is 02000 (<code>HXCF_IND</code>). When this flag is set, the driver interprets the first word of each user output buffer as the <code>I-field</code> value. The driver puts the incoming <code>I-field</code> value in the first word of the user input buffer.</p> |
|-------|---|

5.3.7.7 *direction* Argument for Model E Based Systems

The *direction* argument has the following syntax:

```
direction;
```

The *direction* argument (used only with HIPPI devices) can be either input or output.

5.3.7.8 *device type* Statement for Model E Based Systems

The *device type* statement has the following syntax:

```
device type device ;
```

device defines the device connected to the HIPPI channel.

5.3.7.9 *np_spec* Statement for Model E Based Systems

The *np_spec* statement has the following syntax:

```
np_spec name
```

name specifies the customized network device specification describing the network device connected to the low-speed channel.

5.3.7.10 network Section Example Common to All Systems

The following example configures basic network parameters that are needed for all systems:

```
network {
    8 nfs_static_clients;
    8 nfs_temp_clients;
    8 cnfs_static_clients;
    8 cnfs_temp_clients;
32768 nfs_maxdata;
    256 nfs_num_rnodes;
1200 nfs_maxdupreqs;
    3 nfs_duptimeout;
    0 nfs_printinter;
8000 tcp_nmbospace;
0700 hidirmode;
0600 hifilemode;
}
```

5.3.7.11 network Section Example for GigaRing Based Systems

The following is an example of a network section for a GigaRing based system:

```
network {
  gether 0 {
    iopath { ring 01; node 02; channel 05; }
    maxusers 1;
    maxinputs 64;
    maxoutputs 64;
  }
  gfddi 0 {
    iopath { ring 01; node 02; channel 04; }
    maxusers 1;
    maxinputs 64;
    maxoutputs 64;
  }
  gatm 0 {
    iopath { ring 01; node 03; channel 01; }
    maxusers 1;
    maxinputs 64;
    maxoutputs 64;
  }
  ghippi 0 {
    iopath { ring 01; node 04; channel 02; }
    maxusers 4;
    maxinputs 64;
    maxoutputs 64;
  }
  gr 0 {
    iopath { ring 01; node 05; }
  }
}
```

5.3.7.12 network Section Example for Model E Based Systems

The following example configures FDDI, HIPPI, and NP devices and only those custom network device specifications (`np_spec`) that are needed:

```
network {
  4 npmaxdevs;
  32 npmaxpaths;
  16 npmaxppd;
```

```
100 npito;
100 npoto;
600 nprto;
  10 nprthresh;
  10 npwthresh;
  4 himaxdevs;
  8 himaxpaths;
  1 fdmaxdevs;
 16 fdmaxpaths;
0700 fddirmode;
0600 fdfilemode;
0700 npdirmode;
0600 npfilemode;
np_spec FEI3FY {
  device type FEI_3FY;
  channel mode 6MB;
  driver type FY;
  driver mode 8;
  device function 0;
  input timeout 100;
  output timeout 100;
  read timeout 600;
}
npdev 0 {
  iopath { cluster 0; eiop 0; channel 030; }
  np_spec FEI3FY;
}
hidev 0 {
  iopath { cluster 0; eiop 3; channel 030; }
  logical path 0 { flags 00; I_field 00; ULP_id 00; }
  flags 00;
  input;
  device type PS_32;
}
hidev 1 {
  iopath { cluster 0; eiop 3; channel 032; }
  logical path 0 { flags 00; I_field 00; ULP_id 00; }
  flags 00;
  output;
  device type PS_32;
}
fddev 0 {
  treq 8;
}
```

```
        padcnt 3;
        maxwrt 10;
        maxrd 10;
        iopath { cluster 0; eiop 0; channel 034; }
    }
}
```

5.3.7.13 network Section Example for Model V Based Systems

The following example configures Ethernet, FDDI, and ATM devices for Model V based systems.

```
network {
    2 himaxdevs;
    4 himaxpaths;
    1 fdmaxdevs;
    1 enmaxdevs;
    0 npmaxdevs;
    2 atmmmaxdevs;
131072 atmarp_recv;
65536 atmarp_send;
    1024 atmarp_entries;

    endev 0 {
        iopath { cluster 1; eiop 0; channel 020; }
    }

    atmdev 0 {
        iopath { cluster 0; eiop 0; channel 020; }
    }

    atmdev 1 {
        iopath { cluster 3; eiop 0; channel 020; }
    }

    fddev 0 {
        iopath { cluster 2; eiop 0; channel 020; }
    }
}
```

5.3.8 revision Section

The `revision` section marks the CSL parameter file with a site-defined name for identification purposes, particularly for programs and other Cray Research products. The `revision` string is set automatically when you use the ICMS.

The `revision` section is specified in the CSL parameter file by the following statement:

```
revision text_string
```

The *text_string* should be a string that is significant for your site and allows you to identify the file.

Run-time Configuration Scripts and Files [6]

This chapter describes scripts and files that are important in the configuration of various aspects of a UNICOS system. These scripts and files are executed or read during system initialization or when changing run levels. For a discussion of system initialization and run-level configuration, see *General UNICOS System Administration*.

Each of the following sections discusses one script or file. The scripts and files appear alphabetically by file name (not path name).

The following scripts and files are discussed:

- `/usr/lib/cron/at.deny` and `/usr/lib/cron/at.allow`
- `/etc/bcheckrc`
- `/etc/brc` and `/etc/coredd`
- `/usr/lib/cron/cron.deny` and `/usr/lib/cron/cron.allow`
- `/etc/cshrc`
- `/etc/config/daemons`
- `/etc/fstab`
- `/etc/gettydefs`
- `/etc/hippi#.arp` (GigaRing based systems only)
- `/etc/group`
- `/etc/hycf.local_network` (Model E based systems only)
- `/etc/inittab`
- `/etc/config/interfaces`
- `/etc/issue`
- `/etc/config/ldchlist`
- `/etc/motd`
- `/etc/netstart`
- `/etc/config/netvar.conf`

- /etc/passwd
- /etc/profile
- /etc/rc
- /etc/config/rcoptions
- /etc/shutdown
- /etc/umountem

6.1 at.deny and at.allow Files

You can alter the `/usr/lib/cron/at.deny` and `/usr/lib/cron/at.allow` files. These files define users who are permitted or excluded from using the `at(1)` command.

Users are permitted to use `at` if their login appears in the file `/usr/lib/cron/at.allow`. If that file does not exist, the file `/usr/lib/cron/at.deny` is checked to determine if the user should be denied access to `at`. If neither file exists, only `root` is allowed to submit a job. An empty `at.allow` file means no user is allowed to use `at`; an empty `at.deny` file means no user is denied the use of `at`.

UNICOS is released with an empty `at.deny` file. The `allow/deny` files consist of one user name per line. Change one file or the other to allow or deny user access to `at`.

6.2 bcheckrc Script

The `/etc/bcheckrc(8)` script performs the commands necessary (such as setting the system time or checking file system consistency) before the `rc` script (see `brc(8)`) is to be executed and the file systems are to be mounted. See the `bcheckrc(8)` man page for more information.

Note: This script is not intended to be modified directly. To modify the execution of this script, change the options in `/etc/config/rcoptions`.

6.3 `brc` and `coredd` Scripts

The `brc(8)` script is used for processing system dumps. It copies system dumps to a separate file system by executing the `coredd(8)` shell script.

Note: These scripts are not intended to be modified directly. To modify the execution of this script, change the options in `/etc/config/rcoptions`.

See the `brc(8)` and `coredd(8)` man pages for more information.

6.4 `cron.deny` and `cron.allow` Files

You can alter the `/usr/lib/cron/cron.deny` and `/usr/lib/cron/cron.allow` files during configuration. These files define users who are permitted or excluded from using the `crontab(1)` command.

Users are permitted to use `cron` if their login appears in the file `/usr/lib/cron/cron.allow`. If that file does not exist, the file `/usr/lib/cron/cron.deny` is checked to determine if the user should be denied access to `cron`. If neither file exists, only `root` is allowed to submit a job. An empty `cron.allow` file means no user is allowed to use `cron`; an empty `cron.deny` file means no user is denied the use of `cron`.

