UNICOS® Basic Administration Guide for CRAY J90se™ GigaRing based Systems

SG–2210 10.0

Portions of this product may still be in development. The existence of those portions still in development is not a commitment of actual release or support by Cray Research, Inc. Cray Research, Inc. assumes no liability for any damages resulting from attempts to use any functionality or documentation not officially released and supported. If it is released, the final form and the time of official release and start of support is at the discretion of Cray Research, Inc.

Record of Revision

Version	*Description*
9.2	December 1996 Original printing. This guide is provided to support the 9.2 release of UNICOS on the CRAY J90se system.
9.3	August 1997 Supports basic administration for the 9.3 release of UNICOS on the CRAY J90se system.
10.0	November 1997 Supports basic administration for the 10.0 release of UNICOS on the CRAY J90se system.

Contents

Page

Page

Page

Page

Page

Page

Page

Page

Page

Page

Page

Page

Page

Figures

Tables

Page

Preface

This guide is written for system administrators of CRAY J90se systems running UNICOS 10.0. It includes information required for basic system administration.

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

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

UNICOS system administration publications

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

- *General UNICOS System Administration*, Cray Research publication SG–2301, contains information on performing basic administration tasks as well as information about system and security administration using the UNICOS multilevel (MLS) feature. This publication contains chapters documenting file system planning, UNICOS startup and shutdown procedures, file system maintenance, basic administration tools, crash and dump analysis, the UNICOS multilevel security (MLS) feature, and administration of online features.
- *UNICOS Resource Administration*, Cray Research publication SG–2302, 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*, Cray Research publication SG–2303, provides information about the UNICOS kernel configuration files and the run-time configuration files and scripts.
- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, 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*, Cray Research publication SG–2150, describes how to configure, monitor, and control the Cray Network Queuing Environment (NQE) running on a UNIX system.
- *Kerberos Administrator's Guide*, Cray Research publication SG–2306, 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*, Cray Research publication SG–2307, 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.

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

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011
- *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022
- *UNICOS System Libraries Reference Manual*, Cray Research publication SR–2080

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

- *UNICOS User Commands Ready Reference*, Cray Research publication SQ–2056
- *UNICOS System Libraries Ready Reference*, Cray Research publication SQ–2147
- *UNICOS System Calls Ready Reference*, Cray Research publication SQ–2215
- *UNICOS Administrator Commands Ready Reference*, Cray Research publication SQ–2413

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

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

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

Ordering Cray Research publications

The *User Publications Catalog*, Cray Research publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907, or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

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

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

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers:

1	User commands
1B	User commands ported from BSD
2	System calls
3	Library routines, macros, and opdefs
4	Devices (special files)
4P	Protocols
5	File formats
7	Miscellaneous topics

	7D DWB-related information 8 Administrator commands Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

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

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems.
Cray MPP systems	All configurations of the CRAY T3D series. The UNICOS operating system is not supported on CRAY T3E systems. CRAY T3E systems run the UNICOS/mk operating system.
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release.

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

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

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

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

Reader comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail at the following address:

  `publications@cray.com`

- Contact your customer service representative and ask that an SPR or PV be filed. If filing an SPR, use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

We value your comments and will respond to them promptly.

Introduction [1]

This chapter discusses your role as a UNICOS system administrator, lists the log books you need to administer the system smoothly, and describes the major features of UNICOS. It also provides a brief overview of the contents of this document with cross references to related information and documentation.

1.1 The UNICOS system administrator

The UNICOS system administrator maintains the Cray Research UNICOS computing environment for its users. He or she is responsible for the following:

- Getting the system up and running and available for job submissions.
- Making site-specific configuration changes.
- Resolving hardware and software problems.
- Performing administrative duties necessary to maintain a system for all users.

Administrative duties may consist of the following functions:

- Section 10.1, page 195, Configuring and maintaining system accounting
- Chapter 6, page 105, Backing up and restoring file systems (dumps and restores)
- Chapter 7, page 129, Adding and deleting users
- Chapter 5, page 35, Maintaining file systems and structures
- Chapter 9, page 179, Tracking, analyzing, and resolving problems
- Chapter 11, page 269, Configuring and administering the network
- , Tuning the system and monitoring performance
- Upgrading and modifying the system

1.2 Create and maintain a log book

You must create and maintain a system administration log book that includes the following information:

- An incident report log to record problems, how they occur, and how they are resolved.
- Logs of scripts used to perform backups, the location(s) of backup tapes, and all pertinent backup-related information.
- A system crash log with procedures for crash recovery.
- Local documentation, detailing site-specific procedures, such as operator procedures, backup procedures, and so on.
- Path names of essential scripts or files (especially the current configuration and parameter files).
- Emergency names and phone numbers, as well as any emergency procedures relevant for your site.

Note: You should keep your log book current. It is invaluable for troubleshooting purposes.

1.3 Major features of UNICOS

The Cray UNICOS operation system is based on the UNIX System V operating system with Berkeley extensions. It is an interactive and batch operating system that offers many features for performance, functionality, application portability, and I/O connectivity.

UNICOS combines all of the strengths inherent in UNIX, such as its familiar user interface, with production-oriented features. These including high-performance I/O, multiprocessing support, ANSI/IBM tape support, resource allocation and control, and enhanced process scheduling.

The following subsections describe the major UNICOS characteristics.

1.3.1 Cray Scalable I/O (SIO)

The Cray Scalable I/O (SIO) architecture consists of an array of I/O nodes (IONs) connected by a new high-speed channel called the *GigaRing channel*. The GigaRing channel is a scalable input/output (I/O) and networking channel that supports upcoming I/O peripherals for the CRAY J90se system.

1.3.2 File systems

UNICOS modifies the regular UNIX System V file system with an improved disk block allocation scheme, and the ability to create file systems that can span multiple physical disk devices.

1.3.3 Disk devices

UNICOS permits disk striping and banding techniques to improve file system performance and reliability. A unique language, the *configuration specification language* (CSL), defines the physical and logical characteristics of your UNICOS disk devices.

1.3.4 File system quotas

File system quotas have been implemented under UNICOS to control the amount of disk space that you use for files. Two different types of quotas, file and inode, are already supported. Additionally, you may set quotas for three ID classes, user, group, and/or account IDs.

1.3.5 User database (UDB)

UNICOS uses a data file, called the *user database* (`/etc/udb`) that holds comprehensive resource allocation and control information about users. The UNIX equivalent of this data file, the `/etc/passwd` file, is maintained for compatibility.

1.3.6 Resource control

UNICOS resource control allows you to set limits on CPU, memory, tapes, and file allocation. Limits are applied to processes or jobs to establish the maximum amount system resources that they can use. You can specify per-process and / or per-job limits for interactive and batch workloads. This lets a system provide restricted resources for interactive use without limiting a user's batch resources to the same degree.

1.3.7 Unified Resource Manager (URM)

The Unified Resource Manager (URM) is a job scheduler that balances the demands of batch and interactive sessions. URM provides high-level allocation

control of system resources to run jobs that originated either in batch mode or in an interactive session.

1.3.8 Fair-share scheduler

The fair-share scheduler is a process scheduler that works with the standard System V scheduler to equitably distribute system CPU resources. The fair-share scheduler regularly adjusts the scheduling priorities of all running processes based on users' usage history and their "share" of available CPU resource.

1.3.9 System accounting

UNICOS supports two methods of system accounting: standard System V system accounting, and Cray system accounting (CSA). CSA meets the unique accounting requirements of Cray Research customers. Like the standard System V accounting package, CSA provides a method to collect per-process resource usage data, record connect sessions, monitor disk usage, and charge fees to users. In addition, it permits sites to perform per-job and device accounting, along with daemon accounting. Individual sites can select either accounting system by launching the appropriate shell scripts and programs.

1.3.10 TCP/IP

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite provides network communications that use the TCP/IP family of protocols and applications. It allows Cray Research systems to become a peer node of any established TCP/IP network and permits other users and networks to access the UNICOS environment.

1.3.11 Menu system

UNICOS contains a set of shell scripts, parameter files, and a user interface written in menu specification language (MSL). You may use this menu system to perform system configuration changes. For more information, see Appendix A, page 309.

1.3.12 Data migration

The optional UNICOS Data Migration Facility (DMF) provides on-line file system space by moving selected files off-line to a designated storage device(s). These files remain cataloged in their original directories and behave as if they

are still disk resident. Likewise, an on-line disk can be considered a cached copy of a larger virtual disk space.

UNICOS DMF is not part of the standard UNICOS software package. It is available as an optional software package.

1.3.13 System activity monitor (SAM)

The Cray Research system activity monitor, `sam`, collects and displays system activity data from selected Cray Research computer systems. It consists of a data acquisition daemon, `samdaemon`, and two display clients, `xsam` and `csam`.

For more information see *UNICOS Resource Administration*, Cray Research publication SG–2302.

1.3.14 Network Queuing Environment (NQE)

The optional Cray Research Network Queuing Environment (NQE) feature for the UNICOS consists of the Network Queuing EXtensions (NQX), Network Queuing System (NQS), and the File Transfer Agent (FTA). NQE is a software product that consists of a set of servers and clients that allows batch requests to be executed across a load-balanced network of hosts know as a batch complex. The NQX component of NQE provides a Network Load Balancer (NLB) that supports destination selection and load balancing. The NQS component of NQE lets users submit, monitor, and control batch jobs for execution on a local or remote system running the UNICOS system. The FTA component of NQE queues synchronous or asynchronous outbound and inbound file transfers over the network.

For more information, see *UNICOS NQS and NQE Administrator's Guide*, Cray Research publication SG–2305.

1.4 How this guide will help you

Note: Before you consult any of the procedures in this document, you must first boot your SIO and UNICOS software in multiuser mode. See the *UNICOS Installation Guide for CRAY J90se GigaRing based Systems*, Cray Research publication SG–5296, and the *Open Me First* document for more information.

This guide provides you with information to perform the following tasks:

- Establish and maintain basic system security; see Chapter 3, page 17.

- Start up and shut down the SIO and UNICOS; see Chapter 4, page 25.
- Verify and change date and time of UNICOS; see Chapter 4, page 25.
- Determine existing file systems; see Chapter 5, page 35.
- Plan and configure file systems; see Chapter 5, page 35.
- Create, label, mount, and check the integrity of a file system; see Chapter 5, page 35.
- Monitor disk usage; see Chapter 5, page 35.
- Back up and restore a file system; see Chapter 6, page 105.
- Create and maintain user accounts; see Chapter 7, page 129.
- Communicate with your system users; see Chapter 8, page 173.
- Interpret system logs and determine when to "clean up" logs; see Chapter 9, page 179.
- Set up Cray system accounting (CSA) and monitor accounting functions; see Chapter 10, page 195.
- Add your CRAY J90se system to an existing network; see Chapter 11, page 269.
- Configure NIS; see Chapter 12, page 287.
- Configure NFS; see Chapter 13, page 297.

Although topics described in this guide list publications you can read to get a greater understanding of the topic, the following list identifies topics not covered in this guide that you may wish to pursue to best administer your CRAY J90sesystem.

For information about	Read
File system space monitoring	*UNICOS Resource Administration*, Cray Research publication SG–2302; `df`(1) and `du`(1) man pages
File system quotas	*UNICOS Resource Administration*, Cray Research publication SG–2302
System activity monitoring	*UNICOS Resource Administration*, Cray Research publication SG–2302; `sag`(1), `sar`(8), `sdc`(8), `tsar`(8), and `timex`(1) man pages

Automated incident reporting (AIR)	*UNICOS Resource Administration*, Cray Research publication SG–2302; `aird`(8), `airdet`(8), `airprconf`(8), `airsum`(8), and `airtsum`(8) man pages
Job and process recovery	*General UNICOS System Administration*, Cray Research publication SG–2301; `chkpnt`(1), `chkpnt`(2), and `crash`(8) man pages
Reinstalling your system software	*UNICOS Installation Guide for CRAY J90se GigaRing based Systems*, Cray Research publication SG–5296
Updating your system software	*UNICOS Installation Guide for CRAY J90se GigaRing based Systems*, Cray Research publication SG–5296
Using the `cron`(8) and `at`(8) utilities	*General UNICOS System Administration*, Cray Research publication SG–2301; `at`(1) and `cron`(8) man pages
Configuring network interfaces	*UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
Monitoring networks	*UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
Unified Resource Manager (URM) centralizes resource allocation with a formal method of communication	*UNICOS Resource Administration*, Cray Research publication SG–2302
Fair-share scheduler	*UNICOS Resource Administration*, Cray Research publication SG–2302; `shradmin`(8) and `shrdist`(8) man pages
Memory scheduling	*UNICOS Resource Administration*, Cray Research publication SG–2302
Multilevel security (MLS)	*General UNICOS System Administration*, Cray Research publication SG–2301
UNICOS message system	`explain`(1) man page
Data migration facility (DMF)	`dmmode`(2), `dmofrq`(2), `dm`(4) and `dmf_offline`(3C) man pages

Tape subsystem	*Tape Subsystem Administration*, Cray Research publication SG–2307

1.5 UNICOS on-line glossary

The `define`(1) command allows quick, on-line retrieval of Cray Research technical terms and their definitions, and terms added by your site that match a specified search term. See the following example for definitions retrieved for the word *stripe*:

```
$ define stripe
striped disk slice
  A logical disk device composed of two or more
  physical disk slices (also known as members).

striped group

  The set of disk devices that are written to
  as a single group with data blocks
  interleaved among the members for maximum
  throughput at very high bandwidth.
```

For more information, see the `define`(1) man page. For information on how to add your own terms and definitions to the glossary, see the `builddefs`(1) man page.

Cray Scalable I/O Overview [2]

The new GigaRing scalable input/output (I/O) and networking channel from Cray Research supports the next generation of I/O peripherals for the CRAY T90 series, CRAY T3E series, and CRAY J90se series of systems.

The GigaRing I/O and networking channel provides improvements in the following areas:

- Performance
- Reliability
- Maintainability
- Availability
- Scalability

2.1 Introduction to Cray scalable I/O

The Cray scalable I/O (SIO) architecture consists of a number of I/O nodes (IONs) connected by a new high-speed system channel called the *GigaRing channel*. GigaRing technology is implemented as a ring-based channel that connects multiple clients together with high-speed, point-to-point links. GigaRing clients consist of system nodes (Cray Research mainframes) and IONs that support I/O peripherals.

Figure 1. GigaRing channel topology

The GigaRing channel topology is modified from the Standard Coherent Interface (SCI) to incorporate a pair of counter-rotating rings. Each node on the ring receives and transmits data on two 32-bit rings referred to as positive and negative rings. Each node provides a host interface (system node or ION) to the rings through a 32-bit or 64-bit full-duplex client interface.

The GigaRing channel is actually a pair of counter-rotating rings. This ring topology, along with sharing of a common channel by nodes on the ring, enhances interoperability.

Each system node and ION on a GigaRing channel has a unique identifier called the *physical node address*. The physical node address is a 13-bit unique physical node ID that consists of seven ring identifier bits and six node identifier bits.

The basic unit of transfer on a GigaRing channel is called a *packet*. A GigaRing packet is limited to 32 64-bit words of payload.

