UNICOS® Configuration Administrator's Guide

S–2303–10011

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

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.

New Features

UNICOS® Configuration Administrator's Guide S–2303–10011

This document was updated for the UNICOS 10.0.1.1 operating system release, specifically:

- The `NSIDEBUF` kernel parameter of the `config.h` file was changed so that it is now valid and user changable on Cray SV1ex systems.
- The `BTE_STATE` keyword was removed from the CSL parameter file because it was only valid during development.
- On GigaRing based systems, the common disk parameters `PDDMAX` and `PDDSLMAX` can now each have a minimum of 0 specified.

Record of Revision

Version	Description
9.0	May 1995 Original Printing. Documentation to support the UNICOS 9.0 release running on all Cray computer systems. This manual contains the contents of and supersedes the information formerly provided in the “Configuring UNICOS” section in the *UNICOS System Administration*, 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.
003	July 1999 Revision to support the UNICOS 10.0.0.6 release.
004	February 2000 Revision to support the UNICOS 10.0.0.7 release.
10008	November 2000 Revision to support the UNICOS 10.0.0.8 release.
10010	October 2001 Revision to support the UNICOS 10.0.1.0 release.
10011	May 2002 Revision to support the UNICOS 10.0.1.1 release.

Contents

Page

Page

Page

Tables

Page

Page

Preface

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

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

Warning: Starting with the UNICOS release 10.0, 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 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 security (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 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 runtime 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 man page manuals contain additional information that may be helpful.

- *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 publication is useful for establishing connectivity between the High Performance Parallel Interface (HIPPI) network of a Cray mainframe and any host that has a physical path to any of the network interfaces of the Cray L7R.

- *Cray L7R Release Overview and Software Installation Guide*, contains information on the Cray L7R release and details regarding software installation and configuration for the Cray L7R. This publication contains chapters documenting an overview of the release, purpose and function of the Cray L7R, system and network configuration requirements, software installation and configuration instructions, and troubleshooting.

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.

Ordering Documentation

To order software documentation, contact the Cray Software Distribution Center in any of the following ways:

E-mail:
`orderdsk@cray.com`

Web:
`http://www.cray.com/craydoc/`

Click on the `Cray Publications Order Form` link.

Telephone (inside U.S., Canada):
1–800–284–2729 (BUG CRAY), then 605–9100

Telephone (outside U.S., Canada):
Contact your Cray representative, or call +1–651–605–9100

Fax:
+1–651–605–9001

Mail:
Software Distribution Center
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120–1128
USA

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items, such as file names, pathnames, man page names, command names, and programming language elements.
variable	Italic typeface indicates an element that you will replace with a specific value. For instance, you may replace *filename* with the name `datafile` in your program. It also denotes a word or concept 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 syntax representation for a command, library routine, system call, and so on.
`...`	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, including Cray SV1 series systems.
Cray MPP systems	All configurations of the Cray T3E series. The UNICOS operating system is not supported on Cray T3E systems. Cray T3E systems run the UNICOS/mk operating system.

Reader Comments

Contact us with any comments that will help us to improve the accuracy and usability of this document. Be sure to include the title and number of the document with your comments. We value your comments and will respond to them promptly. Contact us in any of the following ways:

E-mail:
swpubs@cray.com

Telephone (inside U.S., Canada):
1–800–950–2729 (Cray Customer Support Center)

Telephone (outside U.S., Canada):
Contact your Cray representative, or call +1–715–726–4993 (Cray Customer Support Center)

Mail:
Software Publications
Cray Inc.
1340 Mendota Heights Road
Mendota Heights, MN 55120–1128
USA

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 service representative.

The *Cray ML-Safe configuration* refers to a configuration of the UNICOS system that supports processing at multiple security labels and system administration using only non-super user administrative roles. The Cray ML-Safe configuration consists of the subset of UNICOS software that offers these capabilities.

> **Warning:** Starting with the UNICOS 10.0 release, the term *Trusted UNICOS* has been replaced by the term *Cray ML-Safe*. 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.

1.1 Special GigaRing Configuration Rules for the Cray SV1ex System

The following sections describe rules that apply to all Cray SV1ex systems.

1.1.1 Minimum GigaRing Configuration

All Cray SV1ex systems must follow the following minimum configuration rules:

1. Each node must have at least one Cray-supplied disk drive for storing the UNICOS operating system. All new Cray SV1ex systems are shipped with fiber disks. Existing Cray J90 systems that are being upgraded with new processor modules can use existing fiber, IPI, or SCSI disks. Existing Cray SV1 systems that are being upgraded with only new processor modules or are being upgraded with both new processor and new memory modules can use existing fiber, IPI, or SCSI disks.

2. The Processor 0 GigaRing channel is always present and is used to boot the system. Therefore, this ring must include the disk drive(s) that contain UNICOS.

3. Each node requires an Ethernet connection for logging in. This requires one MPN with an ETN-11.

4. The first PC-10 cabinet (joined to the right side of the processor cabinet, as viewed from the front) contains the GigaRing channels from processors 0-3. The GigaRing bulkhead in this PC-10 cabinet should be filled starting with Processor 0. Do not skip processors.

5. Whenever possible fill the first PC-10 cabinet (processors 0-3) in the system first.

6. If present on the system, the second PC-10 cabinet (joined to the left side of the Cray SV1 series cabinet, as viewed from the front) contains the GigaRing channels from processors 4-7.

7. Ring channels must be configured with at least one ION (MPN or SPN).

8. MPNs are not allowed on shared rings that are used to connect multiple mainframe nodes (that is, a ring connecting a Cray T90 system to a Cray SV1ex system). GigaRing channels with multiple mainframe nodes must include at least one SPN (FCN, HPN, IPN, or ESN).