UNICOS is released with an empty `cron.deny` file. The files consist of one user name per line. Change one file or the other to allow or deny user access to `cron`.

6.5 `cshrc` File

The `/etc/cshrc` file is the equivalent of `/etc/profile` (see Section 6.21, page 106) for `csh` (the C shell). The release version of `/etc/cshrc` sets the user's file creation mode, prints the `/etc/motd` file on the user's screen, and tells the user if mail and news exists.

If the `/etc/cshrc` file exists and your login shell is `/bin/csh`, the file is executed by the shell upon login before your session begins. Then, if your login directory contains a file named `.cshrc`, it is executed by the shell before the session begins. See the `cshrc(5)` and `csh(1)` man pages for more information.

6.6 daemons File

The `/etc/config/daemons` file is used by the `sdaemon(8)` command for controlling the starting and stopping of system daemons. (The `sdaemon(8)` command is used by the system startup procedures supplied with UNICOS to start daemon processes necessary for system operation.) To access this file through the UNICOS installation and configuration menu system (ICMS), use the following menu selection:

```
Configure system
->System daemons configuration
  ->System daemons table
```

An entry in the `/etc/config/daemons` file has the following format:

| |
|---|
| <i>daemongroup tag groupstart kill pathname arguments</i> |
|---|

| | |
|--------------------|--|
| <i>daemongroup</i> | The group to which this daemon belongs. |
| <i>tag</i> | A convenient tag to use when referring to the daemon. You can use the name of the daemon's executable file or any other name you choose. |
| <i>groupstart</i> | Indicates whether this daemon should be started when its group is started. A value of YES indicates this daemon will be started as part of its group. A value of ASK indicates that the operator will be asked at startup time if the daemon should be started. Any other value (NO is suggested) indicates that the daemon will not be started as part of its group. A daemon with a <i>groupstart</i> of NO may be started individually. |
| <i>kill</i> | A value indicating how the daemon process may be terminated. |
| <i>pathname</i> | The path name of the daemon executable. |
| <i>arguments</i> | Command-line arguments to be used when starting the daemon. |

A full explanation of the format of the `/etc/config/daemons` file and its use by the `sdaemon(8)` command may be found on the `sdaemon(8)` man page.

The following *daemongroup* values are used by the system startup procedure supplied with UNICOS to start groups of daemons in the following order:

| | |
|------|---|
| SYS1 | Non-networking daemon processes started before the networks are present |
| TCP | Daemon processes related to TCP/IP |
| NFS | Daemon processes related to NFS |
| SYS2 | Non-networking daemon processes started after the networks are present |

You may choose to define additional groups to suit your local configuration.

6.7 *fstab* File

The `/etc/fstab` file (see `fstab(5)`) contains a list of file systems that the `bcheckrc(8)` script checks by invoking the `mfscck(8)` command. UNICOS programs read `/etc/fstab`, but do not write information to it; you must create and maintain the information in this file.

Note: The `mfscck(8)` command was formerly called `genocat(8)`.

Use the following menu selection to change this file:

```
Configure System
  ->File System (fstab) Configuration
```

However, if you want to change the file manually, this section contains the information you need.

The `/etc/fstab` file is an ASCII file. The fields are separated by white space (tabs or spaces); each group is separated from the next by a newline character. Each entry in `fstab` has the following format:

| |
|---|
| <i>filesystem directory type options frequency passnumber</i> |
|---|

The first and last fields are used by `mfscck` (see the `mfscck(8)` man page for more information). See also the routines `getfsent`, `getfsspec`, `getfstype`, and `getfsfile` described on the `getfsent(3)` man page for information on reading records from `/etc/fstab`.

The `/etc/fstab` file contains entries to make mounting file systems easier. When the `mount(8)` command is invoked with only a directory argument or an incomplete argument list, it searches `fstab` for the missing arguments.

The `type` field must have one of the following values. Each value specifies a file system type.

| <u>Type</u> | <u>File system</u> |
|-------------|--------------------------------|
| NCIFS | UNICOS file system |
| NFS | Network file system |
| PROC | <code>/proc</code> file system |

See the `fstab(5)` man page for further information on `fstab`. See also the `mount(8)` man page.

6.8 `gettydefs` File

The `/etc/gettydefs` file contains information used by the `getty(8)` command to set up the speed and terminal settings for a line. `getty` is used for the IOS-E terminals.

The information in `gettydefs` also specifies the appearance of the login prompt.

The `/etc/gettydefs` file is an ASCII file. The fields are separated by pound signs (#); each group is separated from the next by a newline character. Each entry in `gettydefs` has the following format:

label#*initial_flags*#*final_flags*#*login_prompt*#*next_label*

The fields contain the following information:

| | |
|----------------------|---|
| <i>label</i> | String against which <code>getty</code> tries to match its second argument. |
| <i>initial_flags</i> | Initial <code>ioctl(2)</code> settings to which the terminal is to be set if a terminal type is not specified with <code>getty</code> . |
| <i>final_flags</i> | Flags that are set just before <code>getty</code> executes <code>login(1)</code> . |

| | |
|---------------------|--|
| <i>login_prompt</i> | The login prompt. All characters and white space in the field are included in the prompt. |
| <i>next_label</i> | The desired speed, specified by a break character. If this field does not contain the speed, <i>getty</i> searches for the entry with <i>next_label</i> as its <i>label</i> field and sets up the terminal for those settings. |

See the *gettydefs(5)* and *getty(8)* man pages for detailed information on the */etc/gettydefs* file. Also see the *login(1)*, *vi(1)*, and *ioctl(2)* man pages for more information on *gettydefs*.

6.9 *ghippiX.arp* File

The */etc/ghippiX.arp* file lets you configure the hardware addresses of known hosts on local networks that are directly connected to a GigaRing based system via HIPPI. *X* is the ordinal of the HIPPI interface.

For more information, see *UNICOS Networking Facilities Administrator's Guide*.

6.10 *grX.arp* File

The */etc/grX.arp* file lets you configure the hardware addresses of known hosts on local networks that are connected using host-to-host GigaRing interfaces. *X* is the ordinal of the GigaRing interface.

For more information, see *UNICOS Networking Facilities Administrator's Guide*.

6.11 *group* File

The */etc/group* file (see *group(5)*) is provided to translate group names to group IDs and group IDs to names. It is not used for user validation.

The */etc/group* file is an ASCII file that contains the following information for each group:

- Group name
- Encrypted password
- Numerical group ID (GID)
- Comma-separated list of user names allowed in the group

The password field is not used in UNICOS and is set to an asterisk (*). See the `group(5)` man page for more information on the `/etc/group` file and `udb(5)` and *General UNICOS System Administration*, for more information on the user database (UDB) and the `group` file.

6.12 `hycf.local_network` Files

The `/etc/hycf.local_network` files let you configure the hardware addresses of known hosts on local networks that are directly connected to an Model E based system. There is one file for each local network connected to the mainframe.

6.13 `inittab` File

The `/etc/inittab` file controls the actions of the `init(8)` process. It contains information on processes and scripts that are to be executed at the various run levels. See *General UNICOS System Administration*, for a description of UNICOS run levels and run-level configuration.

The `/etc/inittab` file is composed of position-dependent entries that have the following format:

| |
|--|
| <i>id</i> : <i>rstate</i> : <i>action</i> : <i>process</i> |
|--|

Each entry is delimited by a newline character; however, a backslash (\) preceding a newline character indicates a continuation of the entry. Up to 512 characters for each entry are permitted. Comments may be inserted in the process field, using a # sign at the beginning of the field.

The entry fields are as follows:

| <u>Field</u> | <u>Description</u> |
|---------------|--|
| <i>id</i> | 1 to 4 characters that uniquely identify an entry. |
| <i>rstate</i> | Run level in which this entry is to be processed. |

action Keywords that specify how to treat the process in the *process* field. The following actions are recognized by *init*:

| | | |
|-------------|----------|---------|
| boot | bootwait | generic |
| initdefault | off | once |
| ondemand | respawn | sysinit |
| timezone | wait | |

process An *sh*(1) command to be executed.