2.1.1 I/O nodes

Cray Research has developed a variety of I/O nodes (IONs) to support a wide range of connectivity requirements. The two main types of IONs are single-purpose nodes and the multipurpose node. Single-purpose nodes (SPNs) support specific channel interfaces and/or devices (for more information on SPNs, see Section 2.1.1.1, page 12). The multipurpose node (MPN-1) provides an interface based on the SBus standard to support industry-standard I/O channels (for more information on MPNs, see Section 2.1.1.2, page 14).

IONs based on GigaRing technology provide access to mass storage devices such as disks and tapes, as well as to industry-standard computer networks such as High Performance Parallel Interface (HIPPI) networks. For more information on I/O products and equipment, see Table 1 and Table 2.

The Cray scalable I/O (SIO) design provides resiliency and the ability to perform *hot swaps*, in which power supplies, IONs, I/O cables, and other components that are designated field replaceable units (FRUs) can be replaced without powering down or interrupting other equipment in the PC-10 cabinet (peripheral cabinet) or on the GigaRing channel. The capability to perform a hot swap depends heavily on the system configuration; for example, the configuration of alternate paths to disks is required if IONs are to be swapped.

The CRAY SSD-T90 solid-state storage device also connects directly to the GigaRing channel, providing dynamic random access memory (DRAM) secondary storage. The CRAY SSD-T90 storage device is used primarily with CRAY T90 systems.

Table 1. Cray scalable I/O product names and descriptions

Model number	Product name	Channel capacity	Description
IPN-1	IPI-2 I/O Node	Five IPI	Can be configured to five independent IPI disk channels or as a four data plus one parity RAID-3 disk array.
BMN-1	Block Mux Tape I/O Node	Two block mux tape	Connects tape drive and tape storage subsystems.
HPN-1	HIPPI Channel I/O Node	100 Mbyte/s (32-bit)	Each node contains two 100 Mbyte/s HIPPI channels.
HPN-2	HIPPI Channel I/O Node	200 Mbyte/s (64-bit)	Can be configured to one 200 Mbyte/s HIPPI channel or one 100 Mbyte/s HPPI channel.
FCN-1	Fibre Channel I/O Node	Five fibre channel arbitrated loops (FCALs)	Supports 100 Mbyte/s burst rate per loop; each loop can operate several disks concurrently; supports RAID–3 and RAID-5.
ESN-1	ESCON I/O Node	Four independent channels per node	Supports four ESCON channels with bandwidth of 17 Mbyte/s per channel.
MPN-1	Multi-Purpose I/O Node	Up to eight SBus channels	Contains two SBuses, each supporting up to four SBus cards. SBus-based channel adapters support Ethernet network connections, FDDI network connections (requires 2 SBus channels), ATM network connections, SCSI disks, and Supervisory channel SBus (SC01).

Table 2. Cray scalable I/O support equipment

Model number	Product name	Description
ION support equipment		
PC-10	I/O peripheral cabinet	Single air-cooled I/O cabinet with DE-100 type skins for housing ION assemblies. (There are two types of PC-10 cabinets, designated as PC-10A and PC-10B. The only functional difference between these two cabinets is that the PC-10B cabinet supports WACS while the PC-10A cabinet does not.
NSR-1	ION subrack	Contains space for one to four channel adapters (does not support MPN-1).
DSF-1	Fibre disk subrack	Contains n+1 power and space for up to 10 Fibre Channel interface disk drives.
DSS-1	SCSI disk subrack	Contains n+1 power and space for up to 8 SCSI-2 interface disk drives.
DD-308	Fibre Channel interface	8 Gbyte Fibre Channel interface 3.5 inch drive.
GigaRing channel support equipment		
FOX-1	Fibre Optic GigaRing channel extender	Optical extension to the standard Cray GigaRing channel.

2.1.1.1 Single-purpose nodes

Single-purpose nodes (SPNs) connect I/O peripherals to the GigaRing channel to provide various I/O services to system nodes (CRAY T90 series, CRAY T3E series, and CRAY J90se series of systems) on the ring. The GigaRing architecture enables full connectivity between nodes on a ring.

Each SPN contains a SPARC processor, which runs the VxWorks real-time operating system with Cray Research node-specific I/O software. This provides I/O capabilities to a system on a GigaRing channel.

SPNs are housed in a stand-alone, air-cooled peripheral cabinet (the PC-10) and support the connection of various I/O peripherals such as intelligent peripheral interface (IPI) disk arrays, fibre channel disk arrays, block mux tape, and so on, to the system.

For a list of peripherals supported by SPNs, see Table 3.

Certain I/O-related peripherals do not require a unique I/O node (ION). For example, tape robots (autoloaders) are controlled through Transmission Control Protocol/ Internet Protocol (TCP/IP) network connections to the GigaRing channel. Messages are sent to the loaders through whatever node is carrying network traffic (Fiber Distributed Data Interface (FDDI), HIPPI, Ethernet, and so on).

Table 3. Peripherals supported by single-purpose nodes

ION	Devices	Description
IPN-1	DD-60	IPI-2 disk drive
IPN-1	DA-60	IPI-2 disk array (RAID-3)
IPN-1	DD-62	IPI-2 disk drive
IPN-1	DA-62	IPI-2 disk array (RAID-3)
IPN-1	DD-301	IPI-2 disk drive
IPN-1	DA-301	IPI-2 disk array (RAID-3)
IPN-1	DD-302	IPI-2 disk drive
IPN-1	DA-302	IPI-2 disk array (RAID-3)
FCN-1	DD-308	Fibre Channel disk drive
FCN-1	DD-308	Fibre Channel disk array (4 + 1 RAID-3)
BMN-1	IBM 3420	9-track tape (256-Kbyte block limit) or (IBM 3420 compatible 9-track reel tapes, such as STK 4670)
BMN-1	IBM 3480	18-track tape
BMN-1	IBM 3490	36-track tape
BMN-1	STK 4480	18-track tape
BMN-1	STK 4490	36-track tape
BMN-1	STK 4400	Libraries and robots
BMN-1	STK 9310	Libraries and robots
BMN-1	STK 9360	Libraries and robots
ESN-1	IBM 3490E	36-track tape (extended capacity)

ESN-1	IBM 3590	Magstar
ESN-1	STK 4490	36-track tape
ESN-1	IBM 3490E	36-track tape (extended capacity
ESN-1	STK 9490	Timberline
ESN-1	STK SD-3	RedWood
ESN-1	IBM 3494	Libraries and robots
ESN-1	STK 4400	Libraries and robots
ESN-1	STK 9310	Libraries and robots
ESN-1	STK 9360	Libraries and robots
HPN-1 and HPN-2		HIPPI network connections
HPN-1 and HPN-2	ND-12	Network disk
HPN-1 and HPN-2	ND-14	Network disk
HPN-1 and HPN-2	ND-40	Network disk with semaphore

For more information on SPNs, see *Cray Scalable I/O Functional Overview*, Cray Research publication SD–2208.

2.1.1.2 Multipurpose node

The peripheral cabinet also houses a special I/O node (ION) called the Multipurpose I/O Node, MPN-1. The MPN-1 provides SBus-based I/O connections for industry standard peripherals such as small computer system interface (SCSI) and Asynchronous Transfer Mode (ATM) to the GigaRing channel interface. An MPN-1 supports up to eight SBus peripheral connections. It also enables communication between the system workstation (SWS) and the GigaRing channel.

Each MPN-1 contains a SPARC processor, which runs the VxWorks real-time operating system with Cray Research node-specific I/O software. This provides industry standard I/O capabilities to a system on a GigaRing channel. MPNs are also housed in the PC-10 cabinet.

Table 4 lists peripherals supported by the Multipurpose I/O Node, MPN-1, for Cray Research mainframes.

Table 4. Peripherals supported by MPN

ION	SBus controller	Devices	Description
MPN-1	SCS-10	DD-314	SCSI-2 disk drive
MPN-1	SCS-10	DD-318	SCSI-2 disk drive
MPN-1	SCS-10	STK 4781	SCSI 18-track tape
MPN-1	SCS-10	STK 4791	SCSI 36-track tape
MPN-1	SCS-10	STK 4890	SCSI tape, Twin Peaks
MPN-1	SCS-10	STK 9490	SCSI tape, TimberLine
MPN-1	SCS-10	STK SD-3	SCSI tape, RedWood
MPN-1	SCS-10	IBM 3490E	SCSI 36-track tape
MPN-1	SCS-10	IBM 3590	SCSI tape, Magstar NTP
MPN-1	SCS-10	EXABYTE 8505	SCSI tape, 8mm
MPN-1	SCS-10	HP C1533A	SCSI tape, 4mm
MPN-1	SCS-10	Quantum DLT 4000	SCSI digital linear tape
MPN-1	SCS-10	Quantum DLT 7000	SCSI digital linear tape
MPN-1	SCS-10	STK 4400	SCSI libraries and robots
MPN-1	SCS-10	STK 9310	SCSI libraries and robots
MPN-1	SCS-10	STK 9360	SCSI libraries and robots
MPN-1	SCS-10	STK 9710	SCSI libraries and robots
MPN-1	SCS-10	IBM 3494	SCSI libraries and robots
MPN-1	ETN-10		Ethernet
MPN-1	FDI-10		FDDI networks
MPN-1	ATM-10		ATM OC3

For more information on multipurpose nodes (MPNs), see *Cray Scalable I/O Functional Overview*, Cray Research publication SD–2208.

2.1.1.3 Solid-state storage device

The next generation of Cray Research solid-state storage device, the CRAY SSD-T90 device, provides secondary storage to systems on a GigaRing channel. The CRAY SSD-T90 device uses the GigaRing channel as its only interface to other devices.

2.1.2 System workstation for GigaRing environments

The Cray system workstation (SWS) is a SPARC 5 workstation that serves in these capacities:

- As the operator workstation for CRAY T90 series, CRAY T3E series, and CRAY J90se series of systems.
- As a maintenance platform, providing support for diagnostics, ring management and maintenance, node monitoring, and hardware failure detection and isolation.
- As the system workstation, running management and maintenance software that provides support for both IONs and system nodes on a GigaRing channel.

The SWS operational model describes how a single SWS is designed to operate multiple system and IONs, which can run various levels of software.

The IONs are connected to the SWS by an Internet Protocol (IP) network connection.

For more information on the SWS and SWS operational model, see *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204.

Basic System Security [3]

This section discusses security issues in the following areas: system security (ensuring that the super-user privileges are safe), user security, partition security, and tape device access.

3.1 Related basic system security documentation

The following documentation contains more detailed information about the material presented in this chapter:

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `diskusg`(8) man page
- *General UNICOS System Administration*, Cray Research publication SG–2301
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `chown`(1), `du`(1), `find`(1), `login`(1) `su`(1), and `umask`(1) man pages

3.2 Monitoring system security

Maintaining security on UNICOS systems is largely a matter of vigilance on the part of the system administrator, who should maintain constant surveillance for potential security problems and for evidence of past security breaches. Fortunately, UNICOS includes programs that provide the necessary tools to create a set of procedures that allows you to automate much of the daily work of monitoring system security. This section discusses security issues in three areas: system security (ensuring that the super-user privileges are safe), user security, and partition security.

Note: This section does **not** apply to systems using the UNICOS multilevel security (MLS) feature systems. The MLS feature provides mechanisms to protect both system integrity and sensitive information. Design specifications for the UNICOS MLS feature were derived from the U.S. Department of Defense (DoD) evaluation criteria for trusted computer systems.

3.2.1 Super-user privileges

In the UNICOS operating system, the user identification number (user ID) of `0`, associated with the account named `root`, has special privileges and may override the security features governing the activity of normal users. Such a user is referred to as a *super user*, and the super user's powers allow the administrator great flexibility in responding to system problems and keeping the system running smoothly. The dominant security concern for a UNICOS administrator is ensuring that access to super-user privileges remains solely in the hands of the administrator and the administrator's staff. Failure to guard this access allows an unauthorized user to acquire super-user privileges. At best, one user could then look at other users' sensitive files without authorization and, at worst, an outside intruder (knowingly or unknowingly) could cause damage to the entire system.

3.2.1.1 Password security for super user

The password to the super user (`root`) account is the first line of defense against security breaches. Anyone logging in as `root` or using the `su`(1) utility to acquire super-user privileges uses this password.

Cray Research recommends the following steps to maintain secure access to the `root` account:

- The `root` password should not be obvious and should be very difficult to guess. Do not use a normal word in any language that might be known to a majority of the system's users. Additionally, capitalizing a random letter or two (not the first letter of the password), or including a punctuation character or a numeral in the password, or both, helps to keep super-user privileges safe from an intruder who is trying to guess the `root` password.
- The `root` password should be changed frequently, at least once a month.
- The `root` password should never be written down anywhere.
- The `root` password should be known to as few people as possible. Generally, these should be the system administrator and the administrator's staff.

Use of the `root` password can be monitored, and potential security breaches caught, by compiling the `su` utility so that it logs each use of the utility in the `/usr/adm/sulog` file. The administrator can then use the `grep`(1) utility to generate periodic lists of successful and unsuccessful attempts to assume super-user privileges by use of `su`. These lists can be compared against the names of users known to have valid authorization, alerting the administrator to

unauthorized super users (a security breach) or users who are repeatedly trying to gain super-user privileges (a security risk).

3.2.1.2 Physical security

A person with access to the SWS and ION consoles and a knowledge of how to halt and reboot the system could do so and thus acquire unauthorized super-user privileges.

To guard against this possibility, Cray Research recommends that the SWS and the ION consoles and the system itself be physically accessible only to those persons with genuine need for that access. If this is not possible, they should at least be monitored to prevent unauthorized persons from attempting to enter commands on the system console.

3.2.1.3 setuid programs

An executable UNICOS program may have the setuid bit in its permissions code set, indicating that whenever any user executes the program, the program runs with an effective user ID of the owner of the file. Thus, any program that is owned by `root` (user ID `0`) and has its setuid bit set is able to override normal permissions, regardless of who executes the program.

This feature is useful and necessary for many UNICOS utilities and commands, but it can be a potential security problem if an astute user discovers a way to create a copy of the shell owned by `root`, with the setuid bit on. To avoid this possible security breach, the administrator should make regular checks of all disk partitions on the system for programs that have a setuid (or setgid) of 0.

The `find`(1) utility can generate a list of all setuid/setgid 0 files on the system (if all file systems are mounted), as follows:

```
find / \ -user 0 -perm -4000 -o -group 0 -perm -2000 \ -print
```

This list may be compared against a list of known setuid/setgid 0 programs. Any new setuid/setgid 0 programs that are not on the known list and whose creation you cannot account for may indicate a security breach.

The administrator should check the list of known setuid/setgid 0 programs regularly to ensure that none have been modified since the last check and that any modifications that have been made are known (in other words, were made by the system administrator or a member of the administrator's staff). Unknown modification of a setuid/setgid 0 program may indicate a security breach.

Finally, the list of known setuid/setgid 0 programs should be checked to ensure that write permission on each file is properly restricted.

Because checking the entire system for setuid/setgid programs uses a large amount of CPU time, Cray Research recommends that this check be performed during off-peak hours. Use of the `cron`(8) or `at`(1) utility to perform the check automatically and to notify the administrator of any suspicious results should make the task unobtrusive.