1.1.2 SSD Memory Access

Cray SV1ex systems have a new faster memory subsystem. Depending on system configuration, the Cray SV1ex can have extended memory, a portion of which includes an auxiliary memory known as an internal static storage device (SSD-I) and a high speed block transfer engine (BTE). SSD-I is internal to the Cray SV1ex main memory modules. Main memory occupies the lower 32 Gbytes of the memory subsystem, while SSD memory occupies the upper address space. Depending on the system configuration, each mainframe cabinet can have an SSD that is 0-, 32-, or 96-Gbytes. All memory words are 64 bits wide.

The main purpose of SSD is to temporarily store data sets of a job to speed up access to data sets. It is used essentially as the SSD-T (Cray SSD-T90) and SSD-E (Model E SSD) are used on other Cray PVP systems. Supported uses for SSD-I include fast swap space, file system space, logical device (disk) cache, and secondary data segment (SDS). The BTE provides a CPU-controlled data path for direct memory to memory transfers, for example, between main memory and extended memory or even within main memory.

SSD memory is accessible in two ways: front door and back door. All systems that have an SSD have front door access capability, which allows the SSD to be used as a swap device and/or SSD file system. Only system configured with back door access have full SSD functionality, such as direct disk transfer to and from SSD space and `ldcache`.

Front door access is defined as the movement of data between main memory and the internal SSD (SSD-I). This is accomplished by using `ssread` and `sswrite` system calls for access to SDS space. Control logic transfers words directly between Main memory and SSD. The transfer rate depends on buffer alignment, that is, BTE versus data movement via CPU. SSD-resident file systems and swap I/O also use front door access.

Back door access is defined as the movement of data directly between SSD-I and disk devices. Back door transfers can occur over either a conventional point-to-point GigaRing or a SuperRing consisting of two to four mainframe GigaRing adapters. Back door transfer rates are a function of the number of mainframe nodes, disk controllers, and disk devices, connected to the GigaRing or SuperRing channel. Whereas, the number of rings that can be configured is dependent on the number of processor modules in the system.

1.1.3 Back Door I/O Rules

Back door I/O can be provided on a Cray SV1ex system as follows:

1. The minimum configuration is a point-to-point ring consisting of one mainframe GigaRing connection and one ION, where the ION is an FCN, IPN, HPN, or disk or network MPN (not tape).

2. Alternatively, a SuperRing configuration, consisting of two to four mainframe GigaRing connections and one or more IONs on a single ring, can in some cases provide improved I/O bandwidth. See Section 1.1.4, page 4 for more information.

3. Although a ring that is set up to allow back door I/O can also be used for I/O transfers between disk and main memory, no networking or tape IONs can exist on the same ring. Any attempt to open a tape device on a ring configured with back door capability will fail and generate an error message on the system console.

4. Back door I/O access to ldcache in the SSD is limited to file systems that exist entirely on disk partitions that reside on properly configured back door-capable rings, including both primary and alternate I/O paths.

5. Back door I/O is required to use the SSD for secondary data segment (SDS) space (to support SDS space swapping).

1.1.4 SuperRing Configuration Rules

The following rules apply to all systems that have a SuperRing configuration:

1. A SuperRing is defined to be a single GigaRing consisting of two to four mainframe GigaRing channel adapters (on either processor modules or I/O modules) and one or more IONs on a single ring. Although a SuperRing can be configured to support front door (regular) I/O, a SuperRing is primarily intended to support back door (direct disk to SSD) I/O transfers.

2. There is no maximum supported configuration on a single SuperRing. However, if the total number of mainframe GigaRing adapters and IONs on a ring is greater than 6, performance might be degraded significantly.

3. A SuperRing configuration of three mainframe GigaRing connections with two FCNs provides the best balanced overall capability for back door I/O. The FCNs can move data at ~300 Mbyte/s and the mainframe nodes operate at 200 Mbyte/s.

4. Multiple SuperRings can be configured on a single mainframe; however, a SuperRing can not be a shared ring between multiple mainframe nodes.

5. On systems that use an 8x8 backplane, all the mainframe GigaRing adapters that make up a single SuperRing must be on the same side of the mainframe. (CPU slots 0-3 are on one side, 4-7 are on the other.)

6. On systems that use a 4x4 backplane, only one SuperRing consisting of either two or three mainframe GigaRing adapters is possible unless the system boot ring is configured as a SuperRing.

sn.h File [2]

Note: Typically, this 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:

Bits/chip	CHIPSZ value
65536	M64KCH
262144	M256KCH
1048576	M1MCH
2097152	M2MCH
4194304	M4MCH
8388608	M8MCH
16777216	M16MCH
33554432	M32MCH – (not valid for UNICOS kernel)
67108864	M64MCH

   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.

Option	Description
CRAY_TS	Cray T90
CRAYYMP	Cray SV1 series Cray J90se

   Example:

   ```
   #define MFTYPE     CRAYT90
   ```

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

Option	Description
YMPJ90	Cray Y-MP architecture, including Cray SV1 series, and Cray J90se series
T4XXX	Cray T94
T16XXX	Cray T916
T32XXX	Cray T932

   Example:

   ```
   #define MFSUBTYP   T900XX
   ```

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/c1/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 2 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 ulimit(2)).
DMODE	1	Default offline file retrieval mode for sites running data migration. 1 indicates automatic retrieval; 0 indicates manual-only retrieval.
QUEUE_IO_WITHOUT_NICE_PRIORITY_VALUES	0	Enable/disable the use of nice priorities on the I/O queues that access mirrored disks, striped disks, and non-GigaRing high performance parallel interface (HIPPI) disks. A value of 0 means that a low-priority process may not get access to a disk that is running with the priority-ordered I/O queue. A non-zero value means that even a low-priority process will have access to a disk because the priority order is not being used.
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 MEM. 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. MAXASYN has considerable performance impact on applications utilizing asynchronous I/O (for example, listio and aqio 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.