Be sure to change the time zone entry in the */etc/inittab* file to reflect your site's time zone. The ICMS will copy your time zone entry if you are performing an upgrade installation, but initial installations must make the change to */etc/inittab* manually. For instructions on changing the time zone, see *General UNICOS System Administration*.

For more information about the structure of the */etc/inittab* file, see *inittab*(5) in the *UNICOS File Formats and Special Files Reference Manual*.

6.14 interfaces File

The */etc/config/interfaces* file contains the list of TCP/IP interfaces that may be configured using the *initif*(8) command. The *initif* command is used by the system startup procedures supplied with UNICOS. A full explanation of the format of the */etc/config/interfaces* file may be found on the *initif*(8) man page and in the *UNICOS Networking Facilities Administrator's Guide*. To access this file through the ICMS, use the following menu selection:

```
Configure system
->Network configuration
    ->General network configuration
        ->Network interface configuration
```

6.15 issue File

The */etc/issue* file (see *issue*(5)) is displayed before the login prompt. It is used to echo any important messages users might need to know prior to logging in, such as the following:

```
The system is dedicated.
Please do not log on.
```

See *General UNICOS System Administration*, for more information on the `/etc/issue` file.

6.16 `ldchlist` File

The `/etc/config/ldchlist` file is used by the `rc` system start-up script supplied with UNICOS to initialize the logical device cache to be used during normal system operation. To access this file through the ICMS, use the following menu selection:

```
Configure system
->Disk configuration
   ->logical device cache
```

For more information, see the description of `rc` on the `brc(8)` man page and in *General UNICOS System Administration*.

6.17 `motd` File

The `/etc/motd` file (see `motd(5)`) is the UNICOS message-of-the-day file. It generally contains the UNICOS release level and any important messages that must be conveyed to each user as they log in. It is released containing a warning to unauthorized users. The following is the released version of the `/etc/motd` file:

```
cat /etc/motd
This is a private computer facility.  Access for any reason must be
specifically authorized by the owner.  Unless you are so authorized,
your continued access and any other use may expose you to criminal
and/or civil proceedings.
```

6.18 `netstart` Script

The `netstart` script is executed by the `rc` script to initialize UNICOS networking software. It starts each type of UNICOS networking software (for example, TCP/IP or OSI) by calling the appropriate startup script or command for that software type.

The list of scripts executed by `/etc/netstart` is as follows:

| <u>Command</u> | <u>Description</u> |
|--------------------------------|---|
| <code>/etc/netstart.pre</code> | The script to be run before <code>netstart</code> to accommodate local requirements. |
| <code>/etc/nwmstart</code> | Command used to initialize the underlying network media |
| <code>/etc/tcpstart</code> | Command to initialize TCP/IP software. |
| <code>/etc/nfsstart</code> | Command to initialize the network file system (NFS) software. |
| <code>/etc/ypstart</code> | Command to initialize network information service (NIS) software. (NIS was formerly known as yellow pages.) |
| <code>/etc/netstart.pst</code> | The script to be run after <code>netstart</code> to accommodate local requirements. |

See the `netstart(8)` man page for additional information about the `netstart` script.

6.19 netvar.conf File

The `/etc/config/netvar.conf` file contains a list of command-line arguments for the `netvar(8)` command. These arguments are used by the `/etc/tcpstart` script supplied with UNICOS to initialize TCP/IP kernel variables at system startup. For a complete list of arguments to the `netvar(8)` command and further information about its operation, see the `netvar` man page and the *UNICOS Networking Facilities Administrator's Guide*.

To access this file through the ICMS, use the following menu selection:

```
Configure system
  ->Network configuration
    ->TCP/IP network configuration
      ->kernel parameters
```

6.20 passwd File

The `/etc/passwd` file contains one entry per user. Each entry consists of the login name, encrypted password (this field is set to an asterisk (*) or to `*,pw->pw_age` if password aging is active), user ID (UID), group ID (GID), a

comment field, initial working directory, and the program the user uses as a shell.

The `/etc/passwd` file exists in UNICOS for compatibility with predecessor systems and plays no role in the validation and management of system users. That function is performed by the user database (UDB). For information on the relationship of `/etc/passwd` to the UDB, see *General UNICOS System Administration*, and the `udb(5)`, `udbgen(8)`, and `udbsee(1)` man pages. For the format of `/etc/passwd` and more information on its content, see the `passwd(5)` man page.

6.21 profile File

If the `/etc/profile` file exists, it is executed by the standard shell upon login before your session begins. Then, if your login directory contains a file named `.profile`, it is executed by the shell before the session begins. The file `.profile` is useful for setting exported environment variables.

6.22 rc Script and rcoptions File

The `rc` (see `brc(8)`) script is executed by the `init(8)` command using an entry in the `/etc/inittab` file (see `inittab(5)`) when changing the run level from single-user to multiuser mode.

To access this file through the ICMS, use the following menu selection:

```
Configure system
  ->Startup (/etc/rc) configuration
```

Note: The `rc` script is not intended to be edited directly. To modify the execution of this script, change the options in `/etc/config/rcoptions`. To perform tasks not controlled by `rcoptions`, create the following files:

- `/etc/rc.pre`
- `/etc/rc.mid`
- `/etc/rc.post`

These files are executed before, during, and after the `rc` script.

The UNICOS `rc` script is intended to provide a high degree of system startup configuration without modification of the script. This includes mounting file systems; activating accounting logging, error logging, and system activity

logging; and starting system daemons. To do this, `rc` references a number of subsidiary configuration files and calls various configurable system utilities.

For a complete discussion of the capabilities and configuration of the `rc` script supplied with UNICOS, see the `brc(8)` man page and *General UNICOS System Administration*.

The `/etc/config/rcoptions` file contains a list of environment variable definitions that control the startup configuration of the system. These definitions are read in by the `rc` (see `brc(8)`) script supplied with UNICOS. For a list of environment variables and explanation of their effect on system startup, see the description of the `/etc/rc` script in *General UNICOS System Administration*.

6.23 shutdown Script

The `shutdown(8)` script is a part of the UNICOS operation procedure. It primarily terminates all currently running processes in an orderly and cautious manner. The procedure is as follows:

1. All users logged on the system are notified by a broadcast message to log off the system. You may display your own message at this time. Otherwise, the standard file save message is displayed.
2. All user processes are killed and daemons are shut down.
3. `shutdown` unmounts all file systems.

After the script completes, the dynamic blocks of all file systems are updated before the system is to be stopped (see `ldsync(8)` and `sync(1)`). This must be done before rebooting the system to ensure file system integrity.

After `shutdown` completes, the system is in single-user mode and is essentially equivalent to the state of the system after a reboot procedure. The release version of `shutdown` also shuts down the NQS daemon so that jobs can be recovered when the system reboots.

See *General UNICOS System Administration* for more information on the `shutdown` script.

6.24 `umountem` Script

The `umountem(8)` script unmounts all mounted file systems except the current root file system. It is called by the `shutdown(8)` shell procedure to ensure that all file systems are truly unmounted before going to single-user mode. This script may also be called manually.