3.2.1.4 `root` PATH

The `PATH` environment variable consists of a list of the directories searched by the shell for typed commands. This means that the `PATH` for the `root` account must have the following security features:

- It must never contain the current directory (.).
- All directories listed in the `root PATH` must never be writable by anyone other than `root`.

The `root PATH` is set in two separate places:

- The `/.profile` file sets the `PATH` for `root` whenever `root` logs in on the system console.
- The `su`(1) utility changes the `PATH` after a user has successfully entered the `root` password to assume super-user privileges.

Both places should be monitored from time to time to make sure they have not been changed since the last approved change known to the administrator.

Keeping the current directory out of the `root PATH` is somewhat inconvenient; super users must remember to precede the names of any programs or scripts they want to run from their current directory with `./`, as in `./newprogram`, because the shell does not search the current directory for a command name. However, convenience should not take precedence over system security. Failure to follow these guidelines leaves the system open to a security breach.

For example, suppose a knowledgeable user creates a program that mimics a commonly used system utility, such as `ls`(1). In addition to performing the expected system function (listing the files in the current directory), the new `ls`(1) utility makes a copy of a program such as `ksh`(1) and turns on the setuid bit on the copy. An unsuspecting super user with the current directory in `PATH`, having changed directories to a user's directory and inadvertently run the

bogus `ls`, then creates a setuid 0 shell, which gives anyone executing it complete control over the system.

3.2.2 User security

In addition to general system security, the administrator should ensure that files owned by system users are secure from examination and modification by other users.

3.2.2.1 The `umask` utility

The system default umask value is normally set in the `/etc/profile` file by using the `umask`(1) utility. It allows you to choose the permissions that will typically be set when users create new files. For example, a umask value of 027 means that the group and other write permissions and the other read and execute permissions are not set when a user creates a file. For possible umask values and descriptions, see the `umask`(1) man page.

In general, only the owner of the file should have write permission, which makes a default umask value of 022 appropriate. If you do not want members of a given user group to be able to read the files of other user groups, using a umask value of 026 to remove other read permission is recommended.

You should choose a umask value that restricts default access permissions to a level appropriate to the desired security of the system. However, because users can override the default value by using the `umask` utility themselves, do not make the default umask value too stringent, as users may find that the default value interferes with their work. For instance, if two users are working on a joint project, and each needs access to the other's files, they may want to change their umask values so that, on any new files they create, the permissions will be more open.

3.2.2.2 Default `PATH` variable

The default `PATH` variable for the system's users is set in the `/etc/profile` and `/etc/cshrc` files. It specifies the system directories that will be searched for command names typed by the users.

The users expect to be able to execute programs in the current directory without having to precede the program name with `./` to explicitly indicate the current directory. However, many UNICOS systems traditionally place the current directory first in the `PATH`, which can make the users vulnerable to a security

breach, as described in Section 3.2.2.4, page 22. The current directory should thus be the last entry in the default PATH, after the normal system directories.

3.2.2.3 User groups

User security can be enhanced by the careful placement of users into groups. In general, it is a good idea to use factors external to the system when deciding upon the placement of users into groups. Some examples might be the following:

- Members of a specific software project
- Accounts for a client company purchasing system time
- Intercompany divisions

Having many groups, each containing a small number of users, is safer than having fewer groups, each with large numbers of users with access to each other's files. Members of most logical groups (for example, members of a software development project) may want to share files with one another, and the default umask should permit this.

To prevent inappropriate sharing of data, rather than create a default "other" or "miscellaneous" group for a user who does not fit elsewhere, you should create a group with only one user in it. Because users may belong to more than one group, and groups are active simultaneously, you may also choose to create a separate group for each individual user at the time you create the account, and then add users to additional logical groups as necessary.

3.2.2.4 File-owner fraud

Neither the listed owner ID of a file nor its location in the directory tree always leads to the actual creator and owner of the file. That is, users tend to think of the files residing in their home directory as their only files, even though they may own files in another home directory, such as those being used for a project involving several other users. Conversely, it may not be completely appropriate to count files that reside in one user's home directory tree but are owned by another user.

Users may realize this confusion and try to avoid a disk usage monitoring system by using the `chown`(1) utility to change the ownership of some of their files to another user (most likely one who will cooperate and give the file back when requested). Nevertheless, `diskusg`(8) and `du`(1), when used together, provide a general idea of the users who are perennial problems.

3.2.2.5 Login attempts

Unauthorized users might attempt to gain access to the system by making repeated attempts to login. To help prevent such attempts, you can configure the number of bad login attempts that will be allowed before the `login` terminates. By default, the system will allow an unlimited number of bad login attempts. To put a limit on such attempts, edit the `/etc/config/confval` file (see `login`(1)).

3.2.3 Partition security

When administered properly, the UNICOS file system should provide adequate protection for user and system files. You can enhance system security, however, by mounting partitions only when they are needed. In particular, if there are users who will be allowed dedicated time on your system, you can provide extra protection for those accounts by not mounting the file systems that contain other users' accounts.

To prevent users from accessing disk partitions directly, without going through the UNICOS file system, the disk device nodes in `/dev/dsk` and `/dev/rdsk` must never be readable or writable by anyone other than `root`.

3.3 Tape device access

For CRAY J90se systems, you should use the tape daemon character-special tape interface. The character-special tape interface provides unstructured access to the tape hardware similar to the traditional UNIX method of accessing tape devices. It is useful in performing specific tasks, such as the following:

- System administrators use the interface for routine tape manipulations such as copying. To manage their tapes, they can use standard UNIX commands and `ioctl`(2) requests.
- Programmers use the interface to develop file management applications.

For more information on tape devices, see the *Tape Subsystem Administration*, Cray Research publication SG–2307, and the *Tape Subsystem User's Guide*, Cray Research publication SG–2051.

Startup and Shutdown [4]

This section includes procedures to do the following:

- Start the CRAY J90se mainframe and bring up UNICOS to a multiuser run state mode (startup; also called *booting*).
- Bring UNICOS back to single-user mode (shutdown).

This section also briefly describes various aspects of single user and multiuser mode.

If you have access to a windowing environment, UNICOS provides a point-and-click, X Window System based interface to the UNICOS Installation / Configuration Menu System.

4.1 Related startup and shutdown documentation

The following documentation contains more detailed information about the material presented in this section:

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022
- *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204
- *SWS-ION Reference Manual*, Cray Research publication SR–2205
- *SWS-ION Release Overview*, Cray Research publication RO–5292
- *SWS Operating System and Devices Installation Guide*, Cray Research publication SG–5293

4.2 Booting the default configuration

It is recommended that you first boot the system with the default configuration. Later, you can make and test basic configuration changes, and finally configure file systems and make other site-specific changes. Using this approach can help isolate any errors that might be introduced during the configuration process.

Note: The CRAY J90se mainframe, SWS, and MPN must be correctly cabled in order to boot the CRAY J90se system. In addition, the SWS and MPN must be booted and running. For more information, see the *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204.

To boot your CRAY J90se mainframe over a GigaRing channel, execute the `bootsys`(8) command. The `bootsys` command boots one or more system components. A system component can be a ring, I/O node, or mainframe. If you do not specify any options, `bootsys` boots the I/O nodes, initializes the rings, and boots the mainframes (all the system components that are specified in the `topology`(5) file). The `topology` file identifies rings, I/O nodes, and mainframes and the manner in which they are connected. See the `topology`(5) man page for details. If you specify the `-c` option, a console is started for each mainframe that is booted.

Caution: Do not execute `bootsys`(8) on a running system. The system will crash, possibly causing a loss of data or corrupted file systems. Before executing this command, shut down UNICOS according to the methods described in UNICOS administrator documentation.

`bootsys` uses the defaults provided by the low-level commands, such as `bootj90`, unless they are overridden by specifications in the `options` file. For more information about the `options` file, see the `options`(5) man page.

If you specify one or more system components on the `bootsys` command line, only those components are booted. The order in which components are specified on the command line is insignificant.

For more information about the `bootsys`(8) command and about the SWS, see the `bootsys`(8) man page and the *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204.

Note: By default, the CRAY J90se system is booted in single-user mode. (The default mode, or *run-level*, is controlled by the `run_level` set on the UNICOS `inittab init` default line (normally S). At this point in the initialization process, the system should remain in single-user mode.

At this point, your system is running the default configuration.

Procedure 1: Booting your system

1. To boot your mainframe over a GigaRing channel, execute the `bootsys`(8) command at the prompt (if you specify the `-c` option, a console is started for each mainframe that is booted):

   ```
   # bootsys -c
   ```

After executing the `bootsys` command, UNICOS is in single-user mode.

Only a few processes are running: `init`, `swapper`, `idle`, and `sh`. The `root` (/) file system is the only file system available.

When you are in single-user mode with only the `root` (/) file system available, you must do all editing by using the `ed` editor, because the `vi` editor is located in the `/usr` file system. If you want to use the `vi` editor before going to multiuser mode, you must mount the `/usr` file system. Before going to multiuser mode, or if you intend to work in single-user mode, you should check the `root` (/) file system by using the `fsck` command.

The first time you use the `vi` editor, you may see the following error message:

```
I don't know what kind of terminal you are on - all I have is
'unknown'.[Using open mode]
```

Type the following command lines:

```
# TERM=xterm
```

(or sun-cmd, if you are using the command tool)

```
# export TERM
# resize
# echo $TERM
# echo $SHELL
```

2. Bring the system to multiuser mode by signaling the `/etc/init` process to change to a new run level by entering the following command:

```
# /etc/init 2
```

Multiuser mode is usually run-level 2. Although you can configure a system to run in multiuser mode at any level between 0 and 6, you may want to reserve some states for the future.

As the system boots into multiuser mode, output is produced on your terminal. You will be asked whether you want to run `mkfs /tmp (y/n)`, which you must respond to for the process to proceed. At approximately midpoint in the process, the `Administrative cleanup` message appears. This message indicates that the system is moving into multiuser mode

properly. You will be prompted for the system date, which is an optional entry. When the system boot is complete, you will see the following prompt:

```
Console Login:
```

4.2.1 Multiuser mode

Traditionally, run level 2 is the system's primary run level for multiuser mode. Among the initializations generally performed for multiuser mode are the following:

- Recording system start-up time in `/etc/wtmp`.
- Mounting all file systems required for normal system operation. This includes the regular system file systems (`/usr` and `/tmp`), the file system or systems that contain the home directories' `/tmp` file system of the system's users, and other file systems that contain files to which the users must have access.
- Removing any lock files that may interfere with normal system operation (for example, a lock file for a system daemon).
- Running daemons that provide various system services. The list may include, but is not restricted to, the following:
  - `errdemon`
  - `slogdemon` (for the UNICOS multilevel security (MLS) feature)
  - `cron`
  - `tapestart` (for online tapes)
  - `syslogd`
  - `nqsdaemon` (for NQS)
- Running the `netstart` script to initialize the system's TCP/IP network connections.
- Starting system accounting.
- Moving or truncating log files (for example, `/usr/lib/cron/log` or `/usr/spool/nqs/log`) to prevent them from growing without limits.
- Allowing users to log in.

4.2.2 Typical tasks you can perform while in multiuser mode

The following are some typical system administration tasks that you can perform while UNICOS is running in multiuser mode. The most important areas to monitor include how efficiently the system is performing and the rate at which system resources are being consumed.

- Checking which file systems are mounted by using the `/etc/mount` command.
- Checking all mounted file systems to ensure that no mounted file system consumes all available free disk blocks by using the `/bin/df` command or the `/etc/fsmon` file system monitor.
- Checking the number of system users by using the `who` command. To identify idle users, enter `who -u`. To determine the number of users, enter `who | wc -l`. To generate the number of users and a list of their names, enter `who -q`.
- Informing users of system changes by using `/etc/wall`.
- Monitoring how your UNICOS system is running by using the `/usr/bin/sar` utility. The `/usr/bin/sar`(1) utility has many options used to gain information about disk performance, character list buffers, CPU performance, and IOS throughput. The most useful options for a system administrator include `-d` (disk), `-x` (IOS), and `-v` (critical internal system table sizes). For more information, see the `/usr/bin/sar`(1) man page.
- Checking all running processes by using the `ps`(1) command to determine whether a process is using an abnormally large amount of CPU time. The `-eaf` options generate a full listing for all running processes.
- Checking the contents and size of your UNICOS error logs. Usually, error logs are found in the `/usr/adm` directory. Also, ensure that the error logging daemon is executing and that IOS disk errors are being logged into the `/usr/adm/errfile` file. For log information, see Chapter 9, page 179. For details on disk error reporting, see the `/etc/errpt`(8) man page.
- Checking `mail` by using the `/bin/mail` command while logged on to `root`, or the login that receives requests to restore files. If a problem occurs, the system itself sometimes sends mail to `root`.

4.2.3 Dedicated system

It is sometimes necessary to provide dedicated system time so that a particularly large or time-critical job can run unencumbered by other user processes. There also will be times at which system development work requires that the system be brought up as though it were running in multiuser mode, when access to the machine is actually restricted to the system staff. To lock out all users except yourself, use `/etc/udbrstrict -r -L` *your_userid*. Do not use just `/etc/udbrstrict -r`, because this limits logins to only root, which can then be done only on the console device. For more information about the UDB `ue_permbits` field, see *General UNICOS System Administration*, Cray Research publication SG–2301.

4.2.4 Run-level configuration

A *run level* is a software configuration of the system. Each run level allows only a selected group of processes to exist. Although run levels are most commonly used to configure the system in single-user or multiuser operation modes, thoughtful management of the run-level configuration on the system is a convenient method of tailoring the system's resources to accommodate users' needs.

Two main modes of operation exist for UNICOS: single-user and multiuser. Single-user mode is always indicated by run level s or S. Multiuser mode is typically run level 2, although it may be level 0 through 6.

One common use of the `/etc/inittab` file is to set up a run level so that certain procedures are followed automatically only the first time a run level is entered. For example, usually you are asked to verify the date and to check the file systems the first time you change your system to multiuser mode. These actions are caused by an entry in the `inittab` file. Subsequent changes in run level do not result in this procedure automatically unless you specifically change the `inittab` file.

4.2.4.1 Changing run level

As system administrator, you can change the run level by issuing the following command; *level* is the run level you want to initiate:

`/etc/init` *level*

The `/etc/inittab` file controls the specific actions that occur when a run level is initiated. The following subsections discuss the strategies for using run levels for various purposes.

4.2.4.2 Strategies for using run levels

Successful use of run levels requires that you think through the requirements for the system and tailor the initializations of the various run levels to provide for convenient transitions from one run level to another.

All systems have a single-user mode (for system work that must be performed unencumbered by the presence of other users on the system) and at least one multiuser mode. If the system is restricted at various times to dedicated use by one or more users, you should devote one or more run levels to initializing the system for this dedicated use. In all cases except for single-user mode (which requires little or no initialization), the `rc` (see the `brc`(8) man page) script performs initialization.