Parameter	Default value	Description
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 NBUF parameter in the CSL parameter file. There is now a default maximum of 10000 system buffers. This limit can be overridden by explicitly defining NBUF in the sn.h file, config.h file, or startup parameter file, or by altering NBLK_FCTR in config.h.
NCALL	NPROC	Number of callout table entries.
NCLIST	1000	Maximum number of clists. 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 NINODE. 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 NINODE.
NEXECS	4	Maximum number of parallel exec 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	400	Number of file-lock regions.

Parameter	Default value	Description
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 systems without SSD and 2000 for systems with SSD. 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.
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 -M CMEM option of the pcache(8) 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. This also applies to Cray SV1ex systems.
NSU_ASYN	4	Number of asynchronous I/O headers reserved for the system.

Parameter	Default value	Description
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 MEM. 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.
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 pc_acnice 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.

Parameter	Default value	Description
NC1INODE	NINODE	Number of in-core inodes for NC1FS (file system migration parameter). Must be equal to NINODE.
NEXECS	4	Maximum number of parallel exec operations with maximum arguments.
U_MAXPACK	16	Maximum number of outstanding user packets allowed. For Cray SV1 series and Cray J90se systems only.

Table 3 lists the dynamic kernel memory parameters.

Table 3. Dynamic Kernel Memory Parameters

Parameter	Default value	Description
KM_CHM_PADSZ	(((KSTACKSIZE + KM_WPU-1) >> KM_WPU_SHIFTC) * (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 lowmem.c 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 coremap, it must be a multiple of coremap units. That is: (KM_EXPAND_UNITS * KM_WPU) == 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
		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 KM_WPU and KM_EXPAND_UNITS. For example, if KM_EXPAND_UNITS is 512, each expansion requires 8 additional words of bitmap memory to manage the space. If KM_WPU is 16, this will cause the bitmap to be reallocated and moved every two expansions. If KM_WPU is 128, the bitmap will need to be reallocated every 8 expansions. 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 KM_WPU, 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).
KM_UNITS	1408	Initial dynamic kernel memory units from coremap. **Caution:** The value of (KM_UNITS * KM_WPU)should be a multiple of (MEMKLIK*NWPC) to avoid wasting space acquired from coremap. That is, at boot time, a chunk of memory is acquired from coremap to be managed at a finer granularity.
KM_WPU	128	Words per unit. This value must be a power of 2.
KM_WPU_SHIFTC	7	Number of shifts to multiply or divide by the KM_WPU value.

Table 4 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 4. 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 MEMKLIK units, this should be a multiple of MEMKLIK. 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 0 and less than the SHMMAX value.
SHMMNI	20	Maximum number of shared memory identifiers available concurrently in the system (that is, the size of the shmem 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 5 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 5. 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.
MAXUSRORE	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 6 lists the parameters in config.h that are used for interprocess (IPC) semaphore defines and message defines.

Table 6. 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 0 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 msgque table). This value must be greater than 0.
MSGSEG	1024	Number of message segments. The size of the message area is MSGSEG * MSGSSZ bytes. (This will be rounded up to MEMKLIK.) This value must be greater than 0 or equal to the MSGTQL 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 0 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 semop(2) system call by specifying the SEM_UNDO 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 sema table). This value must be greater than 0 and less than 1000.
SEMMNS	100	Number of semaphores in the system (that is, the size of the sem 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 semu table). One entry is used by each process that registers semaphore operations to be undone at process termination. This value must be greater than 0 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 semop(2) 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 SEMMSL 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 SEMMNS value. To conserve table space, set SEMUME to a value less than the SEMMNI 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 0 and less than 2**32.

Table 7 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 7. 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 /dev/console when any user exceeds MAXLOGS login attempts.
DECLASSIFY_DISK	0	Deactivates declassification of the disk; activates when the value is not 0.
DECLASSIFY_PATTERN	0	Declassifies disk write pattern.
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.

<table>
<tr><th>Parameter</th><th>Default value</th><th>Description</th></tr>
<tr><td>MLS_OBJ_RANGES</td><td>NORMAL</td><td>Allow object ranges outside the system label range in mount and setdevs.</td></tr>
<tr><td>NFS_REMOTE_RW_OK</td><td>1</td><td>Enables (nonzero) or disables (zero) NFS-mounting of a remote file system in read-write mode.</td></tr>
<tr><td>NFS_SECURE_EXPORT_OK</td><td>1</td><td>Enables (nonzero) or disables (zero) exporting a secure file system with NFS.</td></tr>
<tr><td>NFS_SECURE_PORTMON</td><td>1</td><td>Enables (nonzero) or disables (zero) whether NFS clients are required to use privileged ports (ports < IPPORT_RESERVED) in order to get NFS services. This, along with NFS_PORTMON (see Table 9, page 28), must be disabled for non-privileged access on MLS systems.</td></tr>
<tr><td>OVERWRITE_COUNT</td><td>3</td><td>Declassifies disk overwrite count.</td></tr>
<tr><td>PASS_MAXSIZE</td><td>8</td><td>Maximum size for machine-generated passwords.</td></tr>
<tr><td>PASS_MINSIZE</td><td>8</td><td>Minimum size for machine-generated passwords.</td></tr>
<tr><td>PRIV_SU</td><td>1</td><td>When set to 1, root (UID 0) has privilege.</td></tr>
<tr><td>RANDOM_PASS_ON</td><td>0</td><td>When set to yes (1), machine-generated passwords are enabled for all users. Setting to no (0) disables this.</td></tr>
<tr><td>SANITIZE_PATTERN</td><td>0</td><td>The pattern used to scrub disks if SECURE_SCRUB is enabled. The default pattern is 0.</td></tr>
<tr><td>SECURE_NET_OPTIONS</td><td colspan="2">(NETW_SOCK_COMPAT | NETW_RCMD_COMPAT)</td></tr>
</table>