Reserved Keywords in CSL [A]

The following keywords have special meaning in the configuration specification language (CSL):

12MB
12MB_LP
6MB
A130
A400
ALL
C90
C90M
CNT
Cray
D90
DA301
DA302
DA60
DA62
DD10
DD11
DD19
DD29
DD3
DD301
DD302
DD304
DD308
DD309
DD310
DD314
DD318
DD39
DD4
DD40
DD41
DD42
DD49
DD50
DD501

DD5I
DD5S
DD60
DD61
DD62
DD6S
DDAS2
DDESDI
DDIMEM
DDLDA
DD_U
DIAG
DX4130
DX8130
EN643
FAULT_RESPONSE
FEI_3
FEI_3FY
FEI_4
FEI_CN
FEI_DS
FEI_UC
FEI_VA
FEI_VB
FEI_VM
FY
GATEWAYS
GUESTMAX
Gword
Gwords
HD16
HD32
HD64
HDDMAX
HDDSLMAX
I/O
IEEE
IMP_3
IMP_4
IMP_5
IMP_6
IMP_7
I_field

LCP
LDCHCORE
LDDMAX
LLC
MDDSLMAX
MP
Mword
Mwords
N130
N400
NBUF
NGRT
NLDCH
NPARTITION
NPBUF
NPLCHCTL
NPOOL
NSC_MP
NTRANSACT
NTYDEV
NYPEDEV
OLNET
PB
PBUFSIZE
PDDMAX
PDDSLMAX
PLCHCORE
PS_32
P_8
RAM
RAW
RD-1
RDDSLMAX
REGT
SBUFSIZE
SDDSLMAX
SMT
SSD
SSDDSLMAX
SSDTMAX
SSDTSLMAX
SSD_E
SSD_F

T3D
T90_config
TAPE_MAX_CONF_UP
TAPE_MAX_DEV
TAPE_MAX_PER_DEV
TCP
ULP_id
VAXBI
VME
XDDMAX
XDDSLMAX
YMP
access
alternate
atmarp_entries
atmarp_recv
atmarp_send
atmdev
atmmxdevs
bank
bias
block
blocks
boot
bootable
both
c100d100
c100d200
c200d200
channel
cluster
clusters
cnfs_static_clients
cnfs_temp_clients
configure
cpu
cpus
cylinder
cylinders
device
disk
dmpdev
driver

dumpinfo
eiop
endev
enmaxdevs
enmaxpaths
fddev
fddirmode
fdfilemode
fdmaxdevs
fdmaxpaths
fiber_vhisp
filesystem
flags
floating
function
gateway
gatm
gether
gfddi
ghippi
gigaring
gr
gr_route
gr_union
group
half
halfs
halves
harp_entries
harp_recv
harp_send
hdd
hidev
hidirmode
hifilemode
himaxdevs
himaxpaths
hippi_disk
hisp
ifield
input
io_connect
io_cpus

io_cpus_mask
iopath
ios_e
iounit
is
ithreshold
ldd
length
logical
logical_machine
lower
lowspeed
lunit
mainframe
maintenance
maxinputs
maxoutputs
maxrd
maxusers
maxwrt
mdd
memory
minor
miop
mixed
mode
mpp
mtu
msps
muxiop
network
nfs3_async_max
nfs3_async_time
nfs_duptimeout
nfs_maxdata
nfs_maxdupreqs
nfs_num_rnodes
nfs_printinter
nfs_static_clients
nfs_temp_clients
nfs_wcredmax
no
no-direction

node
np_spec
npdev
npdirmode
npfilemode
npito
npmaxdevs
npmaxpaths
npmaxppd
npoto
nprthresh
nprto
npwthresh
othreshold
output
owner
ows
padcnt
path
pdd
phase_III
physical
piopaths
point
primary
profile
pseudo
qdd
range
read
register
reserve
revision
rft
ring
rootdev
route
sdd
sdsdev
section
sector
sectors
select

serial
spare
spares
ssdt
start
stream
subsection
swapdev
system
tcp_nmbospace
timeout
tmtype
to
track
tracks
treq
type
unicos
unit
upper
user_channel
vhispl
with
word
words
write
xdd
xdisk

A

- Access request logging, 20
- ACNICE, 11
- Address resolution protocol (ARP), 75
- Adjust-on-exit value , 16
- ARP
 - See "Address resolution protocol", 75
- Async write cache, 23
- Asynchronous I/O headers, 8
- Asynchronous I/O headers reserved for the system , 10
- Asynchronous I/O structures allowed per process, 8
- Asynchronous transfer mode, 78
- Asynchronous transfer mode (ATM) address resolution protocol (ARP), 75
- asynchronously written data , 23
- at.allow file, 96
- at.deny file, 96
- ATM
 - See "Asynchronous transfer mode", 75
- ATM device
 - Model V based, 83
- atmarp_entries parameter, 75
- atmarp_recv parameter, 75
- atmarp_recvspace parameter, 76
- atmarp_send parameter, 76
- atmmxdevs parameter, 78
- Audit criteria logging , 21
- Automatic mixed buffer, 11

B

- Bit matrix multiply functional unit, 31
- Bits per memory chip, 4
- Boot process in CSL, 35
- boot statement, 43

- brc shell script, 97
- Buffer cache factor, 8
- Buffer headers, 52
- Buffer size for NFS data, 76
- Buffers, 77
- Bus Based Gateway, 31

C

- C default, 29
- Cache, 76
- Cache blocks based on memory size, 9
- Cache size of duplicate requests (NFS_MAXDUPREQS), 24
- Callout table entries, 9
- CDLIMIT, 8
- Channel assignments, 49
- Channel declarations
 - GigaRing based systems, 47
 - IOS-E based systems, 49
- Channel information, 41
- channel statement
 - GigaRing based systems, 47
 - Model E based systems, 49
- chdir request logging, 21
- Checkpoint and restart buffers, 9
- Chip size, 4
- CHIPSZ , 4
- Client handles in CNFS, 24
- Client handles in CNFS, 24
- Client handles in NFS, 24
- clist, 9
- Clock period, 6
- cluster, 50
- cluster statement, 42
- Clusters , 41

- See "Configuration specification language (CSL) parameter file", 42
- CNFS_STATIC_CLIENTS, 24
- cnfs_static_clients, 25
- cnfs_static_clients parameter, 76
- CNFS_TEMP_CLIENTS, 24
- cnfs_temp_clients, 25
- cnfs_temp_clients parameter, 76
- Comments in CSL, 35
- COMPART_ACTIVE_DEFAULT, 17
- COMPART_VALID_DEFAULT, 17
- config.h configuration file, 7
- config.h configuration file, 1
 - CNFS_STATIC_CLIENTS, 24
 - CNFS_TEMP_CLIENTS, 24
 - MSGMNI, 15
 - MSGSSZ, 16
 - NFS kernel parameters, 23
 - NFS3_ASYNC_TIME, 23
 - NFS_PRINTINTER, 24
 - NFS_STATIC_CLIENTS, 24
 - NPBUF, 10
 - NPLCHCTL, 10
 - security log (SLG_STATE), 22
 - SLG_STATE, 22
 - static client handles in CNFS (CNFS_STATIC_CLIENTS), 24
 - static client handles in NFS (NFS_STATIC_CLIENTS), 24
 - temporary client handles (CNFS_TEMP_CLIENTS), 24
 - temporary client handles (NFS_TEMP_CLIENTS), 24
 - U_MAXPACK, 11
- config.mh configuration file for UNICOS kernel subsystems, 27
- config.mh configuration file for UNICOS kernel subsystems, 1
 - CONFIG_CRAYLIBS, 32
- CONFIG_BBG, 31
- CONFIG_BMM, 31
- CONFIG_CPP, 29
- CONFIG_CPSAVE, 29
- CONFIG_CRAYLIBS, 32
- CONFIG_CRL, 31
- CONFIG_CVT, 31
- CONFIG_DFS, 31
- CONFIG_DIAGDIR, 27
- CONFIG_DM parameter, 31
- CONFIG_ELS, 31
- CONFIG_FQUOTAS, 31
- CONFIG_GCC, 29
- CONFIG_GEN_SEGDIR, 29
- CONFIG_GENBIN, 30
- CONFIG_GENCMDS, 30
- CONFIG_GENPROD_RULES, 30
- CONFIG_HPI3, 31
- CONFIG_HSX, 31
- CONFIG_ID, 27
 - GigaRing based systems, 28
 - Model E based systems, 29
- CONFIG_IOS-F
 - GigaRing based systems, 28
 - Model E based systems, 29
- CONFIG_IOSA_SN, 29
- CONFIG_IOSB_SN, 29
- CONFIG_IPI3, 31
- CONFIG_KERBEROS, 31
- CONFIG_LD_STD_DIR parameter, 32
- CONFIG_MIXED, 31
- CONFIG_MIXEDTARGET, 31
- CONFIG_MK
 - GigaRing based systems, 28
 - Model E based systems, 29
- CONFIG_MPP, 31
- CONFIG_MPP_CPP, 30
- CONFIG_NETMON parameter, 31
- CONFIG_NETTOLS, 32
- CONFIG_NFS parameter, 32
- CONFIG_NFS3, 32
- CONFIG_NFSKRB, 32
- CONFIG_NIOS, 29
- CONFIG_NODE, 28
- CONFIG_OWS, 32
- CONFIG_PACKAGE, 30