Procedure 2: Shutting down UNICOS

To shut down UNICOS, execute the following steps:

1. Make sure that you are logged in as `root` and that you are in the `root` (`/`), `/etc`, or `/ce` directory; to change to the `root` (`/`) directory, enter the following command:

```
# cd /
```

2. You may want to send active users a special message about when the system will be shut down. The `/etc/shutdown` script is designed to return the UNICOS system to single-user run state in a clean, orderly manner. The `/etc/shutdown` script prompts you for a message that will be sent to all users; if you want to include a message, use the `wall`(8) command to provide the message (see section Chapter 8, page 173). Before executing `/etc/shutdown`, you can use the `ps -eaf` command to see processes that are running, and the `who -u` command to see whether people are actively using the system. The `shutdown`(8) command uses the following format:

   `/etc/shutdown` *grace-period-in-seconds*

   The following command instructs the system to wait 5 minutes (300 seconds) before terminating all processes and shutting down the system:

```
# /etc/shutdown 300
Do you want to send your own message?  (y or n):  y
Type your message followed by a <Return> and then ctrl d....
System shutting down in 5 minutes for test time-Please log out now.
CONTROL-d
```

The time it takes for the shutdown to complete depends on the number of processes that must terminate and file systems that must be unmounted; however, the shutdown process may take 3 to 5 minutes.

When the shutdown program is complete, the following message is displayed, and you should type the following highlighted commands:

```
Message: INIT: SINGLE-USER MODE.
# /bin/sync
# /bin/sync
# /bin/sync
# /etc/ldsync
```

(if you are using ldcache)

```
# df
```

(to verify that all file systems have been unmounted cleanly)

```
# /bin/sync
```

At this point, you are in single-user mode but UNICOS is still running. You can perform any system administration work as necessary.

4.2.4.2.1 Single-user mode

Many system maintenance, modification, testing, configuration, and repair procedures are performed while the system is in single-user mode to protect system users from potential instability and to ensure that user processes do not interfere with the system's work while it is in progress. Therefore, the purpose of performing any initialization before the system is in single-user mode is to ensure that the system is known to be in an idle state.

When the UNICOS system is in single-user mode, all network connections and hard-wired terminals are disabled, and only the console terminal can interact with the system. This mode of operation lets you make necessary changes to

the system without doing any other processing. When UNICOS is in single-user mode, the # symbol (or `sn`*xxx*#) is the system prompt.

Typically, the system is brought into single-user mode either following a system boot or by using the `shutdown`(8) command. In neither case should any user processes be running after the system is in single-user mode (no user processes will have started following a boot, and `shutdown` kills all user processes before entering single-user mode). Thus, there should be no need for initialization related to user processes when the system enters single-user mode.

As an extra measure of protection against inadvertent damage done to a mounted file system by single-user mode development work or testing, you should unmount all file systems except the current `root` file system. Traditionally, users doing the system work or testing while in single-user mode mount only the partitions they require. To help with this aspect of system work, you can provide a script in `/etc` that mounts the file systems that contain system commands not usually found on the root partition (the `/usr` file system) and the `home` user file system directories of the system staff.

File Systems [5]

5.1 UNICOS file systems

All files that are accessible from within the UNICOS system are organized into *file systems*. File systems store data in formats that the operating system can read and write. This section describes how to plan, configure, create, and monitor UNICOS file systems. As a system administrator, you must do the following:

- Plan the file systems
- Configure the file systems
- Create the file systems
- Monitor disk usage to ensure that users have sufficient free space on their file systems

No single configuration of available disk drives into file systems and logical devices will prove best for all purposes. Optimizing file system layout is usually an iterative process: make your best attempt, then run it for a while and monitor it for disk use monitoring information (see Section 5.4, page 40). You will adjust your configuration based on information you gather about your users' needs. As the needs of your users change, you will reconfigure your file systems to retain a well-balanced configuration. In the absence of a set of absolute rules, the facts and guidelines presented in this section will help you decide on a file system plan for your system.

Note: Although all UNICOS file systems have some common aspects, file system creation and organization varies on Cray Research systems. If you have Cray Research systems other than CRAY J90se systems, see *General UNICOS System Administration*, Cray Research publication SG–2301, and *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303, to determine differences in file systems and how to configure them.

5.2 Related file systems documentation

The following Cray Research publications contain information related to this section:

- *General UNICOS System Administration*, Cray Research publication SG–2301

- *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303
- *UNICOS Resource Administration*, Cray Research publication SG–2302
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `df`(1), `du`(1), `mkdir`(1), and `rm`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `dir`(5), `dsk`(4), `fs`(5), `fstab`(5), `inode`(5), `ldd`(4), `mnttab`(5), and `pdd`(4) man pages
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `ddstat`(8), `diskusg`(8), `dmap`(8), `econfig`(8), `fsck`(8), `fsmap`(8), `fuser`(8), `labelit`(8), `mkfs`(8), `mknod`(8), `mount`(8), `stor`(8), and `umount`(8) man pages

5.3 An overview of file systems

A *file system* is a group of addressable disk blocks used to store UNICOS directories and files. A file system can either be mounted (accessible to users) or unmounted (unavailable to users). The system mount table records which file systems are currently mounted. The mount table is named in `/etc/mnttab`.

File systems are in an inverted tree structure, with a file at each node of the tree. A base file system named `/` or `root` always exits. The `root` file system is always available for use and contains required files needed for booting UNICOS. When a file system is mounted, it is attached to a mount point (directory), which might be part of another file system. Mounting file systems on each other creates a series of cascading directories below the `root` file system.

To maintain data consistently and correctly, individual files are in **only** one file system. Each file system resides on unique physical locations on a physical disks, and UNICOS carefully controls the file systems. This isolation of data prevents security violations and data corruption.

Note: When you are in single-user mode, with only the `root` (`/`) file system available, you must do all editing by using the `ed` editor. This is because the `vi` editor is located in the `/usr` file system. If you want to use the `vi` editor before going to multiuser mode, you first must check (using `fsck`) and mount the `/usr` file system.

5.3.1 Terminology

This subsection provides terminology associated with file systems. Everything is viewed by UNICOS as a file, whether it is an ASCII file of user data or a physical disk device. UNICOS supports five types of files: regular, directory, block special (such as a disk drive), character special (such as a tape drive), and FIFO special.

Regular files	These files hold user data of various formats.
Directory files	These files contain the names of "regular" files and other directories, along with their corresponding inode numbers. When block or character special files are accessed, device drivers are invoked that communicate with peripheral devices, such as terminals, printers, and disk drives. FIFO special files, also called *named pipes*, allow unrelated programs to exchange information.
physical device	This is a tape or disk device. Physical disk devices are read from and written to in units of 512-word (4096-byte) blocks. The smallest unit of I/O disk devices can perform is one block. UNICOS file systems are defined in regions of contiguous blocks called *slices*. File systems can be built on many different slices.
partition	This is one slice on one physical device.
logical device	One or more slices creates a logical device. Although a logical device appears to be one device, its slices can be located across several physical devices. Logical devices become file systems when the disk is initialized with a file system structure by using the `/etc/mkfs` command.
inode	This contains information such as permissions and file size for all types of files.
Regular files	These files are composed of readable characters; these can include data, text, or program files that can be executed.
special files	These files are not composed of readable data. Instead, they serve as a connection between a

path name (such as `/dev/dsk/root`) and the device handling routines in the UNICOS kernel to control I/O to the device.

- *Block special files*: Block special files are used to communicate with file systems. The drivers for these files process data in blocks. Block devices have a minimum transfer unit size of one block (4096 bytes or 512 words). All I/O for CRAY J90se file systems use block special files. You can address block special files and their related devices by using various I/O techniques.
- *Character special files*: The drivers for these files process "raw" data, bypassing UNICOS kernel buffering. Data is transferred directly between the user's memory area and the physical device. UNICOS character special files are used to support tape and tty connections, among others. You can use character special files and their related devices only for sequential I/O.

major device number
: This number refers to the type of device. Major device numbers are used as an index into a table of device drivers appropriate for that kind of physical device. These routines open, close, read, write, and control a physical device.

minor device number
: This number is used by the appropriate driver (determined by the major number) to specify a particular logical disk device, tape drive, or physical device. Minor device numbers range from 0 to 255 and must be unique within the same major number; however, numbers 250 through 255 are reserved for use by the operating system. For example, on CRAY J90se systems, minor number 253 is used for the `ce` partition. For additional information, see *General UNICOS System Administration*, Cray Research publication SG–2301.

All UNICOS special files are located in the `/dev` directory or one of its subdirectories. Your CRAY J90se system is initially supplied with sufficient UNICOS special files for most basic device configurations. You should create additional (block) special files to match your unique file system layout. All special files are created using the `mknod` command.

When a device special file is examined by using an `ls -l` command, the device special file's major and minor numbers, separated by a comma, are displayed where the number of bytes would appear for a regular or directory file.

The following are the directory paths of some UNICOS special files and scripts for file systems:

File	Description
`/dev`	Directory of special files and subdirectories of other special files.
`/dev/dsk`	Directory that contains all block special files that represent logical disk devices for current file system configuration. The major device number is 34 for disk devices. A `b` in the directory permissions field (`ls -l` output) indicates a block special file.

5.3.2 UNICOS file system structure

UNICOS file systems are often stored on several different physical devices. When you configure a file system, you first specify the physical locations that compose the file system. This information is stored in the `/opt/CYRIos/`*sn9xxx*`/param` file, and it is written using the menu system. You can store file systems on disk or in random-access memory (RAM), or a combination of both.

The definition of your system's logical and physical disk devices is defined in the `/opt/CYRIos/`*sn9xxx*`/param` file. You must initialize that area of disk, using the `mkfs` command.

The `mkfs` command structures the physical disk area with the following elements:

Element	Description
Super block	Used to store file system size and the number of inodes in the file system, as well as internal parameters such as allocation strategy. It is updated when the `mkfs` or `setfs` command is run. Several copies of the super block are kept for robustness (redundant copies make it easier to recover information if a catastrophic failure occurs). The super block is read into memory when the file system is mounted, and it is flushed to disk when it is modified or when the file system is unmounted.

Inode region	Each file in a mounted file system is identified with a unique pointer called an inode number. The *inode* itself contains file information such as permissions, file size, whether the file is a directory, and so on. The inode region contains a maximum of 32,768 inodes. You can have a maximum of four inode regions per partition.
Dynamic block	A block that contains the file system information that changes during system operation. The dynamic block contains block counts for a specific partition. This information is flushed to disk when the file system is modified, when the file system is unmounted, or when `sync`(2) is executed.
Block allocation bit map	A bit map that controls block allocation across the entire file system.
Map blocks	A bit map of the disk sectors.
Partition data blocks	The disk area for directories and user data.
`setfs`(8) command	This command changes dynamic information in the file system super block without remaking the file system.

5.4 Commands for examining files and file systems

One of the system administrator's most important responsibilities is to monitor system disk usage and to ensure that users have sufficient free space on their file systems to accomplish their work.

To display information about files and file systems, use the following commands:

Command	Description
`/usr/lib/acct/diskusg`	Summarizes the disk usage on the file system you specify by file ownership and identifies users who are using most of the space on a file system. The `/usr/lib/acct/diskusg -h` command is the preferred command for summarizing disk use; the `-h` option provides headings.

`/etc/econfig`

The `-d` option prints out `mknod` commands to generate file systems. You may want to do this when you first get your system in case you have to manually recreate these nodes.

`/etc/dmap`

Displays information about the configuration of a disk subsystem.

`/etc/bmap`

Displays which file is using the block on a given file system.

`/etc/fsmap`

Displays file system free block layout.

`/bin/df`

Displays the number and percentage of free blocks available for mounted file systems; the `-p` option is particularly useful.

`/bin/du`

Summarizes the disk usage on a file system, by directory structure. The `-s` option provides the total number of disk blocks used under each directory (or file) specified.

`/etc/errpt`

Processes errors report generated by `errdaemon`. This UNICOS command is for disk hardware errors; `errpt -a` produces a detailed list of errors; `errpt -d` *device-type* produces list of errors for the specified device type.

`/etc/mount`

Displays the list of all currently mounted disk files and their mount points when issued without arguments.

`/etc/stor`

Sorts special files by physical device numbers and writes to standard output information about starting and ending disk addresses and sizes.

`/ce/bin/olhpa`

Displays hardware errors by reading the `/usr/adm/errfile` file. The `-d` option lets you view disk errors.

`/bin/fck`

Displays information concerning the names files that are gathered from reading the inode and the address blocks from the block special device in the `/dev/dsk` directory.

`/etc/ddstat`

Displays configuration information about disk type character and block special devices.

`/etc/pddconf`

Controls the state of an IOS model E disk drive.

`/etc/pddstat`

Displays information from the disk table, which controls disk I/O in an IOS model E.

`/etc/sdstat`

Displays disk activity information for all types of disks (xdd, hdd, and pdd).

`/etc/sdconf`

Controls the state of disk drives for all types of physical disk devices (xdd, hdd, and pdd).

5.5 File system planning

When planning a file system, you must decide which parts of a disk will be used for each file system. This section provides minimum size requirements for file systems as well as device recommendations.

First, a UNICOS system administrator must plan which slices of a physical device will be used to make up each file system, as well as which file systems should be striped, if any, and which should be banded. (For information about disk striping, see Section 5.6, page 45, and for information about disk banding, see Section 5.7, page 46.) You must consider disk capacity and transfer rate, as

well as file system size and usage, along with the number of users and types of applications your Cray Research representative installed on your system.

The file systems listed in this subsection are found on most UNICOS systems.

Note: The disk storage discussed is the **minimum** amount of storage required, not the **recommended** amount. The information is provided here to help you plan your file system space needs.

For up-to-date information regarding minimum file system and amount of free blocks needed to install other Cray Research software products, see the *UNICOS Installation Guide for CRAY J90se GigaRing based Systems*, Cray Research publication SG–5296.

5.5.1 The `root` (`/`) file system

Size recommendations: You should define a **minimum** region of 150,000 blocks to hold your `root` file system.

If possible, the remaining blocks on the same physical device used for your `root` (`/`) file system are good locations for your smaller or lesser used file systems.

For details on peripherals supported by single-purpose nodes and multi-purpose nodes, see Section 2.1.1, page 10.

5.5.2 The `/usr` file system

Size recommendations: You should define a **minimum** region of 400,000 blocks. The contents of the `/usr/adm` subdirectory tend to grow very large because the UNICOS accounting data is kept here.

Device recommendations: To avoid contention, you should configure the `/usr` file system on a different controller, disk, and IOS than the one on which the `root` (`/`) file system resides.

Be sure to size your `/usr` file system to meet the space requirements for any software to be installed later.

5.5.3 The `/usr/src` file system

Size recommendations: The recommended **minimum** value for a CRAY J90se system is 200,000 blocks. This size is sufficient to hold all of the files necessary to relink the UNICOS kernel. You also must allow enough space in your

default value to handle additional Cray Research asynchronous products you may load on this file system (for this information, see your UNICOS installation guide and related errata).

5.5.4 The /opt file system

Size recommendations: The recommended **minimum** value for a CRAY J90se system is 300,000 blocks. You also must allow enough space in your default value to handle additional Cray Research asynchronous products you may load on this file system (for this information, see your UNICOS installation guide and related errata).