Parameter	Default value	Description
		Options for running TCP / IP on a secure system. The value may be any combination of the following: • NETW_STRICT_B1, 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. • NETW_SOCK_COMPAT, in which sockets are automatically made multilevel if the creating process has PRIV_SOCKET. • NETW_RCMD_COMPAT, in which traditional hosts.equiv and .rhosts behavior is allowed. If NETW_RCMD_COMPAT is not used, remote logins with rlogin(1) to root 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 NETW_SOCK_COMPAT and NETW_RCMD_COMPAT. To add the strict B1 evaluation rules, you should add the NETW_STRICT_B1 value; that is, (NETW_SOCK_COMPAT | NETW_RCMD_COMPAT | NETW_STRICT_B1).
SECURE_MAC	0	When set to 1, enforces system high / system low MAC.
SECURE_OPERATOR_CONSOLE	"/dev/console"	Secure console for use by the MLS operator. This parameter is unused.
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.

Parameter	Default value	Description
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.
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.

Parameter	Default value	Description
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 chdir(1) 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_SHUTDWN	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.
SLG_MAXSIZE	8192000	Size of security log to retire. When SLG_FILE is larger than this, it is moved to a file prefixed with SLG_FPREFIX.
SLG_MKDIRV	SLGON	Enables (SLGON) or disables (SLGOFF) logging of all mkdir violations.
SLG_NETWV	SLGON	Enables (SLGON) or disables (SLGOFF) logging of network access violations.

Parameter	Default value	Description
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 rm violations.
SLG_RMDIRV	SLGON	Enables (SLGON) or disables (SLGOFF) logging of all rmdir violations.
SLG_STATE	SLGOFF	Enables (SLGON) or disables (SLGOFF) security log.
SLG_SUID_RQ	SLGON	Enables (SLGON) or disables (SLGOFF) logging of setuid system calls.
SLG_SULOG	SLGON	Enables (SLGON) or disables (SLGOFF) logging of all su(1) 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	"/dev/console"	Default console for MLS administration. The value of this parameter must be /dev/console.
SYSVCOMPS	0	Bit map that defines valid system compartments. By default, system compartments are turned off.

Table 8 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 8. 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 *UNICOS Networking Facilities Administrator's Guide*. This pool of buffers is allocated at boot time in module `/usr/src/uts/tcp/kern/uipc_mbuf` and cannot be increased while the system is running. The ICMS places this value in both the `config.h` file and the CSL parameter file. The value in the CSL parameter file overrides the value in `config.h`.

Table 9 shows the NFS kernel parameters in `config.h`.

Table 9. 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 *rnodes*) at any one time in the kernel. Each active NFS file or directory requires an rnode. Files may have multiple requests outstanding at any one time. If the number of active NFS files exceeds the NFS_NUM_RNODES value, the rnodes are shared among active files.
NFS_PORTMON	0	If set non-zero, then NFS clients are required to use privileged ports (ports < IPPORT_RESERVED) in order to get NFS services.
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 10. 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 10. 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_portmon	NFS_PORTMON

CSL tag	config.h
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 the following:

- General system parameters
- Kernel generation parameters
- Kernel subsystem parameters,
- Programming environment parameters

Default values are either literals or 1 (which enables a subsystem) or (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 11. 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.
CONFIG_NODE	*node_name*	Node name of the system, by which it is known to a communications network. See uname(1).
CONFIG_SN	sn*number*	Serial number of the system to generate.
CONFIG_SYS	*system_name*	Name of the system. This information is used by the uname(1) command. Typically, *system_name* is the same as the serial number.
CONFIG_TARGET	(null)	Target of the system to generate. If null, the existing default is used. See target(1).
CONFIG_TMPDIR	/tmp	Name of the desired temporary-file directory. If the TMPDIR environment variable is already defined, this value is not used. See tmpnam(3).
CONFIG_VERSION	*version*	Version name of the system. If unspecified, it defaults to the value of the environment variable LOGNAME. See uname(1).

Table 12. 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 13. 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 14. 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.
CONFIG_GENBIN	/usr/gen/bin	Full path of the directory that contains the generation software.

Parameter	Default value	Description
CONFIG_GENCMDS	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 nmake(1).
CONFIG_GENPROD_RULES	0	Repeatable relocatables capability.
CONFIG_MPP_CPP	CONFIG_GENBIN/../reqs/cc/mppcpp	Sets the nmake(1) CPP variable to the generation mppcpp for the cray-t3e target.
CONFIG_PACKAGE	0	Certain nmake(1) 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 PATH is set to the value of CONFIG_PATH.
CONFIG_RLS_MAJOR	*integer*	UNICOS major release number.
CONFIG_RLS_MINOR	*integer*	UNICOS minor release number.
CONFIG_RLS_REVISION	*integer*	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 umask(1) 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	*target*	Desired set of cross-targeted libraries, such as cray-c90 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	*target*	Desired set of cross-targeted libraries for a mixed mode system, such as cray-ts,ieee for use on a Cray floating-point Cray T90 mainframe.

Table 15. Kernel Subsystem Parameters

Parameter	Default value	Description
CONFIG_BBG	0	Bus Based Gateway (BBG).
CONFIG_BMM	0	Bit matrix multiply functional unit (uts 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 series 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 (uts kernel).
CONFIG_IPI3	1	IPI-3 / IPI capability in the kernel.
CONFIG_KERBEROS	0	Kerberos support.
CONFIG_MPP	1	CRAY MPP system is attached.
CONFIG_NETMON	1	Network monitor.
CONFIG_NETTOLS	0	Network testing tools.
CONFIG_NFS	1	Network File System (NFS).