-
- CONFIG_PATH, 30
 - CONFIG_RLS_MAJOR, 30
 - CONFIG_RLS_MINOR, 30
 - CONFIG_RLS_REVISION, 30
 - CONFIG_RPC, 32
 - CONFIG_SN, 28
 - CONFIG_SUPPORT_DIR, 30
 - CONFIG_SYS, 28
 - CONFIG_TAPE, 32
 - CONFIG_TARGET, 28
 - CONFIG_TCP, 32
 - CONFIG_TMPDIR, 28
 - CONFIG_TRUSTED, 32
 - CONFIG_VERSION, 28
 - CONFIG_X11, 32
 - CONFIG_XLIBS, 30
 - CONFIG_XLIBTARGET, 30
 - Configuration
 - shell scripts, 104, 106
 - Configuration logging, 20
 - Configuration specification language
 - boot-time equivalents for NFS kernel
 - parameters, 25
 - Configuration specification language (CSL)
 - parameter file, 1, 33, 52, 75–77
 - asynchronous transfer mode (ATM) address
 - resolution protocol (ARP), 75
 - ATM devices, 78
 - atmarp_entries parameter, 75
 - atmarp_rcv parameter, 75
 - atmarp_send parameter, 75
 - atmmaxdevs parameter, 78
 - boot process and, 35
 - buffer headers, 52
 - buffer size, 76
 - cnfs_static_clients parameter, 76
 - cnfs_temp_clients parameter, 76
 - Cray NFS clients, 76
 - customized network device prototypes for
 - Model E based systems, 79
 - device formats
 - GigaRing based systems, 84
 - IOS-E based systems, 85
 - disk, 58
 - disk parameters
 - common, 52
 - GigaRing based systems, 53
 - Model E based systems, 53
 - dumpinfo section, 37
 - duplicate request cache, 76
 - endev device, 83
 - error message syntax, 36
 - Ethernet devices, 78
 - fault tolerance, 51
 - FAULT_RESPONSE parameter, 51
 - fddev device, 83
 - fdmaxdevs parameter, 78
 - fiber distributed data interface (FDDI)
 - (FCA-1) channel adapter, 78
 - gatm device, 83
 - gether device, 83
 - gfddi device, 83
 - ghippi device, 83
 - GigaRing device types, 83
 - gigaring section, 39
 - gr_route subsection, 39
 - gr_union subsection, 40
 - gr device, 83
 - guest resource table entries, 52
 - guest systems, 51
 - GUESTMAX parameter, 51
 - harp_entries parameter, 76
 - harp_rcv parameter, 76
 - harp_send parameter, 76
 - HDDMAX parameter, 53
 - HDDSLMAX parameter, 53
 - hi device, 83
 - hidirmode parameter, 76
 - hifilemode parameter, 76
 - High Performance Parallel Interface, 78
 - High Performance Parallel Interface (HIPPI)
 - , 76
 - himaxdevs parameter, 78
 - himaxpaths parameter, 78
 - HIPPI directory mode, 76

- HIPPI disk device (HDD) slices, 53
- HIPPI disk devices (HDDs), 53
- hippi_disk device type , 58
- interface configuration templates , 81
- ios_ section
 - clusters, 42
 - IOP declarations, 42
- ios_e section, 41
 - HISP declarations, 43
 - IOP boot declarations, 43
- IPI-2 disk type, 59
- ithreshold parameter, 77
- ldcache blocks, 51
- ldcache headers, 51
- LDCHCORE parameter, 51
- LDDMAX parameter, 52
- logical disk devices (LDDs), 52
- low-speed channel communication devices, 78
- mainframe configuration
 - menu, 44
- mainframe section, 44
 - channel declarations (GigaRing based systems), 47
 - channel declarations (Model E based systems), 49
 - common parameters, 45
 - CPUs, 45
 - mainframe cluster registers, 45
 - memory size, 46
- maximum limits parameters, 51
- maxinputs parameter, 77
- maxoutputs parameter, 77
- maxusers parameter, 77
- MDDSLMAX parameter, 52
- mirrored disk device (MDD) slices, 52
- NBUF parameter, 52
- network device types
 - GigaRing based, 83
 - Model E based, 83
 - Model V based, 83
- network parameters
 - common, 75
 - GigaRing based, 77
 - Model V based, 78
 - network section, 74
 - menu, 74
 - NFS requests, 76
 - nfs3_async_max parameter, 77
 - nfs3_async_time parameter, 77
 - nfs_duptimeout parameter, 76
 - nfs_maxdata parameter, 76
 - nfs_maxdupreqs parameter, 76
 - nfs_num_rnodes parameter, 76
 - nfs_printinter parameter, 76
 - nfs_static_clients parameter, 76
 - nfs_temp_clients parameter, 76
 - nfs_wcredmax parameter, 76
 - NGRT parameter, 52
 - NLDCH parameter, 51
 - NPBUF parameter, 51
 - npdev device, 83
 - npfilemode parameter, 78
 - npmaxdevs parameter, 78
 - npmaxpaths parameter, 79
 - npmaxppd parameter, 79
 - npoto parameter, 79
 - nprthresh parameter, 79
 - nprto parameter, 79
 - npwthresh parameter, 79
 - othreshold parameter, 77
 - output time-out, 79
 - parameter file sections, 36
 - common, 44, 50, 55, 74
 - GigaRing based systems, 39
 - Model E and Model V based systems, 41
 - Model E based systems, 37
 - PDDMAX parameter, 52
 - PDDSLMAX parameter, 52
 - physical devices, 53
 - physical disk devices (PDDs), 52
 - physical I/O buffers, 51
 - physical slices, 53
 - RAM device type , 58
 - RAM disk device (RDD) slices, 52
 - RDDSLMAX parameter, 52

- revision section, 93
 - rnode table for NFS, 76
 - SDDSLMAX parameter, 52
 - semicolon in, 35
 - Socket space , 75, 76
 - SSD device type , 58
 - SSD-T90 device type, 58
 - SSD-T90 devices, 53
 - SSD-T90 slice allocation, 53
 - SSDD slices, 53
 - SSDDSLMAX parameter, 53
 - SSDTMAX parameter, 53
 - SSDTSLMAX parameter, 53
 - statement terminator, 35
 - striped disk device (SDD) slices, 52
 - syntax
 - comments, 35
 - constants, 34
 - identifier, 33
 - operators, 35
 - separators, 35
 - table size parameters, 51
 - TAPE_MAX_CONF_UP parameter, 51
 - TAPE_MAX_DEV parameter, 51
 - TAPE_MAX_PER_DEV parameter, 51
 - TCP/IP managed memory buffers, 77
 - tcp_numbspace parameter, 77
 - termination of statements, 35
 - token classes, 33
 - unicos section, 50
 - menu, 51
 - usage, 35
 - verification of configurations, 35
 - XDDMAX parameter, 53
 - XDDSLMAX parameter, 53
 - xdisk, 58
 - /CONFIGURATION, 35
 - Console for MLS administration
 - (SYSTEM_ADMIN_CONSOLE), 22
 - CONSOLE_MSG, 17
 - Constants in CSL, 34
 - Contraction value, 13
 - coredd shell script, 97
 - coremap initial dynamic kernel memory units, 13
 - CPP, 30
 - CPP variable, 29
 - CPUs, 6, 45
 - Cray Data Migration Facility (DMF), 31
 - Cray floating-point and IEEE), 31
 - Cray NFS clients, 76
 - CRAY SV1, CRAY J90 and CRAY J90se
 - support, 31
 - CRAY T3D system, 31
 - Cray Visualization Toolkit, 31
 - cray-t3e assembler, 29
 - Cray/REELlibrarian, 31
 - Cray/REELlibrarian activity logging, 21
 - CRL
 - See "Cray/REELlibrarian", 31
 - cron.allow file, 97
 - cron.deny file, 97
 - Cross-targeted libraries, 30
 - cshrc file, 97
 - CSL
 - See "Configuration specification language (CSL) parameter file", 1
 - CSL reserved keywords, 109
 - Customized network device prototypes for
 - Model E based systems, 79
 - CVT
 - See "Cray Visualization Toolkit", 31
 - Cycles per second, 6
- D**
- daemons file, 98
 - Data Migration Facility
 - See "Cray Data Migration Facility", 31
 - data read or written by way of NFS, 23
 - Data written asynchronously, 23
 - DCE
 - See "Distributed Computing Environment", 31
 - Declarations in CSL
 - HISP, 43