5.5.5 The /tmp file system

Size recommendations: You should define a **minimum** region of 50,000 blocks. You may want to allocate `/tmp` and `/home` in a 2 to 1 ratio (2 blocks `/tmp` per 1 block of `/home`).

Device recommendations: If two or more IOSs are present, to avoid contention, you should configure `/tmp` and `/home` on a different controller, disk, and IOS than the one on which the frequently accessed system file systems and logical devices reside. This file system is best handled by allocating slices from several different disks to compose the logical file system. This disk allocation strategy is called *banding*.

5.5.6 The swap device

Size recommendations: You should configure the `swap` device to be the **minimum** number of blocks, as follows:

Central memory size	Minimum blocks for swap device
256 Mbyte/32 Mwords	187,500 blocks
512 Mbyte/64 Mwords	375,000 blocks
1,024 Mbyte/128 Mwords or larger memory	750,000 blocks

Device recommendations: If possible, put the `swap` device on a separate drive from either the `root` (`/`) or `/usr` file system.

If your system's job mix swaps frequently, you may want to configure your swap device as a striped device. For more information about striping, see Section 5.6, page 45, and *General UNICOS System Administration*, Cray Research publication SG–2301.

5.5.7 The dump device

Size recommendations: The **minimum** size of your `dump` device should be a little larger than the amount of memory you actually want to examine to allow an additional 1200 blocks for a `dump` header. You should start with a minimum of 50,000 blocks for the dump size.

You cannot stripe the `dump` device because it is not a file system.

Place the `/dev/dsk/dump` file system at the end of the disk on which it resides. This prevents inadvertent writes into another file system.

5.5.8 The back-up `root` (`/`) and back-up `/usr` file systems

Size recommendations: The backup `root` (`/`) (called `rootb`) and back-up `/usr` (called `usrb`) file systems are equal in size to the original `roota` (`/`) and `/usra` file systems.

Device recommendations: Keep `rootb` and `/usra` file systems on different disk drives, controllers, and IOSs, if possible, from `roota` (`/`) and `usrb` file systems. You also should keep the backup version of a file system on a different drive (and controller and IOS, if possible) from your original file system.

To keep the `rootb` file system updated to match the `roota` file system, you can run the `dd` command as a `cron` job. For details, see the `dd` and `cron` man pages.

5.5.9 The `/home` file system

The size and location of your `/home` file system is site specific. A **minimum** of 50,000 blocks is recommended. The file system is used for login account home directories.

5.6 Disk striping

A striped device can be made up of two or more of the same type of disk drives or can be logically the same type. The number of blocks must be the

same in each slice. Several drives are combined together in one logical unit (known by the name of the first slice name), which makes simultaneous I/O operations possible.

Disk striping allows an increase in the amount of data transferred with each I/O operation. In effect, the I/O rate is multiplied by the number of disk devices in the striped group. On baseline systems, however, only `swap` is recommended as a striped disk. Striping is best used only for large I/O moves, such as swapping.

Note: You should not run `ldcache` on a swap file.

5.7 Disk banding

disk banding is the process of distributing a file system across several disk drives. The physical devices do not have to be of the same type or have their block ranges begin at the same block or be of the same length.

5.8 Configuring your devices and their file system allocation

The system configuration file that configures disks is the `/opt/CYRIos/`*sn9xxx*`/param` file on the SWS. The configuration specification language (CSL) is used to define the configuration and parameter settings that are used at boot time. CSL defines the following:

- Number of IOSs
- Mapping IOS channels to specific CPUs
- Physical device attributes and slice layout
- Logical grouping of physical disk slices
- System-defined devices
- Network configuration

  Note: If you use the menu system to configure these settings, it will automatically generate the CSL statements in the `/opt/CYRIos/`*sn9xxx*`/param` file that describe your system configuration.

The following sections provide information and procedures to help you do the following:

- Determine how to configure file systems by using CSL

- Determine the devices that are provided on your system when you receive it and how they are allocated to file systems
- Modify your system configuration
- Create file systems

5.9 Network disk array configuration

For information on configuring network disk arrays (HIPPI disks), see *Network Disk Array (HIPPI Disk) Configuration Options and Performance*, Cray Research publication SN–2185.

5.10 CSL syntax

There are three classes of tokens that make up CSL: identifiers, constants, and operators/separators. White space (horizontal tabs, new lines, carriage returns, and spaces) separates individual tokens.

5.10.1 Identifier

A revision 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: keywords and object identifiers. *Keyword identifiers* have special meaning in CSL and cannot be used to name other

things. For a list of reserved keywords, see Appendix A in the *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303.

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.

5.10.2 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 0 (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
```

5.10.3 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.  */
```

5.11 Placement of CSL statements

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

Section	Description
`gigaring`	GigaRing configuration parameters
`mainframe`	Physical mainframe characteristics parameters
`unicos`	UNICOS kernel parameters
`filesystem`	Physical storage devices and file system layout parameters
`network`	Network parameters and devices parameters
`revision`	Revision identifier parameters

Note: The default parameter file contains sections that are I/O-specific, and will therefore 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` section.
- For IOS-E based systems, remove the `gigaring` section.

The following subsections describe the parameter file sections; sample sections are included.

5.11.1 `gigaring` section

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.

The `gigaring` section consists of two subsections:

- `gr_route`
- `gr_union`

5.11.1.1 The 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 6 {
                        channel 024;
                }
                ring 7 {
                        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 octal integers in the range 1 through 127. Valid node numbers are octal integers in the range 1 through 63 . The following channel numbers are valid for CRAY J90se systems:

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

5.11.1.2 The gr_union subsection

A *GigaRing union* is a logical representation of a device that has more than one ring and node designation.

By convention, ring 128 (0200) is reserved for GigaRing union devices. There is a maximum of 16 nodes (node numbers through 15) reserved for GigaRing union devices. Devices that are configured as GigaRing union devices allow their device drivers to query the low-level master direct memory access (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 128, node 1 {
                        ring 6, node 4;
                        ring 6, node 5;
                        ring 7, node 4;
                        ring 7, node 5;
                }
        }
}
```

A GigaRing union logical device designated as ring 128, node 1, will be created and will consist of four physical destinations.

5.11.2 `mainframe` section

The `mainframe` section defines the following hardware parameters:

- Number of CPUs
- Number of mainframe cluster registers
- Size of the mainframe memory
- Channel information:
  - Physical channel for a GigaRing environment

For GigaRing based systems, this menu selection will not appear but the parameters will be present in the `/CONFIGURATION` file.

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

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

5.11.2.1 Number of CPUs

5.11.2.2 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 *cpus* + 1.

5.11.2.3 Size of memory

The mainframe memory size is specified by the following statement:

```
value units memory ;
```

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

5.11.2.3.1 Channel declarations

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. This assignment appears as follows:

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

ordinal should begin with 0 and should be increased by one for each additional interface of the same type. These channel parameters should appear in numerical order in the parameter file (for example, channel 1 should follow channel 0.) *ring_number* must be an integer in the range 1 through 127. *node_number* must be an integer in the range 1 through 63.

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.

5.11.2.4 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 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;
}
```

5.11.3 `unicos` section

The `unicos` section sets certain tuneable 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 tuneable parameters } ;
```

A UNICOS tuneable parameter is specified by the following statement:

```
value parameter ;
```

All systems have the same tuneable parameters for online tapes, table sizes, and maximum limits. Table 5 through Table 7 show these parameters.

Table 5. 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 6. 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 7. Maximum limits parameters

Parameter	Description
GUESTMAX	Maximum number of guest systems.
NBUF	Number of buffer headers.
NGRT	Maximum number of guest resource table entires.

Table 8 and Table 9 show the parameters for a GigaRing based system.

Table 8. Disk parameters (common)

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

Table 9. Disk parameters (GigaRing based systems only)

Parameter	Description
XDDMAX	Maximum number of physical devices. The default is 32. (GigaRing environment only.)
XDDSLMAX	Maximum number of physical slices. The default is 256. (GigaRing environment only.)

Table 10 shows the valid unit bit and range numbers for IONs.

Table 10. ION unit bit and range numbers

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

5.11.3.1 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 */
}
```

5.11.4 `filesystem` section

The `filesystem` section includes the following:

- Description of the physical devices in the system:
  - xdisks, disks, and RAM for GigaRing based systems
- Description of the device nodes in the system: XDD, QDD, PDD, RDD, MDD, HDD, and SDD for GigaRing based systems

- Identification of the root, swap, SDS, and dump devices
- Description of the GigaRing based systems

Set these parameters by using the following menu selection:

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

For information on striping file systems, see *General UNICOS System Administration*, Cray Research publication SG–2301.

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

```
filesystem {
```

The following subsections describe each portion of the `filesystem` section of the parameter file. For more detailed information on physical device specification, see *General UNICOS System Administration*, Cray Research publication SG–2301.

Set these parameters by using the following menu selection:

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

5.11.4.1 Physical device definition for GigaRing based systems

The following types of physical devices are available for CRAY J90seGigaRing based systems:

- Random access memory (RAM)
- Physical storage devices

Table 11 summarizes disk information for GigaRing based systems.

Table 11. 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	SPN	`xdd`	`/dev/xdd`	`dev_xdd`	`xdisk`	`NO`

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

```
RAM ram_name     { length length_number units ;     slice specification
  }
```

where:

ram_name — Name of the RAM, which must be unique among all devices.

length_number — Size of the RAM, specified in units of `blocks`, `tracks`, `cylinders`, or `sectors`.

slice specification — Specification of the slice.

The physical storage device definition has the following format in a GigaRing based system for IPI-2 disks:

```
disk device_name {   type type;   iopath {         ring
   ring_integer;       node node_integer;        channel
   channel_integer;       }   unit disk_unit_number;    pdd
   pdd_slice_name {       minor minor_number;         unit
   pdd_unit_number;        length length_in_units;        }   }
```

where:

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.
pdd_unit_number	Number of the slice.
length_in_units	Length of the slice in units, where *unit* may be `block`, `track`, `cylinder`, or `sector`).

The xdisk definition applies only to SCSI and Fibre Channel disks on GigaRing based systems. It has the following format:

```
xdisk xdisk_name {
     iounit number;
     iopath {
          ring ring_integer;
          node node_integer;
          channel channel_integer;
     }
     xdd xdd_slice_name {
          minor minor_number;
          sector sector_number;
          length number_of_sectors;
     }
}
```

where:

xdisk_name	Name of the physical storage device, which must be unique among all devices.
iounit_number	I/O unit number.
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. For MPNs, this is the physical SBUS slot 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.
sector_number	Number of the sector.
length_in_sectors	Length of the slice in sectors.

5.11.4.2 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*, Cray Research publication SG–2301.

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

```
Configure System
    ->Disk Configuration
        ->Physical Device Slices
```

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.

5.11.5 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 (these must be LDD definitions):

```
rootdev is ldd name swapdev is ldd name;
```

Set these parameters by using the following menu selection:

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

5.11.6 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:

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

5.11.7 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 5.11.3, page 53.

```
filesystem {
    /* Physical device configuration */
    xdisk d04026.3 {
        iounit 1;
        iopath { ring 4; node 2; channel 6; }
        unit   3;
        xdd 04026.3_usr_i { minor 231; sector 0;      length 444864 sectors; }
        xdd 04026.3_ccn   { minor 232; sector 444864; length 222432 sectors; }
    }
    xdisk d03020.0 {
        iounit 1;
```

```
        iopath { ring 3; node 2; 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; }
}
    /* 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     ; }

    rootdev is ldd root_e;
    swapdev is ldd swap;

}
```

5.11.8 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
- 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 {       integer network_parameter_statement;
   physical_network_device_statement; }
```

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

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

5.11.8.1 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:

integer network_parameter_statement;

The *network_parameter_statement* argument can have the following values:

Table 12. 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. Should be a power of 2.
`atmarp_send`	Amount of socket space used for `send` for ATM ARP traffic. Should be a power of 2.
`cnfs_static_clients`	Maximum number of active Cray NFS static clients.
`cnfs_temp_clients`	Maximum number of active CNFS temporary clients.

Parameters	Description
`hidirmode`	Sets permissions or file mode for `/dev/hippi` directories.
`hifilemode`	Sets permissions or file mode for `/dev/hippi/*` files.
`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).
`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.
`tcp_numbspace`	Number of clicks of memory set aside for TCP/IP MBUFs (TCP/IP managed memory buffers).

Table 13. Network parameter values (GigaRing-based systems)

Parameters	Description
`maxinputs`	Maximum number of asynchronous read 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 64.
`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 64.
`maxusers`	Maximum number of applications that can share the HIPPI device. This parameter only applies to HIPPI devices. For HIPPI devices, the value must be an integer in the range 1 through 8. The default is 2.

For more information about ATM, see the *Asynchronous Transfer Mode (ATM) Administrator's Guide*, Cray Research publication SG–2193.

5.11.8.2 Network devices

The network device statement, which describes specific devices, consists of the `iopath` statement, which has the following syntax:

```
iopath {
       ring ring_integer;
       node node_integer;
       channel channel_integer;
 }
```

Where:

ring_integer The number of the I/O path ring.

node_integer Number of the I/O path node.

channel_integer	Number of the I/O path channel. For HPNs, this is the hardware channel (`0` or `1`).

5.11.8.3 Device types

The following tables describe the types of devices for all systems and for GigaRing based systems.

Table 14. Network device types (common)

Device type	Description
`hidev`	High-speed HIPPI device.
`npdev`	Low-speed channel; if this device is specified, then the *integer* argument is the ordinal of the network device.

Table 15. Network device types (GigaRing based systems only)

Device type	Description
`gfddi`	GigaRing FDDI device.
`gatm`	GigaRing asynchronous transfer mode (ATM) device.
`gether`	GigaRing Ethernet device.
`ghippi`	GigaRing HIPPI device.

5.11.8.4 Device formats for GigaRing based systems

The following formats apply to GigaRing based systems:

```
gatm 0 {
     iopath {iopath_information}
     maxusers integer;
     maxinputs integer;
     maxoutputs integer;
}
gether 0 {
     iopath {iopath_information}
     maxusers integer;
     maxinputs integer;
```

```
     maxoutputs integer;
}
gfddi 0 {
     iopath {iopath_information}
     maxusers integer;
     maxinputs integer;
     maxoutputs integer;
}
ghippi 0 {
     iopath {iopath_information}
     maxusers integer;
     maxinputs integer;
     maxoutputs integer;
     ithreshold integer;
     othreshold integer;
}
```

5.11.8.5 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 1; node 2; channel 5; }
    }
    gfddi 0 {
        iopath { ring 1; node 2; channel 0; }
    }
    gatm 0 {
        iopath { ring 1; node 3; channel 1; }
    }
    ghippi 0 {
        iopath { ring 1; node 4; channel 0; }
        ithreshold 2;
        othreshold 2;
    }
}
```

5.11.9 `revision` section

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

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.

5.12 Checking your disk configuration parameter file

To verify configurations, use either the menu system or the `/etc/econfig` command. If you are using the menu system, you can verify configurations by selecting the `Configure System ==> Disk Configuration ==> Verify the disk configuration ...` menu option.

To verify the configuration manually, perform the following procedure.