Parameter	Default value	Description
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 (OWS). 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 16. 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]

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

> **Note:** It is strongly recommended that you use the installation 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*

When the configuration is activated, a copy of the CSL parameter file is stored in `/etc/config/param`. For all Cray 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:
- *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:

Section	Description
`dumpinfo`	Cray IOS Model E based system dump parameters
`gigaring`	GigaRing configuration parameters
`ios_e`	Cray IOS Model E based system and Cray SV1 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 Cray 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 operator workstation (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 ;
```

type Either `memory` or `SSD`, indicating central memory or SSD memory to be dumped.

start Integer indicating the beginning location of a memory section to be dumped.

stop Integer indicating the ending location of a memory section to be dumped.

units 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 Cray Model E based systems or Cray Model V based systems.

All Cray SV1ex systems must follow some minimum configuration rules; see Section 1.1, page 2.

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.

This applies to the Cray SSD-T90 and also to the Cray SSD-I on Cray SV1ex systems.

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 J90se and Cray SV1 Series Systems*.

5.3.3.2 HISP Declarations

Note: This section does **not** apply to Cray 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 `C90`, which means 200 Mbyte control / 200 Mbyte data.

5.3.3.3 IOP Boot Declarations

Note: This section does **not** apply to Cray 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 {
                miop; 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 C90;

        boot miop with "/home/ows7001/cri/os/ios/iopmux";
        boot eiop 0 with "/home/ows7001/cri/os/ios/eiop.comm";
        boot eiop 1 with "/home/ows7001/cri/os/ios/eiop.bmx";
        boot eiop 2 with "/home/ows7001/cri/os/ios/eiop.dca2";
        boot eiop 3 with "/home/ows7001/cri/os/ios/eiop.dca1";
    }
    cluster 1 {
        miop; 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 C90;

        boot miop with "/home/ows7001/cri/os/ios/iopmux";
        boot eiop 0 with "/home/ows7001/cri/os/ios/eiop.dca1";
        boot eiop 1 with "/home/ows7001/cri/os/ios/eiop.dca1";
        boot eiop 2 with "/home/ows7001/cri/os/ios/eiop.dca2";
        boot eiop 3 with "/home/ows7001/cri/os/ios/eiop.hippi";

    }
}
```

5.3.4 `mainframe` Section

The `mainframe` section defines the following hardware parameters:

- I/O modules
- Number of CPUs (less those used as I/O modules)
- Number of multistreaming processors (MSPs)
- Number of mainframe cluster registers
- Size of the mainframe memory
- Size of the SSD-I memory
- Channel information
  - Physical channel for a GigaRing environment
  - Low-speed channel information (channel and target) for Model E based systems

– Very high speed (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 (less those used as I/O modules)
- Number of mainframe cluster registers
- Size of memory
- Down CPUs
- CPU Cache

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;
```

units may be either `words` or `Gwords`. 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.

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

5.3.4.1.4 Size of Extended Memory

The Cray SV1ex mainframe extended memory size is specified by the following statement:

```
value units xmemory;
```

units can be `words`, `Mwords`, or `Gwords`. 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.

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

5.3.4.1.5 Down CPUs

The CPUs to be placed in a down condition at deadstart time are specified by the following statements:

```
down cpu cpuname;
down cpus cpuname1 cpuname2 ...;
```

cpunameX is the logical number of the CPU.

5.3.4.1.6 CPU Cache

The ability to disable CPU cache, either system-wide or for the specified CPUs, is specified by the following statements:

```
cacheoff user;
cacheoff kernel;
cacheoff user cpu cpuname;
cacheoff user cpus cpuname1 cpuname2 ...;
cacheoff kernel cpu cpuname;
cacheoff kernel cpus cpuname1 cpuname2 ...;
```

The `cacheoff user;` and `cacheoff kernel;` directives disable user cache or kernel cache on a system-wide basis. The remaining directives disable user cache or kernel cache for a specific CPU or set of CPUs. *cpunameX* is the logical number of the CPU on which cache will be disabled.

For example, the following statement will disable kernel cache in all CPUs and will disable user cache in CPUs 1, 3 and 21:

```
cacheoff kernel;
cacheoff user cpus 1 3 21;
```

5.3.4.2 Parameters for Cray SV1 series GigaRing Based System

- Multistreaming Processors (MSPs)
- I/O Modules

5.3.4.2.1 Multistreaming Processors (MSPs)

The maximum number of multistreaming processors (MSPs) is specified by the following statement:

```
msp_number msps;
```

The variable *msp_number* specifies the maximum number of MSPs in the machine. *msp_number* 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 17.

Table 17. CPU / MSP Correlation

Number of CPUs	Maximum Number of MSPs
4	0
8	1
12	2
16	3
20	4
24	5
28	6
32	6

5.3.4.2.2 I/O modules

An I / O module is a CPU module that will be used primarily to provide increased I / O bandwidth and / or I / O capacity on a SV1 series system. An I / O module will have a GigaRing channel connection. I / O modules perform operating system services, such as I / O interrupt handling, clock interrupt handling, system process handling (such as network and utility). I / O modules are not available for user job assignment.

SV1 I / O modules are specified by the following statement:

```
iomodules module1 module2 ...;
```

modulenameX is the logical number of the CPU.

The variables *modulename1* and *modulename2* specify the CPU modules to use as I / O modules. For example, if the value in *modulename1* is 5 and *modulename2* is 6, the statement

```
iomodules 5 6;
```

configures CPU modules 5 and 6 (using 0-based module numbering) as I / O modules to perform operating system services. In this configuration, CPUs 16 - 23 would be used to perform operating system services.

5.3.4.3 Parameters for Cray SV1 series VME Systems Only

The channels to be used for memory high performance parallel interface (HIPPI) are specified by the following statement:

```
channel channel_number is memhippi to mainframe;
```

channel_number specifies memory HIPPI input channel.

For example:

```
channel 040 is memhippi to mainframe;
```

This parameter is required for all Cray SV1 series and VME systems.

5.3.4.4 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 J90se GigaRing based systems and Cray SV1 series 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 18 shows the channels for I/O module 0.

Table 18. 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 19 shows the channel number ranges for a Cray T932 configuration of four IO2 modules.

Table 19. 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

5.3.4.5 Parameters for Model E Based Systems Only

The Channel declarations are for Cray Model E based systems only. Channel declarations are either for a low-speed channel pair or a very high speed (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 no longer must specify `pseudo TCP` for TCP/IP to be operational. If `pseudo TCP` is included in a parameter file, the setting will be ignored by the UNICOS kernel during the booting of the mainframe.

5.3.4.6 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 msps;
        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;
}
```

5.3.4.7 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 vhisp  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 20 through Table 22 show these parameters.