IOP, 42
 DECLASSIFY_DISK, 17
 DECLASSIFY_PATTERN, 17
 DELAY_MULT, 18
 Destination routing, 40
 /dev/comm, 78
 /dev/fddi*, 78
 DEV_ENFORCE_ON, 18
 Device labeling, 18
 device type statement, 80
 DFS
 See "Distributed Computing Environment", 31
 Directory name look-up cache (DNLC)
 number of entries, 9
 path name , 9
 Disable a parameter, 7
 DISABLE_ACCT, 18
 DISABLE_TIME, 18
 Discretionary access control (DAC) logging, 21
 Discretionary access violations, 20
 disk, 57, 58
 Distributed Computing Environment (DCE)
 distributed file system (DFS)
 (CONFIG_DFS), 31
 DMA driver, 40
 DMF
 See "Cray Data Migration Facility", 31
 DMODE , 8
 Dump requirements, 37
 dumpinfo section, 37
 Duplicate request cache, 76
 Duplicate request cache size , 24
 Duplicate time-out, 23
 Dynamic kernel memory parameters, 12

E

econfig command and configuration
 verification, 36
 eiop statement, 42
 Enable a parameter, 7
 enmaxdevs, 78

Environment variables
 definitions in rcoptions file, 107
 Error exit maximum, 15
 Error message syntax for CSL, 36
 Error messages and CSL, 36
 /etc/config/param, 33
 Ethernet device, 83
 Ethernet devices, 78
 Examples
 filesystem section, 69
 CRAY SSD-T90, 72
 GigaRing based, 70, 72
 SSD for Model E based, 74
 third-party I/O, 71
 ios_e section, 43
 mainframe section
 GigaRing based systems, 50
 Model E based systems, 50
 unicos
 CRAY SSD-T90 (additional parameters), 54
 GigaRing based systems, 54
 Model E based systems, 55
 exec operations, 9
 exec operations with maximum arguments , 11
 executables (CONFIG_GENCMDS), 30
 Expansion units, 12
 EXTDCORE, 8
 Extended core file naming, 8

F

Failed login attempt logging, 22
 Failed logins, 18
 Fair share scheduler, number of users, 10
 FAULT_RESPONSE parameter, 51
 FCN bit and range numbers, 54
 fddev device, 83
 fddev device format, 85
 FDDI device, 83
 FDDI channel, 78
 fddirmode, 78

fdfilemode, 78
 fdmaxdevs parameter, 78
 fdmaxpaths, 78
 Fiber distributed data interface (FDDI) (FCA-1)
 channel adapter, 78
 File mode permission, 78
 File quotas, 31
 File retrieval mode, 8
 File size in blocks, 8
 File system log records, 11
 File systems open concurrently , 9
 File systems that can be mounted, 10
 file transfer logging, 20
 File-lock regions (NFLOCKS) , 9
 Files that can be open at one time system-wide
 (NFILE) , 9
 filesystem section, 55
 Flush buffered data to target I/O devices, 8
 FLUSHONPANIC, 8
 FORCED_SOCKET, 18
 FSETID_RESTRICT parameter, 18
 FSLG_BUFSIZE, 11
 fstab file, 99

G

gatm device, 83
 gatm device format, 84
 General configuration, 7
 Generation parameters, 27
 Generation software (CONFIG_GENBIN), 30
 Generation software directory paths, 30
 gether device, 83
 gether device format, 84
 getty command, 100
 gettydefs file, 100
 gfddi device format, 85
 ghippi device, 83
 ghippi device format, 85
 ghippi#.arp file, 101
 GigaRing configuration parameters (gigaring
 section), 39

GigaRing device types, 83
 gigaring section, 39
 GigaRing support
 GigaRing based systems, 28
 Model E based systems, 29
 gr device, 83
 gr device format, 85
 gr_route subsection, 40
 gr_union subsection, 40
 group file, 101
 grX.arp file, 101
 Guest resource table entries, 52
 Guest systems, 51
 GUESTMAX parameter, 51

H

Handles, 76
 harp_entries parameter, 76
 harp_send parameter, 76
 Hash entries (NHBUF_FCTR) , 9
 HDD
 See "High performance parallel interface
 disk devices", 53
 HDDMAX parameter, 53
 HDDSLMAX parameter, 53
 Hertz, 6
 hi device, 83
 hidev device format, 86
 hidirmode parameter, 76
 hifilemode parameter, 76
 High Performance Parallel Interface, 78
 High Performance Parallel Interface (HIPPI) , 76
 High performance parallel interface (HIPPI)
 disk devices, 53
 High-speed (HISP) , 41
 High-speed channel (HISP)
 declarations in CSL, 43
 High-speed HIPPI device, 83
 himaxdevs parameter, 78
 himaxpaths parameter, 78

HIPPI directory mode, 76
 hippi_disk device type, 58
 HISP
 See "High-speed (HISP) channel information", 41
 HPN bit and range numbers, 54
 HSX device driver, 31
 hycf.* file, 102
 HZ, 6

I

I/O clusters, 41
 I/O error logging, 22
 I/O module 0 channels, 48
 I/O node unit bit and range numbers, 53
 I/O processor
 boot declarations, 43
 I/O processor (IOP)
 declarations in CSL, 42
 I/O processors, 41
 ICMS
 See "Installation and configuration menu system (ICMS)", 1
 Identifiers in CSL, 33
 IEEE and Cray floating-point, 31
 In-core inode table, 9
 In-core inodes, 9
 In-core inodes for NC1FS (file system migration parameter) , 11
 In-core quota entries, 10
 Informational messages and CSL, 36
 initif command
 use of interfaces file, 103
 inittab file, 102
 Install tool
 See "Installation and configuration menu system (ICMS)", 1
 Installation, 1
 Installation and configuration menu system (ICMS), 1
 Integrity code in secure.c, 18

Interface configuration templates, 81
 interfaces file, 103
 Internal routing, 39
 Internet Protocol (IP) layer activity logging, 21
 Introduction, 1
 IOC
 See "I/O clusters", 41
 ION
 See "I/O node", 53
 IOP
 See "I/O processors", 41
 iopath, 61
 IOS-E hardware topology definition, 41
 IOS-E machines, 29
 IOS-E serial numbers, 29
 ios_e section, 41
 iounit, 60
 IPC kernel parameters, 15
 IPI-2 disk type, 59
 IPI-3/HIPPI packet driver capability, 31
 IPI-3/IPI capability, 31
 IPN bit and range numbers, 54
 issue file, 103
 ithreshold parameter, 77

J

job end logging , 21
 Job start logging, 21

K

K_OPEN_MAX, 11
 Kerberos support, 31
 Kernel generation parameters, 27
 Kernel memory parameters, 12
 Kernel parameters, 1
 common, 8
 Kernel subsystem parameters, 27
 Kernel subsystems

See "config.mh configuration file ", 1
 Keyword identifiers in CSL, 34
 KM_CHM_PADSZ parameter, 12
 KM_EXPAND_UNITS, 12
 KM_NO_THRASH parameter, 12
 KM_UNITS parameter, 13
 KM_WPU parameter, 13
 KM_WPU_SHIFTC parameter, 13