Procedure 3: Verifying your disk configuration file

1. Check the syntax of CSL by using the following `/etc/econfig` command:

   ```
   # /etc/econfig  your_param_file_name
   ```

The `/etc/econfig` program accepts only valid CSL statements as input. If you use the `/etc/econfig` command, you should use it before booting a new configuration to prevent receiving errors during CSL processing.

2. To generate the `mknod` commands from your parameter file, use the following syntax:

   ```
   # /etc/econfig -d your_param_file_name > /dev/mkdev.sh
   ```

3. Remove the existing devices by using the following commands:

   ```
   # cd /dev
   # rm dsk/* pdd/* mdd/* sdd/* ldd/* xdd/*
   ```

4. Generate the new device definitions by using the following commands:

   ```
   # chmod 755 /dev/mkdev.sh
   # cd /dev
   # ./mkdev.sh
   ```

A sample CRAY J90se `/opt/CYRIos/`*sn9xxx*`/param` configuration file follows.

```
/*
 *
 * Configuration parameter file
 *
 */

revision "sn9703";

dumpinfo {
     memory range is 0 to 12 Mwords
}


/*
 * Update information for "mainframe" section:
 *
 * HARDWARE INFORMATION:
 */
mainframe {
 /*
  * BEGIN SECTION: HARDWARE INFORMATION
  */
 4 cpus;
 256 Mwords memory;
 channel 024 is gigaring to ring 3, node 0;
 channel 0112 is lowspeed to pseudo TCP;
}

gigaring {
    gr_route {
         ring 3 {
              channel 024;
       }
    }
}

/*
 *  UNICOS configuration
 */
unicos {
 /*
  * BEGIN SECTION: KERNEL PARAMETERS
  */
```

```
 17550 LDCHCORE;
 49140 NLDCH;
 4096  NBUF;  /* system buffers */
 850   PDDSLMAX;  /* maximum number of physical slices */
 850   LDDMAX;  /* maximum number of logical devices */
 850   PDDMAX;  /* maximum number of physical devices */
 230   XDDMAX;
 200   XDDSLMAX;
 80    SDDSLMAX;
 12    TAPE_MAX_CONF_UP;
 65536 TAPE_MAX_PER_DEV;
 1     io_connect;
 /*
  * END SECTION: KERNEL PARAMETERS
  */
}

/*
 * Update information for "filesystem" section:
 *
 * DISK CONFIGURATION:
 *
 * SPECIAL SYSTEM DEVICES:
 */
filesystem {
 /*
  * BEGIN SECTION: DISK CONFIGURATION
  */
/*
 * Physical device configuration
 */
 xdisk "03026.0" {iounit 1; iopath {ring 3; node 2; channel 6;} unit 0;
  xdd roota {minor 1; sector 0; length 250000 sectors;}
  xdd usra {minor 2; sector 250000; length 250000 sectors;}
  xdd srca {minor 3; sector 500000; length 500000 sectors;}
  xdd disk0 {minor 4; sector 1000000; length 1342634 sectors;}
 }
 xdisk "03026.1" {iounit 1; iopath {ring 3; node 2; channel 6;} unit 1;
  xdd swap {minor 5; sector 0; length 500000 sectors;}
  xdd disk1 {minor 6; sector 500000; length 1842634 sectors;}
 }
  xdisk "03027.0" {iounit 1; iopath {ring 3; node 2; channel 7; } unit 0;
  xdd core {minor 7; sector 0; length 400000 sectors;}
```

```
 xdd disk1 {minor 8; sector 400000; length 1942634 sectors;}
}
xdisk "03027.1" {iounit 1; iopath {ring 3; node 2; channel 7; } unit 1;
 xdd tmp {minor 9; sector 0; length 1000000 sectors;}
   xdd opt {minor 10; sector 1000000; length 500000 sectors; }
   xdd disk2 {minor 11; sector 1500000; length 842634 sectors; }
}
* Logical device configuration
*/
ldd swap { minor 1;
 xdd swap;
}
ldd core { minor 2;
 xdd core;
}
ldd tmp  { minor 3;
 xdd tmp;
}
ldd roota  {minor 4;
 xdd roota;
}
ldd usra  {minor 5;
 xdd usra;
}
ldd srca  {minor 6;
 xdd srca;
}
ldd opt  {minor 7;
 xdd opt;
}
ldd disk0  {minor 8;
 xdd disk0;
}
ldd disk1  {minor 9;
 xdd disk1;
}
ldd disk2  {minor 10;
 xdd disk2;
}

/*
 * END SECTION: DISK CONFIGURATION
 */
```

```
 rootdev is ldd roota;
 swapdev is ldd swap;
 /*
  * END SECTION: SPECIAL SYSTEM DEVICES
  */
}
/*
 *  Network configuration
 */
network {
 4 himaxdevs;
 8 himaxpaths;
 0700 hidirmode;
 0600 hifilemode;
 gfddi 0 {
  iopath {
    ring 3;
    node 2;
    channel 0;
  }
 }
 gether 0 {
  iopath {
    ring 3;
    node 2;
    channel 2;
  }
 }
}
```

Procedure 4: Identifying devices defined on your system and their file system allocation

Note: To complete this procedure, you must be super user; you will see the `sn9` *xxx* # prompt.

To identify the devices provided on your system and their file system allocation, either use the menu system or execute commands.

If you are using the menu system, complete the following steps:

1. Enter the menu system:

Note: To eliminate the need to change to the `/etc/install` directory to enter the menu system, you can include `/etc/install` in your `PATH` statement in your `.profile` or `.cshrc` file.

```
sn9xxx# cd /etc/install
sn9xxx# ./install
```

2. Select the following menu:

```
UNICOS Installation / Configuration Menu System
Configure system
Disk configuration
```

3. Determine which devices and file systems are configured on your system by viewing the submenus.

   Section 5.11.4, page 56, describes the sections of the `/opt/CYRIos/`*sn9xxx*`/param` file.

   A sample menu screen follows:

```
                        Disk Configuration

M-> Physical devices ==>
    Physical device slices ==>
    Logical devices (/dev/dsk entries) ==>
    Mirrored devices (/dev/mdd entries) ==>
    Striped devices (/dev/sdd entries) ==>
    Logical device cache ==>
    Verify the disk configuration ...
    Review the disk configuration verification ..
    Dry run the disk configuration ...
    Review the disk configuration dry run ...
    Update disk device nodes on activation?
    Import the disk configuration ...
    Activate the disk configuration ...
```

If you are not using the menu system, use the following commands to display information that you can use to identify the devices on your system and their file system allocation:

1. The `/etc/qddstat` command displays the name of the device, its type, and whether it is up or down.
2. The `/etc/ddstat /dev/dsk/*` command displays all disk devices and their file system allocation (or you can execute the command for individual devices). Logical devices are divided into their individual components and presented in a disk-specific format. The output is not formatted (headings are not provided), but the output provides comprehensive information. The following is an example of `ddstat` output from a CRAY J90se system. The fields are defined in the diagram that follows:

```
$ ddstat /dev/dsk/tmp

/dev/dsk/tmp b 34/69 0 0 /dev/ldd/tmp
       /dev/qdd/tmp_1 c dev_qdd/69 12 01036020 0 201600 00 0 0 0
       /dev/qdd/tmp_2 c dev_qdd/96 12 01036020 0 201600 00 0 0 1
       /dev/qdd/tmp_3 c dev_qdd/97 12 01036020 0 201600 00 0 0 2
```

Figure 2. ddstat output field definitions

Procedure 5: Modifying your configuration file

Note: To do this procedure, you must be super user; you will see the `sn9` *xxx* # prompt.

To modify your configuration, either use the menu system or edit the parameter file.

If you are using the menu system to modify your configuration file, follow the procedure on the "Identifying devices defined on your system and their file system allocation" procedure. Then import the disk configuration, modify the menus as needed, and then activate your new configuration (`Activate the disk configuration ...` line of the `Disk Configuration` menu).

If you are not using the menu system, complete the following steps.

1. Create a backup copy of any file system that will be changed in your revised configuration file (`/opt/CYRIos/`*sn9xxx*`/param`) by using the `dump` command. See section Chapter 6, page 105.

2. Create a backup copy of your current configuration file:

```
sn9xxxx# cd /etc/config
sn9xxxx# cp param old.param
sn9xxxx#
```

3. Make sure that you are in `/etc/config` on UNICOS. Copy the configuration file `/opt/CYRIos/`*sn9xxx*`/param` from the SWS disk drive to a UNICOS disk and a file name of your choice (`new.param` in the following example) by using the `/bin/rcp` command so that you can edit it. The following command specifies that the `/opt/CYRIos/`*sn9xxx*`/param` file will be read from the SWS system disk and be named `new.param`:

```
sn9xxx# rcp SWS:/opt/CYRIos/sn9xxx/param  new.param
```

4. Edit your copy of the parameter file on UNICOS (see Section 5.11.4, page 56).

```
sn9xxx# vi new.param
```

5. Check for syntax errors by using the `/etc/econfig` command:

```
sn9xxx# /etc/econfig new.param
```

6. When you are done making your configuration changes, copy your new version of the system configuration (`new.param`) on top of the old original version of the system configuration (`/opt/CYRIos/`*sn9xxx*`/param` on the SWS disk), using the `/bin/rcp`) command. The following command specifies that the `new.param` file will be written to the SWS system disk and be named `/opt/CYRIos/`*sn9xxx*`/param`:

Caution: If you use the `rcp` command as shown in the following example, the file will be overwritten; before doing this, be sure that a back-up copy of your current configuration file exists.

```
sn9xxx# rcp new.param SWS:/opt/CYRIos/sn9xxx/param
```

At this point, the next time UNICOS is booted, it will come up with the new system configuration that you specified, and the system will copy the SWS `/opt/CYRIos/`*sn9xxx*`/param` file to the UNICOS `/CONFIGURATION` file.

7. After the system is booted to single-user mode, you must make, label, check, and mount any file system (old or new) that differs in any way from the way it was previously defined in the original version of the SWS `/opt/CYRIos/`*sn9xxx*`/param` file you changed. (Section 5.15, page 90 describes these additional steps.) You then must restore altered file systems from the back-up tapes you created in step 1 by using the `restore` command.

8. You must create the new mknod information for any new disk devices you have defined. To do this, use the `econfig` command to create a file containing the `mknod` information:

```
sn9xxx# econfig -d new_param_file > /dev/mkdev
```

Remove the existing devices:

```
sn9xxx# rm dsk/* pdd/* ldd/* sdd/* mdd/* xdd/*
```

Generate the new device definitions:

```
sn9xxx# cd /dev
sn9xxx# chmod 755 mkdev
sn9xxx# ./mkdev
```

5.13 File system quotas

The file system quota system allows you to control the amount of file system space in blocks and numbers of files used by each account, group, and user on an individual basis. Control may be applied to some or all of the configured file systems, except for the root file system. Attempts to exceed these limits result in an error similar to the error that occurs if the file system is out of free space.

5.13.1 File system quota overview

File system quotas are implemented to control the amount of file system space consumed by users and are characterized as follows:

- You can set quotas for three different ID classes:
  - User ID (`uid`)
  - Group ID (`gid`)
  - Account ID (`acid`)
- You can set up two types of quotas, file and inode:
  - *File quotas* determine the amount of space an ID may consume in blocks (512 words=4096 bytes).
  - *Inode quotas* determine the number files an ID can create.
- You can apply controls to some or all of the configured file systems (except the `root` file system).
- You can create adjustable warning windows to inform the user when usage gets close to a quota.

5.13.2 Quota control structure

The quota control file, `.Quota60`, which by default resides on the file system it controls, contains all the information the quota system needs. A header in the quota control file contains the default information for IDs, such as default file and inode quotas, default warning window, and warning fractions. The default

values are taken from a header file, `/usr/include/sys/quota.h`, and may be modified through the administrative command, `quadmin`(8).

Every ID number (user, group, and account) up to `MAXUID` has an offset into the quota control file. At that offset, control information exists for each ID class. Each of the following fields exists for each ID in the quota control file:

Field	Contents		
`Flags`	Only one flag is defined, which indicates that the entry has been preset by quadmin rather than the kernel.		
`File quotas` `Inode quotas`	Maximum number of file blocks or inodes allowed by the ID. The following special values are stored in this field:		
	#	Keyword	Description
	0		No value specified.
	2	default	Use the default file/inode quota that appears in the control file hearder.
	3	infinite	Infinite quota (no quota evaluatin is done).
	4	prevent	No blocks/inodes may be allocated by this ID.
`File warning window`	Number of blocks below the maximum number of file blocks when a warning should be issued.		
`Inode warning window`	Number of inodes below the maximum number of inodes when a warning should be issued.		
`File usage`	Current number of blocks used by the ID.		
`Inode usage`	Current number of inodes used by the ID.		
`Quota hit`	Time when the quota is reached.		

5.13.3 Commands

The following commands are used to administer file system quotas:

- `qudu`(8): Reports file system quota usage information
- `quadmin`(8): Administers file quotas
- `mount`(8): Mounts a file system (options for specifying quota control file)

- `quota`(1): Displays quota information

5.13.4 Quotas and the user

Every file system user on the Cray Research system can be controlled by a quota. As file system space is consumed, the user ID, group ID, and account ID, sorted in the file's inode, accumulate the file system usage information. When a user exceeds a quota, an error occurs that is similar to when the file system is out of free space. A `SIGINFO` signal is also sent when a warning is reached or a quota is exceeded. This signal, which is ignored by default, can be caught and interpreted by using a `getinfo`(2) request.

When data migration is turned on and a file is migrated, the space the file occupied is credited to the file owner's ID. When a file is brought back online, the number of blocks is added to the ID's file quota. If bringing a file back would violate an enforced quota limit, that file cannot be brought online.

If a new user is added to the system, the UNICOS kernel automatically creates a quota control entry with default values (taken from the header information in the `.Quota60` file) for any IDs that are not already defined in the quota control files.

If you need to adjust these values for that specific ID, run the `quadmin` command to set up the correct quota information for the ID on each file system.

5.13.5 Quota header file

The header file, `quota.h`, contains global information for file system quotas. It is recommended that you do not change the values in the header file. Use `quadmin` to adjust the value to better suite the needs of your site.

The following is an excerpt from a `quota.h` file:

```
# cat /usr/include/sys/quota.h...
#define  QFV_AFQ          5000     /* account default quota */
#define  QFV_AIQ          200      /* account default inodes */
#define  QFV_GFQ          5000     /* group default quota */
#define  QFV_GIQ          200      /* group default inodes */
#define  QFV_UFQ          5000     /* user default quota */
#define  QFV_UIQ          200      /* user default inodes */
#define  QFV_WARNAFQ      0.9      /* account file warning default */
#define  QFV_WARNAIQ      0.9      /* account inode warning default */
#define  QFV_WARNGFQ      0.9      /* group file warning default */
#define  QFV_WARNGIQ      0.9      /* group inode warning default */
#define  QFV_WARNUFQ      0.9      /* user file warning default */
#define  QFV_WARNUIQ      0.9      /* user inode warning default */
```

5.13.5.1 Soft quotas

Soft quotas is a mode of operation, also called oversubscription, that allows a user to exceed quotas by a controlled number of blocks for a limited period of time. It is selected by setting the algorithm selector in a header field. For more information about setting up oversubscription, see *General UNICOS System Administration*, Cray Research publication SG–2301.