Table 20. 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 21. 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 22. Maximum Limits Parameters

Parameter	Description

FAULT_RESPONSE	Control the fault tolerance feature. The *value* is a bit mask that controls the various options in the feature. Each bit enables or disables an option, as follows:
	Bit — Option Controlled (1=on, 0=off)
	01 — Fault tolerance feature.
	02 — Panic when detecting a lock held by a down CPU.
	04 — Down CPU cache on a user cache parity error.
	The following example disables the feature:
	`0     FAULT_RESPONSE`
	The default for *value* is 07 (all options selected).
	The following example enables the basic feature and downing of CPU cache on a user cache parity error, but does not panic when detecting a lock held by a down CPU:
	`05    FAULT_RESPONSE`

Parameter	Description
GUESTMAX	Maximum number of guest systems.
NBUF	Number of buffer headers. There is now a default maximum of 10000 system buffers. This limit can be overridden by explicitly defining NBUF in the sn.h file, config.h file, or startup parameter file, or by altering NBLK_FCTR in config.h.
NGRT	Maximum number of guest resource table entries.

GigaRing based systems and Model E based systems have common and unique disk parameters. Table 23 through Table 26 show these parameters.

Table 23. 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); minimum is 0.
PDDSLMAX	Maximum number of PDD slices; minimum is 0. 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 24. 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 25. Disk and Block Transfer Engine Parameters (Cray SV1ex Systems with SSD-I Only)

Parameter	Description
SSDIMAX	Maximum number of SSD-I devices. The value must be either 0 or 1. 0 is the default.
SSDISLMAX	Maximum number of slices that can be allocated to the SSD-I device. This value must be an integer in the range 0 through 2^{15} –1, depending upon the value set for SSDIMAX. The default is 32. This value also limits the minor number for this type. The maximum minor number is SSDISLMAX –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	0–14
	Logical unit number 0–7	
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 Additional Parameters for a Cray SV1ex System with SSD-I

The following is an example of additional parameters required for a Cray SV1ex system with SSD-I:

```
unicos {
        1         SSDIMAX;
        16        SSDISLMAX;
}
```

5.3.5.4 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, SSD-T (Cray SSD-T90) for GigaRing based systems and SSD-I (Cray SV1ex auxiliary memory) for Cray SV1ex systems `filesystem` section
  - disks, RAM, and SSD for Model E based systems (SSD-E)
- 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)
- Description of the SSD-I (Cray SV1ex 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. Supported disk types include: IPI-2, SCSI, Fibre Channel, HIPPI, Ethernet (10Base-T and 100Base-T), FDDI, ATM DC3, Network disk, and Special (RAM, SSD, SSD-T, SSD-I).

5.3.6.1.1 Slice Definitions for Physical Storage Devices

Warning: Due to the fact that the UNICOS kernel and `econfig` do not know the size of xdd devices, checks for a slice spilling over the end of the drive and for gaps after the last slice do not work for xdd (nor for hdd 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.

Note: `iounit` is not valid for SSD-I.

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` or `ssdi` 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 28 shows the preferred units 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 28. 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`

Physical storage device type	Preferred units	Optional units	Maximum minor number –1
SSD	blocks	sectors	SSDDSLMAX
SSD-T90	blocks	sectors	SSDTSLMAX
SSD-I	blocks	sectors	SSDISLMAX

5.3.6.1.2 Physical Device Definition for GigaRing Based Systems

Warning: Due to the fact that the UNICOS kernel and `econfig` do not know the size of xdd devices, checks for a slice spilling over the end of the drive and for gaps after the last slice do not work for xdd (nor for hdd devices).

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)
- Auxiliary memory (SSD-I) for a Cray SV1ex system

Table 29 summarizes disk information for GigaRing based systems.

Table 29. 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;
           }
     }
```

device_name	Name of the physical storage device, which must be unique among all devices.
type	Type of the physical storage device.

ring_number	Number of the I/O path ring.
node_number	Number of the I/O path node.
channel_number	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.
disk_unit_number	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.
pdd_slice_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_sector_number	Starting sector number of the slice.
length_in_sectors	Length of the slice in sectors.

The **SSD-T (`ssdt`) definition** applies only to GigaRing based systems. It has the following format:

```
ssdt ssdt_name {
          iounit iounit_value;
          tmtype tmtype_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;
          }
                [piopaths piopaths_bitmask];
          xdd xdd_name {
                minor minor_number;
                block starting_block_number;
                length length_in_blocks blocks;
          }
  }
```

ssdt_name — Name of the Cray SSD-T90, which must be unique among all devices.

iounit_value — Block size in 512–byte units.

tmtype_number — 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.

lunit_number — (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 `lunit` defaults to 0.

ring_number — GigaRing ring number for the Cray SSD-T90 device. (This can be a logical ring / node address in the form of a `gr_union` device; see Section 5.3.2.2, page 43.)

node_number	GigaRing node number for the Cray SSD-T90 device. (This can be a logical ring / node address in the form of a `gr_union` device; see Section 5.3.2.2, page 43.)
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 **SSD-I (`ssdi`) definition** applies only to Cray SV1ex, which are all GigaRing based systems. It has the following format:

```
ssdi ssdi_name {
          tmtype tmtype_number;
          iopath {
                ring ring_number;
                node node_number;
          }
          xdd xdd_name {
                minor minor_number;
                block starting_block_number;
                length length_in_blocks blocks;
          }
          xdd xdd_name {
                minor minor_number;
                block starting_block_number;
                length length_in_blocks blocks;
          }
  }
```

ssdi_name	Name of the SSD-I, which must be unique among all devices.
tmtype_number	Target memory type, which determines the memory address and configuration characteristics for SSD-I. This value must be 3 for SSD-I. There is no default.
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;
            }
     }
```

xdisk_name	Name of the physical storage device, which must be unique among all devices.
iounit_value	Sector size in 4096–byte I/O units.
ring_number	GigaRing ring number of the peripheral channel adapter (PCA).
node_number	GigaRing ring node number of the PCA.
channel_number	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.
disk_unit_number	Device unit number.
xdd_slice_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_sector_number	Starting sector number of the slice.
length_in_sectors	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
      }
```

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 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;
           }
     }
```

device_name — Name of the physical storage device, which must be unique among all devices.

type — Type of the physical storage device.

cluster_number — Number of the I/O cluster.

> **Note:** For Cray Model V based systems, `cluster` specifies the VME IOS, and `eiop` specifies the controller within the VME IOS. For more information, see the *Cray Scalable I/O Messages*.

eiop_number — Number of the EIOP I/O processor.

channel_number — Number of the I/O path channel. You must include a leading 0 to specify the channel number in octal form.

disk_unit_number — 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.

pdd_slice_name — Name of the slice.

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

starting_sector_number — Starting sector number of the slice.

length_in_sectors — 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;
            }
      }
```

ssd_name — Name of the SSD, which must be unique among all devices.

length_number — Size of the SSD.

units	One of the following: `blocks`, `Mwords`, or `sectors`.
pdd_slice_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.

5.3.6.2 Device Node Definition

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

Device type	Description
`pdd`	Physical device for use with the CSL type `disk`.
`ldd`	Logical device.
`sdd`	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 62, for the format.

The `tmtype` 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 30 shows the `tmtype` values.

Table 30. Target Memory Type Values

`tmtype` value	Description
8	An 8-processor system
9	A 16-processor system

5.3.6.4 Cray SV1ex Auxiliary Memory (SSD-I) Description

The Cray SV1ex system has a faster, extended memory, a portion of which includes an auxiliary memory called SSD-I, and a high speed block transfer engine (BTE). SSD-I is internal to the Cray SV1ex memory modules. It is used essentially as the SSD-T (Cray SSD-T90) and SSD-E (Model E SSD) are used on other Cray PVP systems. Supported uses for SSD-I include fast swap space, file system space, logical device (disk) cache, and secondary data segment (SDS). The BTE provides a CPU-controlled data path for direct memory to memory transfers, for example, between main memory and extended memory or even within main memory.

The maximum number of SSD-I devices is designated by the `SSDIMAX` parameter in the `unicos` section of the parameter file. See Section 5.3.6.1, page 62, for the format.

The `tmtype` parameter is the target memory type associated with SSD-I memory. The target memory type determines memory address and configuration characteristics for the SSD-I. It must be set to a value of 3 for SSD-I.

5.3.6.5 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, Cray SSD-T90 slice, or SSD-I 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
```

5.3.6.6 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 55.

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

5.3.6.7 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 55.

```
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;
}
```

5.3.6.8 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 55.

```
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;
}
```

5.3.6.9 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 55.

```
 filesystem {
```

```
        ssdt SSDT00 { iounit 1; tmtype 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;
}
```

5.3.6.10 Example of the `filesystem` Section for a SSD-I Device

The following example shows the `filesystem` section of a working CSL parameter file for an SSD-I device.

Note: The `length` can be 4 Gwords or 8 Gwords. Additional CSL tags are required in the `unicos` section. See Section 5.3.5, page 55.

```
 filesystem {
        /*
         * SSD-I configuration.
         */
        ssdi ssd-sv1ex {
                tmtype 3;
                length 4 Gwords;
                xdd ssdi_test {
                        minor 1;
                        block 0;
                        length 97152 blocks;
                }
                xdd ssdi_swap {
                        minor 2;
                        block 97152;
                        length 1000000 blocks;
                }
                xdd ssdi_sds {
                        minor 3;
                        block 1097152;
                        length 1000000 blocks;
                }
        }
```

5.3.6.11 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 d1234.0  {type DD62; iopath{cluster 1; eiop 2; channel 034;} unit 0;
       pdd stripe0      {minor 70; sector        0; length  667296 sectors;}
  }
   disk d1236.0  {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;
}
```

5.3.6.12 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 55.

```
   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:

```
number network_parameter_statement;
```

The *number* is a valid CSL number for the network statement. *network_parameter_statement* argument can have the values described in Table 31 through Table 35, page 86.

Table 31. 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.
cnfs_static_clients	Maximum number of active Cray NFS static clients.
cnfs_temp_clients	Maximum number of active Cray NFS temporary clients.
harp_entries	Size of the High Performance Parallel Interface (HIPPI) address resolution protocol (ARP) table.
harp_recv	Amount of socket space used for receive for HIPPI ARP traffic. Must be a power of 2.
harp_send	Amount of socket space used for send for HIPPI ARP traffic. Must be a power of 2.
hidirmode	Sets permissions or file mode for the following directories: /dev/ghippi# (GigaRing based system) and /dev/hippi (Model E based system).
hifilemode	Sets permissions or file mode for the following files: /dev/ghippi#/* (GigaRing based system) and /dev/hippi/* (Model E based system).
nfs_duptimeout	Time interval in seconds during which duplicate requests received by the NFS server will be dropped.
nfs_maxdata	Maximum amount of data that can be transferred in an NFS request (the NFS data buffer size).
nfs_maxdupreqs	Size of the NFS server's duplicate request cache.
nfs_num_rnodes	Size of the NFS client's NFS file-system-dependent node table (rnode table for NFS).