L

ldcache blocks, 8
 Ldcache headers, 51
 LDCHCORE, 8, 39
 LDCHCORE parameter, 51
 ldchlist file, 104
 LDD
 Logical disk devices , 52
 ldd logical device, 66
 LDDMAX parameter, 52
 Libraries, cross-targeted, 30
 Limits parameters, 52
 link call logging, 20
 Link violation logging, 21
 LINK_MAX , 8
 Links to a file, 8
 Log records for the file system, 11
 LOGDELAY, 18
 Logging parameters for MLS, 20
 Logical device cache headers, 9
 Logical devices in CSL, 34
 Logical disk devices (LDDs), 52
 logical path statement, 87
 Login account disabling, 18
 Login attempts, 17
 login attempts, 18
 Login failure logging, 22
 Low-speed channel, 83
 Low-speed channel communication devices, 78

M

Machine-specific characteristics, 1
 Mainframe cluster registers, 45
 Mainframe memory size, 46
 mainframe section, 44
 Mainframe serial number, 4
 Mainframe subtype, 5
 Mainframe type, 5
 Major release number, 30
 Managed memory buffers, 23, 77
 Mandatory access request logging, 22
 MAX_UNLINKED_BYTES parameter, 11
 MAXASYN, 8
 maxinputs parameter, 77
 MAXLOGS, 18
 maxoutputs parameter, 77
 MAXPIPE , 8
 MAXRAH, 8
 MAXSLEVEL, 18
 maxusers parameter, 78
 MAXUSRERR, 15
 MAXUSRORE parameter, 15
 MAXUSRPRE parameter, 15
 MDD
 See "Mirrored disk device ", 52
 mdd mirrored device, 67
 MDDSLMAX parameter, 52
 MEMORY, 4
 Memory banks, 4
 Memory buffers, 77
 Memory buffers for TCP/IP, 23
 Memory padding, 12
 memory range, 39
 Memory size, 46
 Menu system
 See "Installation and configuration menu system (ICMS)", 1
 Message header number , 16
 Message interval, 24
 Message queue maximum bytes, 15
 Message queue maximum number, 15

Message segment number, 16
 Message segment size, 16
 Message size , 15
 Messages and CSL, 36
 MFSUBTYP, 5
 MFTYPE, 5
 Minor release number, 30
 MINSLEVEL, 18
 Mirrored disk device (MDD) slices, 52
 Mixed buffer , 11
 Mixed mode (Cray floating-point and IEEE), 31
 mkdir call logging, 20
 mkdir violation logging, 22
 MLS, 7
 MLS administration console, 22
 MLS kernel parameters, 17
 MLS_INTEGRITY, 18
 MLS_OBJ_RANGES, 18
 motd file, 104
 MPN bit and range numbers, 54
 MSGMAX, 15
 MSGMNB, 15
 MSGMNI, 15
 msgque table size , 15
 MSGSEG, 15
 MSGTQL, 16
 Multilevel security (MLS), 7
 muxiop statement, 42

N

Name of the system, 28
 NASYN, 8
 NBANKS, 4
 NBLK_FCTR, 8
 NBUF parameter, 52
 NBUF_FCTR, 8
 NC1FS in-core inodes , 11
 NC1INODE, 9, 11
 NC1MINRAW, 11
 NC_NAMLEN, 9
 NC_SIZE, 9

NCALL, 9
 NCLIST, 9
 NCPU, 6
 NCRBUF, 9
 netvar.conf file, 105
 Network access violation logging, 22
 Network device prototypes for Model E based systems, 79
 Network device types
 Model E based, 83
 Network File System, 32
 Network File System with Kerberos authentication, 32
 Network monitor, 32
 Network parameters
 common, 75
 IOS-E, 78
 menu, 75
 network section, 74
 Network testing tools, 32
 NEXECS, 9, 11
 NFILE, 9
 NFLOCKS, 9
 NFS
 See "Network File System", 32
 NFS activity logging , 20
 NFS kernel parameters
 See "config.h configuration file", 23
 NFS kernel parameters and their CSL boot-time equivalents , 25
 NFS requests, 76
 NFS version 3 Protocol, 32
 NFS-mounting of a remote file system in read-write mode , 18
 NFS3_ASYNC_MAX, 23, 25
 nfs3_async_max, 25
 nfs3_async_max parameter, 77
 NFS3_ASYNC_TIME, 23
 nfs3_async_time, 25
 nfs3_async_time parameter, 77
 nfs_duptimeout, 25
 NFS_DUPTIMEOUT parameter, 23

nfs_duptimeout parameter, 76
 NFS_MAXDATA, 23
 nfs_maxdata, 25
 nfs_maxdata parameter, 76
 NFS_MAXDUPREQS, 23
 nfs_maxdupreqs, 25
 nfs_maxdupreqs parameter, 76
 nfs_num_rnodes parameter, 76
 nfs_num_rnodes&, 25
 NFS_PRINTINTER, 24
 nfs_printinter, 25
 nfs_printinter parameter, 76
 NFS_REMOTE_RW_OK parameter, 18
 NFS_SECURE_EXPORT_OK, 18
 NFS_STATIC_CLIENTS, 24
 nfs_static_clients, 25
 nfs_static_clients parameter, 76
 NFS_TEMP_CLIENTS, 24
 nfs_temp_clients, 25
 nfs_temp_clients parameter, 76
 nfs_wcredmax parameter, 77
 NGRT parameter, 52
 NHBUF_FCTR, 9
 Nice value, 11
 NINODE, 9
 NLDCH, 9
 NLDCH parameter, 51
 NLDMAP, 9
 NMNT, 9
 NMount parameter, 9
 NMT, 12
 Node name of the system, 28
 Node number, 24
 non-inode security system call logging, 21
 Non-tape daemon tapes., 12
 np_spec statement, 79
 NPBUF, 10
 NPBUF parameter, 51
 npdev device, 83
 npdev device format, 86
 npdirmode, 78
 npfilemode parameter, 79
 npito, 79

NPLCHCTL, 10
 npmaxdevs, 78
 npmaxpaths parameter, 79
 npmaxppd parameter, 79
 npoto parameter, 79
 NPROC, 10
 nprthresh parameter, 79
 nprto parameter, 79
 NPTY, 10
 npwthresh parameter, 79
 NQS activity logging, 20
 NQUOTA, 10
 NSESS, 10
 NSIDEBUF, 10
 NSU_ASYN, 10
 NTEXT, 10
 Number of CPUs, 6
 NUSERS, 10

O

Object identifiers in CSL, 34
 Object ranges, 18
 Offline file retrieval mode, 8
 ONFIG_CAM_CPP_LOC, 29
 Online diagnostics location, 27
 Online tape capability, 32
 Online tape parameters, 51
 Open files per process, 11
 Operand range error maximum, 15
 Operating mode information, 41
 Operating system identification, 28

- GigaRing based systems, 28
- Model E based systems, 29

 Operator action logging, 21
 Operator workstation, 32
 Operators in CSL, 35
 othreshold parameter, 77
 Output time-out, 79
 OVERWRITE_COUNT, 18
 OWS

See "Operator workstation", 32
 OWS path names, 41

P

Packaging, 30
 Panic messages and CSL, 36
 Parameter file, 33
 Parameter file sections
 See "Configuration specification language (CSL) parameter file", 36
 Partition cache blocks, 11
 Partition cache headers, 10
 PASS_MAXSIZE, 18
 PASS_MINSIZE, 19
 passwd file, 105
 Password size maximum, 19
 Password size minimum, 19
 Passwords enabled, 19
 Path name access tracking, 22
 Path selection, 39
 PDD
 See "Physical disk devices", 52
 pdd physical device, 66
 PDDMAX parameter, 52
 PDDSLMAX parameter, 52
 Permissions, 78
 Physical devices, 53
 Physical devices in CSL, 34
 Physical disk devices (PDDs), 52
 Physical I/O buffers, 51
 Physical I/O error logging, 22
 Physical I/O headers (NPBUF) , 10
 Physical memory size, 4
 Physical slices, 53
 Pipe device block allocation, 8
 pipes , 18
 PLCHCORE, 10, 39
 PRIV_SU, 19
 Privilege logging, 21
 Process error thresholds, 15