Procedure 6: Setting up a quota control file

When your site decides to turn on the quota system, you must complete the following steps to ensure that a quota file exists:

1. To ensure that your system has a kernel built with the quota system turned on, look at the following UNICOS Installation / Configuration menu item:

   ```
   Configure System
   Major Software Configuration
   S-> File Quotas on
   ```

   Note: If you are implementing quotas on a newly created file system, skip step 2 and go on to step 3.

2. To implement quotas on an existing file system, collect current usage information for all user, group, and account IDs by using the `qudu` command. The output for `qudu` contains directives for the `quadmin` command, which will create or update the quota control file. Either redirect the output to a file, or pipe the output directory to `quadmin`.

```
# umount /dev/dsk/usa
# qudu /dev/dsk/usa > qudu.out
# cut -d' ' -f1-5 qudu.out| sort +0 -1 +4nr
```

(View IDs according to classes (uid, gid, and acid) with each class sorted so that the ID with the greatest inode usage is printed first.)

```
# cut -d' ' -f1,2,6-8 qudu.out| sort +0 -1 +4nr
```

(View IDs according to classes (uid, gid, and acid) with each class sorted so that the ID with the greatest file usage is printed first.)

3. Modify any quota entries in the `quadmin` source file (you can use any editor on the source file). All IDs take on the default values for file and inode quota limits unless they are updated by using the `quadmin` command. You may want to view what the current level of usage is for the file system (the `sort` and `cut` commands may be useful to accomplish this task). If you find that several IDs are already over the quota, you may want to consider raising the quota of those IDs or the overall default quota.

   Note: Root and daemon IDs should not be under quota control. Put quota values of `infinite` in these ID fields. Setting values lower than 10 will result in a value of 10.

```
# vi qudu.out
```

(Append the following information:)

```
   enable uid 0-100
   user * file quota infinite
   user * inode quota infinite
   enable gid 0-100
   group * file quota infinite
   group * inode quota infinite
   enable acid 0-100
   account * file quota infinite
   account * inode quota infinite
   user sue file quota 35000
   user sue inode quota 500

# fsck /dev/dsk/usa
# mount /dev/dsk/usa /usa
```

4. Create the quota control file, `.Quota60`, for the file system. A quota control file is associated with a file system at the time the file system is mounted. Quotas are enforced when the file system is mounted with the `-q` or `-Q` option.

```
# quadmin -F -m qudu.out
# umount /dev/dsk/usa
# mount -Q /usa/.Quota60 /dev/dsk/usa /usa
```

5. Establish ownership of the quota file to be `root` and specify that others cannot access, modify, or delete the file.

```
# chown root /usa/.Quota60
```

6. If you mounted your file system using the `mount -q` or `-Q` option, you can activate file system quotas from one of the site-modified startup scripts (either `/etc/rc.mid` or `/etc/rc/pst`).

7. If you mounted your file system without specifying the `-q` or `-Q` options, you can activate quotas by using the `quadmin` command. The `quadmin` command has the following three activation levels, which can be changed at any time:

   - *count* maintains counts for the quota system, but does not send a warning or quota limit signal and does not enforce quotas.
   - *inform* maintains counts and informs users with a warning or quota limit signal, but does not enforce quotas.
   - *enforce* maintains counts, issues warning and quota limit signals, and enforces quotas.

```
# quadmin -c enforce -s /usa
```

Note: Once quota control is activated, you can change its enforcement mode, but you cannot deactivate it. You must unmount the file system to deactivate quota controls.

5.13.6 Current usage information

When the `/etc/fsck` or `/etc/gencat` utilities find file system errors and try to correct the problem, they may remove an inode or modify information about a file.

Because these commands do not update the quota control file with current inode or file usage, you should run `/etc/qudu` after running `fsck` or `gencat`. Then run `quadmin` immediately after the device is mounted.

```
# fsck -u /dev/dsk/usa
# qudu /dev/dsk/usa > /qudu.out
# mount -q /dev/dsk/usa
# quadmin /qudu.out
```

5.13.7 Warning windows

As administrator, you are responsible for setting the warning window value. This is initially set through parameters in the `quota.h` file and can be adjusted by using `quadmin` directives.

Warning windows can be represented as fractions or absolute window values. A warning fraction, f, must be in the range of $0.0 < f < 1.0$. A number of 10 or greater is considered an absolute window value. A number ranging from 1 through 9 is interpreted as zero, meaning that there is no warning window and no warning will ever occur. An absolute window value is interpreted as the total number of blocks or inodes below the file or inode quota.

The following example causes a warning message to appear when a user has used up 90% of the allowed file quota or inode quota:

```
# quadmin -m infile1
# cat infile1
filesystem dsk/usa ; open dsk/usa
default acid file warning 0.9
default uid file warning 0.9
default gid file warning 0.9
default acid i-node warning 0.9
default uid i-node warning 0.9
default gid i-node warning 0.9
```

5.13.8 Sharing quota controls files between multiple file systems

You may have a single quota control file manage more than one file system. To set up and activate a shared quota control file, you should observe the following guidelines:

- When accumulating current usage information, you must run separate `qudu` commands for each file system. Then you must sum up the usage of all IDs involved in these file systems that are going to share the control file. There is currently no command to accomplish this task.
- You must ensure that the file system on which the quota control file resides is mounted before quotas can be activated on another file system that will share this quota control file.
- When the mount command is executed for the file systems that will share this quota control file, you must specify the `-Q` option.

Observe the following disadvantages of a shared control file:

- If there is too much quota control traffic, the impact on performance is uncertain.
- If a file system containing a quota control file is destroyed, quota control is lost on the file system that was sharing that quota control file.

5.13.9 Monitoring quotas

User warning and limit messages are automatically written to the standard error file, `stderr`, by the Korn, POSIX, and C login shells.

You can examine quotas by using the `quota` command. If you want to check all of a user's authorized account IDs and group IDs, enter the following command:

```
# quota -A -G -r wl

File system: /cyclone
 User: john, Id: 1846
             File blocks    Inodes
User Quota:   4000* ( 2.1%)  4000* ( 0.5%)
 Warning:     3600* ( 2.3%)  3600* ( 0.6%)
   Usage:     84             20
```

5.14 Planning file system change

You may reconfigure your file systems occasionally, usually to allow for growth in your file systems, to add new disks, and to meet new operational requirements. A well-thought-out and well-defined plan that is generated in advance can help smooth out this process.

5.14.1 Configuration objectives

To determine configuration objectives, understand your current configuration and determine your objectives for the final disk layout. Familiarize yourself with the form and syntax of the configuration specific language (CSL) that is used to describe file system layouts. Compare the output of the `ddstat /dev/dsk/*` command with the disk layout `param` file.

5.14.2 Plan preparation

To prepare a plan, separate the process of change into incremental steps, stating the objectives for each step. Do not try to make too many changes in one step, and try to combine changes that complement each other into one step.

If you can unmount a file system safely, you can change it in multiuser mode. While in multiuser mode, do **not** try to change the following:

- `root`, `usr`, `home`, `spool`, `tmp`, and `adm` file systems
- Any file system that may become active because of DMF, NQS, NFS, `cron`, MLS, or other activity
- Swap device area

You can change any file system in single-user mode except `root` and the swap device area, which require `param` file adjustments and a system reboot.

For each step, prepare a plan that details the following items:

- List each disk that changes.
- List each file system destroyed, created, or changed.
- For each file system destroyed:
  - If the data will be saved, verify that the contents from this file system were saved earlier in the plan.
  - Delete redundant `/dev` entries.

- For each file system created:
  - Format the file system by using the `mkfs`(8), `labelit`(8), and `fsck`(8) commands.
  - Populate (restore data to) the file system with the saved data (if any).
- For each file system changed:
  - Check that the data from the file system was saved earlier in the plan.
  - Format the file system by using the `mkfs`, `labelit`, and `fsck` commands.
  - Populate the file system with the saved data.

To ensure that any data required for the next stage is preserved, check your plan. Review your plan with a colleague or Cray Research service representative.

5.14.3 New disks

To bring a new disk online, add the disk to the disk `param` file and then reboot your system.

Your hardware installer will advise you where new disks have been attached to your system by providing channel, device, and unit numbers. Flaw tables, if applicable, will be initialized, but Redundant Arrays of Independent Disks (RAID) devices may require special initialization procedures.

5.14.4 Implementation

To implement the plan, follow these steps:

1. Back up all your data to tape. Verify the saved data (make sure that you verify the backup tapes).
2. Save a copy of the original production disk layout `param` files on the IOS.
3. Check for syntax and slice gaps or overlap by checking the `param` files by using the `econfig`(8) command. This can be done on UNICOS in single-user mode.
4. Allow ample time for the change; costly mistakes are more often made when working under pressure. To determine the amount of time you need, multiply the time it takes to shut down and reboot your system by the

number of restarts in your plan. Then add the time needed to backup and restore the data to be moved.

5.14.5 Apply changes

You can use either of the following methods to apply the changes to the new disk layout `param` file.

Method 1:

1. Unmount the file systems that will change.
2. Load the changes into the UNICOS Installation / Configuration Menu system (the installation tool); that is, change variables and parameters in the installation tool to reflect the new, desired file system configuration.
3. Activate the changes.

Method 2:

1. Unmount the file systems that will change.
2. Generate a new `param` file (copy and modify `/etc/config/param` or `/CONFIGURATION`).
3. Generate the `mknod` commands for your new file system configuration by running the `econfig -d` command.
4. For the file systems that change, run the `mknod` commands generated by the `econfig -d` command.

5.14.6 As you proceed

Perform the following steps as you proceed with your file system change plan:

1. Check off items on your detailed plan as you proceed, noting any diversions.
2. Before you format a file system, carefully verify its placement by using the `dmap`(8) command.
3. Verify that each newly completed file system contains what you expect it to contain.
4. Optimize file system usage by applying appropriate `mkfs`(8) options.

5.14.7 Helpful hints for implementing plan

The following information may be helpful as you implement your plan.

- To copy entire file systems, use the `dump`(8) and `restore`(8) commands; to make partial copies, use `find`(1) and `cpio`(1), or `tar`(1).
- Checkpointed jobs will not continue if the inode numbers of files used by that job change or the minor device number of the holding file system changes; any file system reconfiguration will cause restart failures for checkpointed jobs that have open files on any affected file system.
- The `dump` and `restore` commands change the inode numbers and defragment a file system. You can use the `dd`(1) command only between file systems of the same size and type.
- Moving or reordering slices or adding or removing striping or mirroring changes a file system.
- If you are running in single-user mode, the swap device must exist, but it does not have to be full production size.
- Because the swap device definition comes from the `param` file and not its `/dev` entry, you can move it arbitrarily across system reboots (that is, it needs no preparation before use).
- You can prepare and use the dump device area as a temporary file system; however, be sure that you reinitialize it after you have finished your file system changes.
- If you plan to destroy the `/tmp` file system, notify your users (users like to be warned about changes to the `/tmp` file system).
- When you change a file system that is subject to data migration, you must perform special steps. If you are unsure what is included in these steps, contact your Cray Research service representative.
- Although disk slice names do not have to include the disk number, a logical, ordered naming convention can be useful.
- To tag your data, place a file called `1.am.` *fsname* at the head of each file system (*fsname* represents the name of the file system).
- Do not reuse minor device numbers but keep the highest minor device number under the *type* `MAX` limit for your kernel (in which *type* represents the device type).

- You can use striping only on disk devices that have the same physical type; striping must be between slices of the same size and position on different disks.
- To speed data population, apply logical device cache (`ldcache`) to a destination file system.

5.15 Creating file systems

After you have planned the configuration of your physical and logical devices and defined them using CSL, you must follow the steps described in this subsection to create file systems on your logical devices.

1. Build the file system by using the `/etc/mkfs` command.
2. (Optional) Label the file system by using the `/etc/labelit` command.
3. Check the file system structure integrity by using the `/etc/fsck` command.
4. If it does not already exist, create the mount point directory, using the `/etc/mkdir` command.
5. Mount the directory by using the `/etc/mount` command.

The remainder of this section describes the following:

- `/etc/mnttab` and `/etc/fstab` files
- Configuring a file system to be mounted automatically at the initialization of multiuser mode
- Unmounting a file system by using `/etc/umount`

Note: *General UNICOS System Administration*, Cray Research publication SG–2301, and *UNICOS Resource Administration*, Cray Research publication SG–2302, include information on other aspects of file system maintenance. For example, *UNICOS Resource Administration*, Cray Research publication SG–2302 how the file system space monitoring capability can improve the usability and reliability of the system. Space monitoring observes the amount of free space on mounted file systems and takes remedial action if warning or critical thresholds are reached. *UNICOS Resource Administration*, Cray Research publication SG–2302 explains how the file system quota enforcement feature (also called *disk quotas*) lets you control the amount of file system space in blocks and the number of files used by each account, group, and user on an individual basis. You may apply controls to some or all of the configured file systems, except for the `root` file system. Attempts to exceed quota limits cause an error similar to the error that occurs if the file system is out of free space. Optional warning levels also are available for informing users when usage gets close to a quota limit.

Procedure 7: Create the file system

1. Building the file system

The `/etc/mkfs` command builds the file system structure in the areas of disk that make up the logical device for a given file system. This structure includes designating areas of the logical device to contain the boot block, super blocks, inode region, and so on. On CRAY J90se systems, you always should use the `-q` option when you run `mkfs` to build a structure, which will prevent the disk surfaces from being verified. (When the UNICOS multilevel security (MLS) feature is enabled, `mkfs` provides the new file system with minimum and maximum security levels and authorized compartments.) The format of the `mkfs` command is as follows:

```
/etc/mkfs [-q] [-n blocks] [-a strategy] [-B bytes] [-A blocks]
  device
```

`-q`	Specifies quick mode; bypasses surface check.
`-n` *blocks*	Specifies number of blocks you want the file system to contain.
`-a` *strategy*	Specifies an allocation strategy. This option can take one of the following values:
	`rrf` Round-robin all files (default)

`rrd1`	Round-robin first-level directories
`rrda`	Round-robin all directories

`-B` *bytes*

Specifies the number of bytes after which a file is considered to be big. The default is 32,768 bytes (8 blocks) and is defined by the `BIGFILE` argument in `/usr/src/uts/sys/param.h`; you cannot change the definition. The default might be the value you want to use at your site.

`-A` *blocks*

Specifies the minimum number of 4-Kbyte blocks allocated for a file whose size is greater than or equal to `BIGFILE` (see the `-B` option). The default is 21 sectors (blocks) and is defined by the `BIGUNIT` argument in `/usr/src/uts/sys/param.h`; you cannot change the definition. The default might be the value you want to use at your site. For DD-60, DA-62, and DA-301 disk drives, for which the sector size is 16 Kbytes, the allocation unit is rounded up to the nearest multiple of four. For DA-60 disk drives, for which the sector size is 64 Kbytes, the allocation unit is rounded up to the nearest multiple of 16.