Parameters	Description
nfs_portmon	Enables (nonzero) or disables (zero) whether NFS clients are required to use privileged ports (ports < IPPORT_RESERVED) in order to get NFS services.
nfs_printinter	Time in seconds between server not responding message to a down server.
nfs_static_clients	Number of static client handles reserved for NFS client activity.
nfs_temp_clients	Number of dynamically allocated client handles that can be used for NFS client activity when all of the static client handles are in use.
nfs_wcredmax	Maximum number of credential structures.
nfs3_async_max	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).
nfs3_async_time	The amount of time (in seconds) that data will be held in the NFS async write cache on the client.
tcp_numbspace	Number of clicks of memory set aside for TCP/IP managed memory buffers (MBUFs).

Table 32. Network Parameter Values (GigaRing Based Systems)

Parameters	Description
ithreshold	(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 gr interfaces.
othreshold	(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 gr interfaces.

Parameters	Description
maxinputs	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.
maxoutputs	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.
maxusers	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 33. 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 34. Network Parameter Values (Model V Based Systems)

Parameters	Description
atmmaxdevs	Maximum number of ATM devices.
enmaxdevs	Maximum number of Ethernet devices.

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

Parameters	Description
fddirmode	Sets permissions of file mode for /dev/fddi* directories.
fdfilemode	Sets permission of file mode for /dev/fddi*/* files.
fdmaxpaths	Maximum number of logical paths per physical FDDI channel.
npdirmode	Sets the permissions for subdirectories of /dev/comm.
npfilemode	Sets the permissions for /dev/comm/*CHAN*/lp*XX* files.
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;
       }
```

interface_type — Type of the interface as defined in Table 36, page 89.

device_type — Type of the device as defined in Section 5.3.7.3.

channel_mode — One of the following values:

`12MB`

`12MB_LP`

`6MB`

driver_type — One of the following values:

`FY`

`LCP`

`MP`

`NSC_MP`

`PB`

`RAW`

driver_mode_number — Number of the driver mode.

function_number — Number of the function.

direction — Either `input` or `output` (only used with HIPPI devices).

timeout_number — Time-out value.

5.3.7.3 `device type` for Model E and Model V 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:

Device type	Description
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 provides you with configuration templates for a number of standard interfaces, as shown in Table 36.

Table 36. 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
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 37. 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 38. 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 *number* argument is the ordinal of the network device.

Table 39. 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:

Identifier	Description
`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 `input` 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 each packet. The

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

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_nmbspace;
 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 atmmaxdevs;
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 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.

Runtime 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/ghippi#.arp` (GigaRing based systems only)
- `/etc/gr#.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 112) 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:

daemongroup tag groupstart kill pathname arguments

daemongroup — The group to which this daemon belongs.

tag — 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.

groupstart — 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 *groupstart* of NO may be started individually.

kill — A value indicating how the daemon process may be terminated.

pathname — The path name of the daemon executable.

arguments — 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 `mfsck`(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 `mfsck`(8) command was formerly called `gencat`(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:

filesystem directory type options frequency passnumber

The first and last fields are used by `mfsck` (see the `mfsck`(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.

Type	File system
NC1FS	UNICOS file system
NFS	Network file system
PROC	/proc 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:

label — String against which getty tries to match its second argument.

initial_flags — Initial ioctl(2) settings to which the terminal is to be set if a terminal type is not specified with getty.

final_flags — Flags that are set just before getty executes login(1).

login_prompt — The login prompt. All characters and white space in the field are included in the prompt.

next_label — The desired speed, specified by a break character. If this field does not contain the speed, getty searches for the entry with *next_label* as its *label* 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 ghippi#.arp File

The /etc/ghippi#.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 gr#.arp File

The /etc/gr#.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:

id`:`*rstate*`:`*action*`:`*process*

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:

Field	Description
id	1 to 4 characters that uniquely identify an entry.
rstate	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:

Command	Description
`/etc/netstart.pre`	The script to be run before `netstart` to accommodate local requirements.
`/etc/nwmstart`	Command used to initialize the underlying network media
`/etc/tcpstart`	Command to initialize TCP/IP software.
`/etc/nfsstart`	Command to initialize the network file system (NFS) software.

/etc/ypstart	Command to initialize network information service (NIS) software. (NIS was formerly known as yellow pages.)
/etc/netstart.pst	The script to be run after netstart 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 (listed from left to right, top to bottom) have special meaning in the configuration specification language (CSL):

12MB	12MB_LP	6MB
A130	A400	ALL
CNT	Cray	D90
DA301	DA302	DA60
DA62	DD10	DD11
DD19	DD29	DD3
DD301	DD302	DD304
DD308	DD309	DD314
DD318	DD39	DD4
DD40	DD41	DD42
DD49	DD50	DD501
DD5I	DD5S	DD60
DD61	DD62	DD6S
DDAS2	DDESDI	DDIMEM
DDLDAS	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	NTTYDEV
NYPEDEV	OLNET	PB
PBUFSIZE	PDDMAX	PDDSLMAX
PLCHCORE	PS_32	P_8
RAM	RAW	RD-1
RDDSLMAX	REGT	SBUFSIZE
SDDSLMAX	SMT	SSD
SSDDSLMAX	SSDTMAX	SSDTSLMAX
SSDIMAX	SSDISLMAX	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	atmmaxdevs	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	down	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	iomodule
iomodules	iopath	ios_e
iounit	is	ithreshold
kernel	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	nocache	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	ssdi	ssdt
start	stream	subsection
swapdev	system	tcp_nmbspace
timeout	tmtype	to
track	tracks	treq
type	unicos	unit
upper	user	user_channel
vhisp	with	word
words	write	xdd
xdisk	xmemory	

Index

D

F

G

H

K

L

M

O

W

X