Processes that may be active simultaneously (NPROC) , 10
 profile file, 106
 Program range error maximum, 15
 Programming environment parameters, 27, 32
 Prototype network devices for Model E based systems, 79
 Pseudo terminals (tty/pty pairs), 10

Q

qdd physical device, 67
 Quota entries, 10
 Quotas for files, 31

R

RAID bit and range numbers, 54
 RAM
 See "Random access memory ", 39
 RAM device type, 58
 Random access memory (RAM) disk, 39
 Random access memory (RAM) disk device (RDD) slices, 52
 RANDOM_PASS_ON, 19
 Range numbers
 unit bit and , 54
 rcoptions file, 107
 RDD
 See "Random access memory (RAM) disk device ", 52
 RDDSLMAX parameter, 52
 Read down on pipes , 20
 Read-ahead blocks per read operation, 8
 Release number, 30
 release number (CONFIG_RLS_REVISION), 30
 Remote process control system, 32
 Remove request logging, 20
 Repeatable relocatables capability, 30
 Request cache, 76

Request cache size, 24
 Reserved keywords in CSL, 109
 Restart and checkpoint buffers, 9
 Retired security logs, 21
 Retrieval mode for offline files, 8
 Revision identifiers in CSL, 34
 revision section, 93
 rm call logging, 20
 rm violation logging, 22
 rmdir call logging, 20
 rmdir violation logging, 22
 Rnode number, 24
 Rnode table for NFS, 76
 Root privilege, 19
 ROOTDIR, 50
 Routing, 39
 RPC
 See "Remote process control system", 32

S

SANITIZE_PATTERN, 19
 scrub pattern for disks, 19
 Scrubbing of data, 20
 sdaemon command
 use of daemons file, 98
 SDD
 See "Striped disk device ", 53
 sdd striped device, 67
 SDDSLMAX parameter, 52
 Secure file system export, 18
 SECURE_MAC parameter, 19
 SECURE_OPERATOR_CONSOLE, 19
 SECURE_PIPE, 20
 SECURE_SCRUB, 20
 SECURE_SYSTEM_CONSOLE parameter, 20
 security, 22
 Security level minimum, 18
 Security levelmaximum, 18
 Security log buffer size, 20
 Security log name, 20
 Security log size, 22

security logs location, 20
 Security logs, retired, 21
 SEMAEM, 16
 Semaphore individual value, 17
 Semaphore sets, 16
 Semaphores in the system, 16
 Semaphores per set, 17
 SEMMNI, 16
 SEMMNS, 16
 SEMMNU, 16
 SEMMSL, 16
 semop system call adjust-on-exit value , 16
 semop system call operations, 17
 SEMOPM, 17
 SEMUME, 17
 SEMVMX, 17
 Separators in CSL, 35
 Serial number, 28
 Sessions open simultaneously, 10
 setuid / setgid files , 18
 setuid system call logging, 22
 Shared memory identifiers, 14
 Shared memory kernel parameters, 14
 Shared memory parameters, 14
 Shared segment size maximum, 14
 shared segment size minimum, 14
 Shared segments , 14
 Shared-text programs that can be running
 simultaneously, 10
 shmем table size , 14
 SHMMAX, 14
 SHMMIN, 14
 SHMMNI, 14
 SHMSEG, 14
 shutdown shell script, 107
 Side door buffer, 10
 SIGCPULIM signal, number of CPU seconds
 that a process or session may use following
 , 11
 SLG_ACT_NFS, 20
 SLG_ACT_NQS, 20
 SLG_ALL_NAMI, 20

- SLG_ALL_RM, 20
- SLG_ALL_VALID, 20
- SLG_BUFSIZE, 20
- SLG_CF_UNICOS, 20
- SLG_DIR, 20
- SLG_DISCVconfig.h configuration file
discretionary access violations
(SLG_DISCV), 20
- SLG_FILE, 20
- SLG_FILEXFR, 20
- SLG_FPREFIX, 20
- SLG_JEND, 21
- SLG_JSTART, 21
- SLG_LINKV, 21
- SLG_LOG_AUDIT, 21
- SLG_LOG_CHDIR, 21
- SLG_LOG_CRL, 21
- SLG_LOG_DAC parameter, 21
- SLG_LOG_IPNET parameter, 21
- SLG_LOG_OPER, 21
- SLG_LOG_PRIV, 21
- SLG_LOG_SECSYS parameter, 21
- SLG_LOG_SHUTDOWN, 21
- SLG_MANDV, 21
- SLG_MAXSIZE parameter, 22
- SLG_MAXSIZEconfig.h configuration file
, 22
- SLG_MKDIRV, 22
- SLG_NETWV, 22
- SLG_PATH_TRACK, 22
- SLG_PHYSIO_ERR, 22
- SLG_REMOVEV, 22
- SLG_RMDIRV, 22
- SLG_STATE, 22
- SLG_SUID_RQ, 22
- SLG_SULOG, 22
- SLG_T_PROC, 22
- SLG_USER, 22
- SLGOFF, 21
- Slice structures , 9
- Slices in CSL, 34
- SN, 4
- sn.h configuration file, 1, 3
- Socket space , 75, 76
- sockets, 18
- Spare chip configuration, 49
- SSD, 38, 43, 44, 49, 50
- SSD device type, 58
- SSD logical device cache headers, 9
- SSD side door buffer, 10
- SSD-T90 device type, 58
- SSD-T90 devices, 53
- SSD-T90 slice allocation, 53
- SSDDSLMAX parameter, 53
- SSDT, 56
- SSDTMAX parameter, 53
- SSDTSLMAX parameter, 53
- Static client handles in CNFS
(CNFS_STATIC_CLIENTS), 24
- Static client handles in NFS
(NFS_STATIC_CLIENTS), 24
- Striped disk device (SDD) slices, 52
- Structures in the system, 16
- su command attempt logging, 22
- Subdirectory permissions, 78
- Subsystem parameters, 27
- Subsystems
See "config.mh configuration file ", 1
- Support software directory, 30
- System compartments, 22
- System dump parameters , 38
- system high/system low MAC, 19
- System parameters, 27
- System shutdown logging, 21
- System workstation
See "Operator workstation", 32
- SYSTEM_ADMIN_CONSOLE, 22
- SYSVCOMPS, 22

T

- Table size parameters, 51
- Tape capability, 32
- Tape devices, 51

Tape parameters, 51
TAPE_MAX_CONF_UP parameter, 51
TAPE_MAX_DEV parameter, 51
TAPE_MAX_PER_DEV parameter, 51
Target information, 41
Target of the system to generate, 28
TCP/IP, 7
TCP/IP kernel parameters, 23
TCP/IP managed memory buffers, 23, 77
TCP/IP network system, 32
TCP_NMBSPACE, 23
tcp_numbspace parameter, 77
Temporary client handles in CNFS, 24
temporary client handles in CNFS
 (CNFS_TEMP_CLIENTS), 24
Temporary client handles in NFS, 24
temporary client handles in NFS
 (NFS_TEMP_CLIENTS), 24
Temporary-file directory, 28
Terminal I/O buffers, 9
Time out, 79
tmttype, 60
Trusted process activity logging, 22
Trusted UNICOS, 32
tty/pty pairs, 10
Tunable parameters, 51

U

U_MAXDEV, 12
U_MAXPACK, 11, 12
umountem shell script, 108
Undo entries per process, 17
UNICOS installation, 1

unicos section, 50
UNICOS statement, 50
UNICOS/mk operating system
 GigaRing based systems, 28
 IOS-E based systems, 29
Unique files that can be open at one time, 9
Unlinked files for checkpoint and restart, 11
user devices, 12
User number defined for the fair share
 scheduler, 10
user packets, 11, 12
uts kernel, 31

V

Version name, 28

W

Words per unit of dynamic kernel memory, 13
Write cache, 23

X

xdd, 60
XDDMAX parameter, 53
xddphysical device, 67
XDDSLMAX parameter, 53
xdisk, 57, 58
XTRASEC, 11