The interaction of the `-A` and `-B` options is as follows. If a file creation request exceeds the size of `BIGFILE` (8 blocks), the system will allocate `BIGUNIT` (21) more blocks in an attempt to meet the request. The system then checks to see whether the request has been met. If the amount allocated so far is still less than the request, the system will allocate another `BIGUNIT` number of blocks and again check to see whether the request has been met. This cycle of allocation and checking will repeat until the request has been met.

You must determine the best settings for the `-A` and `-B` options for your file systems and average allocation requests at your site.

device

Full path name of the block special file (`/dev/dsk/` *filename*). When the disk configuration is activated at system startup, block

special files are created for each logical device in your configuration. They are placed in the `/dev/dsk` directory and take on the same name as the logical device. You must know the full path name.

A basic example follows:

```
/etc/mkfs -q /dev/dsk/home
```

The following examples show the syntax and explain each of the three possible allocation strategies.

Example 1 uses a "round-robin, first-level" strategy (`rrd1`) to create a file system called `bob`. It tries to place all files, subdirectories, and directories of a file system on the same partition.

Example 1: round-robin, first-level

```
# /etc/mkfs -q -a rrd1 /dev/dsk/bob
```

Example 2 uses a "round-robin, all-directory" strategy (`rrda`) to create a file system named `jane`. Each directory and its files are allocated to the same partition, but each directory is allocated to a different partition than its subdirectories if possible.

Example 2: round-robin, all-directory

```
# /etc/mkfs -q -a rrda /dev/dsk/jane
```

Example 3 uses a "round-robin, all-files" strategy (`rrf`) to create a file system named `jones`. This strategy tries to place all inodes and directories on partition 0 if possible, and it allocates all files for a file system in a "round-robin" fashion. For example, on a three-partition file system, as files `a`, `b`, `c`, `d`, `e`, `f`, and `g` are created, `a` will be placed on partition 0, `b` on partition 1, `c` on partition 2, `d` on partition 0, `e` on partition 1, `f` on partition 2, `g` on partition 0, and so on.

Example 3: round-robin, all-files

```
# /etc/mkfs -q -a rrf /dev/dsk/jones
```

Continue with step 2.

2. Labeling the file system

To create a label on a newly created file system, use the `/etc/labelit` command. This step is optional, but when not done, a warning message is issued when the file system is mounted. The `mount:warning: <` *file-system-name* `> mounted as </` *mount-point-name* `>` message appears when the file system label does not match the mount point directory name. The syntax of `/etc/labelit` is as follows:

```
/etc/labelit device fsname volname
```

device — The name of the logical device that you want to label.

The actual label consists of the following two required fields:

fsname — The name you want to assign to the file system.

volname — The name you want to assign to the volume.

Note: If you do not specify a label, `labelit` displays current label information about a file system; see the following examples.

Example 4: assign file system name and volume name to unmounted file system

The following command assigns a file system name of `usr01` and a volume name of `vol1` to the unmounted file system on `/dev/dsk/usr01`. Notice the new volume and new file system name as specified in the last command response line.

```
# /etc/labelit /dev/dsk/scr_esdi usr01 vol1
Current fsname: scr_esdi, Current volname: E000_scr, Blocks: 487800, Inodes: 121968
Date last mounted: Sun Sep 26 03:06:50 1993
NEW fsname = usr01, NEW volname = vol1
```

Example 5: `labelit` output

If you do not specify a label, `labelit` displays current label information about a file system, as shown in the following example, which specifies only the file system name:

```
# /etc/labelit /dev/dsk/scr_esdi
Current fsname: scr_esdi, Current volname: E000_scr, Blocks: 487800, I-nodes: 121968
Date last mounted: Sun Sep 26 10:52:53 1993
```

Continue with step 3.

3. Checking the file system

   Note: You must check a file system **before** it is mounted; otherwise, the file system will not be mounted. Before mounting a file system, always perform a consistency check on it to ensure that a reliable environment exists for file storage. When the system is brought to multiuser mode, the `/etc/bcheckrc` multiuser level start-up script automatically checks any file systems listed in the `/etc/fstab` file. The `/etc/fstab` file also has an option that can cause its files to be mounted automatically at multiuser start-up time. Because of the multipass nature of the `/etc/fsck` command, the file systems must be in an inactive state while being checked. You must ensure that all file systems to be checked are unmounted.

The `/etc/fsck` command is an interactive file system check and repair program that uses the redundant structural information in the file system to perform several consistency checks. The `fsck` process has six possible phases; a series of error messages may appear during each phase, and you are prompted to answer `YES` or `NO` to a series of questions about the errors encountered. To assess any potential problems, you may want to answer `NO` to all questions, then rerun `fsck` after you have decided on a plan for any needed repairs. If you use the `-n` option with `fsck`, the default answer to all questions is `NO`. For example, if the `/tmp` file system is truly used as a volatile scratch area, you may not want to bother repairing any errors that `fsck` finds, in which case, you may prefer the `-n` option.

When you are prompted to clear the inode, it is sometimes best to answer `NO` first. The `fsck` command also will display the inode number and size; you can make a note of the number, and then, if you do want to clear the inode, you can rerun `fsck` and clear it.

No matter how many error messages you receive from `fsck`, and no matter how serious the errors may seem, you always can reconstruct your file system from the last version of your back-up media. Therefore, it is absolutely critical that you have a consistent method of doing backups and that you always follow that method. If you have the backups, you can always restore your file system from the backups if all else fails.

The `fsck` program always goes through the following five phases. Phase 6 sometimes occurs if an error occurred during phase 5. Generally, each phase is a "clean up" after the previous phase.

Phase	Description
1: Check blocks and sizes	Examines the file system's inode list for duplicate blocks, incorrect block numbers, or incorrect format.
2: Check path names	Removes directory entries that were modified in phase 1.
3: Check connectivity	Checks the connectivity of the file system, verifying that each inode has at least one directory entry and creating error messages for unreferenced directories.
4: Check reference counts	Lists errors from missing or lost directories, incorrect link counts, or unreferenced files.
5: Check free list	Checks the relationship between the number of allocated blocks in the file system, the number of blocks in use, and the difference between the two (the *free block list*). If the current free block count (immediately calculated) is not the same as the free block list, an error is reported.
6: Salvaging	Occurs only if an error occurred in phase 5 and you answered YES to the SALVAGE? prompt.

You must become familiar with using `fsck` and become comfortable replying to the `fsck` error messages.

If a file system was unmounted cleanly, `fsck` responds with the following message and does not perform the file system check:

```
/dev/dsk/usr01: Filesystem check bypassed
```

If an inconsistency is detected, fsck reports this in the same window in which the command was invoked and will ask whether the inconsistency should be fixed or ignored. The `/etc/fsck` command can often repair a corrupted file system.

The `/etc/fsck` command also checks for orphan files (files not connected to the `root` inode of the file system). A scan is done of all unaccounted blocks in the file system. Each block is checked for the inode magic number. If it is found, blocks that are claimed by the inode are checked to see whether they are valid and do not duplicate block numbers. If this step is accomplished safely, a prompt will appear that will ask whether you want the inode to be salvaged, which you probably will want to do.

Example:

```
/etc/fsck /dev/dsk/usr01
```

For a complete description of all parameters, see the `fsck`(8) man page.

> **Note:** A file system can become corrupted in a variety of ways, the most common of which are hardware failures and improper shutdown procedures. If you do not follow proper startup procedures, a corrupted file system will become further corrupted.

A hardware failure can occur because of the following:

- Disk pack error
- Controller failure
- Power failure

An improper system shutdown can occur because of the following:

- Forgetting to `sync` the system prior to halting the CPU
- Physically write protecting a mounted file system
- Taking a mounted file system offline

If you do not use `fsck` to check a file system for inconsistencies, an improper system startup can occur.

The `/etc/fsck` command primarily detects and corrects corruption of the following two types:

- **Improper file creation:** When a user creates a UNICOS file, the system goes through the following four basic steps:

  1. Allocates an inode from the inode region.
  2. Makes a directory entry, and places the new inode number and file name in the directory.
  3. Allocates any data blocks as needed.
  4. Increments the link count in the inode for the file. If this is a directory file, the system also increments the link count for the parent directory.

  If the system cannot complete all four steps successfully, file system errors will occur.

- **Improper file removal:** When a file is removed using the `rm`(1) command, the system proceeds in reverse order, as follows:

  1. Decrements the link count in the inode for the file. If this is a directory file, the system also decrements the link count for the parent directory.
  2. Deallocates the data blocks (if the file's link count is 0).
  3. Removes the directory entry.
  4. Deallocates the inode (if the file's link count is 0).

  If the system cannot complete all four steps successfully, file system errors will occur.

Because a file might be linked to several different directory entries, the inode and data blocks are removed only when the last link is removed.

Continue with step 4.

4. Creating a mount point for the file system

If a mount point does not exist already for a file system, use the `/bin/mkdir` command to create one. Typically, the mount point is given the same name as the logical device name of the file system on which it will be mounted. For example, if a logical device named `/usr/home` has been configured in the `/opt/CYRIos/`*sn9xxx*`/param` file, the mount point also will be named `/usr/home`. You can create this mount point as shown in the following example.

Example:

```
# mkdir /usr/home
```

Note: The contents of the mount point directory are hidden when a file system is mounted on top of it.

Continue with step 5.

1. Mounting the file system

A file system is a sequential array of data until it is mounted. When the file system is mounted, the UNICOS kernel interprets the data as a UNICOS file system that is available as part of the system's complete directory structure. To be accessible to the UNICOS system, all file systems except `root` (`/`) must first be explicitly mounted by using the `mount`(8) command. The file system is mounted on an existing directory. The directory may have to be created, using the `mkdir`(1) command (see step 4). By convention, the name of the directory corresponds to the name of the logical device. The fourth field of the `/etc/fstab` file controls the automatic mounting of user file systems when going to multiuser mode.

The system keeps a table of mounted file systems in memory and writes a copy of the table to `/etc/mnttab`. `root` is always available to the system and is entered into `/etc/mnttab` at boot time through `/etc/brc`. The `root` inode of the mounted file system replaces the mount-point inode in memory; therefore, any files in the mount-point directory are unavailable while the file system is mounted. That is, you should use only an empty directory as a mount point.

Note: You **must** check the file system by using the `fsck` command **before** it is mounted (see step 3).

Example:

```
# /etc/mount /dev/dsk/home /usr/home
```

For a complete description of all options, see the `mount`(8) man page.

Note: Check the permission of the mounted file system. To change the permission of the root directory of the mounted file system, if necessary, use the `chmod` command (see the `chmod`(1) man page).

5.16 /etc/mnttab and /etc/fstab files

The /etc/mnttab and /etc/fstab files are related to the condition of whether a file system is mounted or unmounted.

5.16.1 /etc/mnttab

The /etc/mount and /etc/umount commands maintain the /etc/mnttab file. Two tables keep track of mounted disk devices. The one maintained internally by the UNICOS kernel is always correct. The other, /etc/mnttab, is maintained as a convenience for such scripts as /etc/mount, which, when issued without any arguments, will display the list of all currently mounted file systems.

When a file system is mounted (using the /etc/mount command), an entry is made in the /etc/mnttab file. When a file system is unmounted (using the /etc/umount command), the entry that corresponds to that file is removed from the /etc/mnttab file.

5.16.2 /etc/fstab

The system administrator maintains the /etc/fstab file. When you set up an /etc/fstab file, it has the following four primary purposes:

Note: The /etc/fstab file provides a way to mount user file systems automatically whenever the system is brought up to multiuser mode. For any file system that you want the /etc/rc script to mount automatically, set the fourth field of the /etc/fstab file for that entry to CRI_RC=YES.

- It contains a list of files that the start-up /etc/bcheckrc script checks by invoking the /etc/mfsck command, which does multiple synchronous file system checks (/etc/fsck).
- It allows a shortcut to be taken by using the /etc/mount command. When a mount command is invoked with only a special file name or only a mount point specified rather than both, the /etc/fstab file is searched for the missing arguments. For example, if you entered the /dev/dsk/usr01 file system information in the /etc/fstab file, instead of typing the following command:

```
/etc/mount /dev/dsk/usr01 /usr01
```

you can type one of the following commands instead:

```
/etc/mount /usr01
```

```
/etc/mount /dev/dsk/usr01
```

- It provides a convenient way to mount file systems with file system quotas enforced.

For descriptions of the `fstab` fields, see the `fstab`(5) man page.

Procedure 8: Configuring a file system to be mounted automatically at the initialization of multiuser mode

If you want any file system to be mounted automatically when multiuser mode is initialized, you must edit the `/etc/fstab` file. Because the `/etc/fstab` file may have read-only permission, you must check the permissions on the file before you try to edit it to ensure that the file has write permission (see step 1). The system can be in either single-user or multiuser mode.

If the system is in single-user mode and the only file system available is `root` (`/`), the only available editor is the `ed` editor. The `vi` editor is located in the `/usr` file system, which is not mounted. If you check (using `fsck`) and mount (using `mount`) the `/usr` file system, the `vi` editor will be available to you even though you are in single-user mode. If the system is in multiuser mode, the `vi` editor is available and can be used to edit the `/etc/fstab` file.

1. Edit the `/etc/fstab` file by using either the `ed` editor or the `vi` editor.

When trying to edit a file, you may encounter a message that a file is "read only." One solution is to change the permissions of the file so that it can be edited, then return the permissions to their original settings when you are finished making changes. The example shown uses the `/etc/config/rcoptions` file.

```
#ls -la /etc/config/rcoptions
-r--r--r-- 1 root root 1914 Mar  8 11:29 /etc/config/rcoptions
# chmod 644 /etc/config/rcoptions
-rw-r--r-- 1 root root 1956 Mar  8 17:28 rcoptions
# vi rcoptions


(make changes)


# chmod 444 /etc/config/rcoptions
-r--r--r-- 1 root root 1914 Mar  8 11:29 /etc/config/rcoptions
```

If you are using the `vi` editor, you can accomplish the same effect by making your changes to the file and, from within the `vi` editor, typing the following command, which forces a write to the file:

```
<escape>:w!
```

2. Uncomment the line for any file system already mentioned in the `/etc/fstab` file that you want to be checked and mounted automatically when the system goes to multiuser mode, or add a line (with the appropriate format) for the desired file system if it is not already mentioned.
3. Edit the fourth field for the desired file system entry to read `CRI_RC=YES`.
4. Save the changes you have made to the `/etc/fstab` file.

Procedure 9: Unmounting file systems

At shutdown, the `/etc/shutdown` script unmounts all file systems; however, you may want to make a file system unavailable during normal operation for maintenance purposes. By convention, the `/mnt` directory is used to mount a file system that needs maintenance. To make a file system unavailable to users, unmount it by using the `umount` command. *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022.

Caution: The file system must be idle before you can unmount it. To determine whether the file system is idle, use the `/etc/fuser` utility.

The argument you specify on the `/etc/umount` command line can be either the name of the mount point or the special device name for the file system you want to unmount.

Examples:

```
# /etc/umount /usr01
```

```
# /etc/umount /dev/dsk/usr01
```

You also can use the `/etc/umountem` script to unmount all file systems quickly while in single-user mode. It executes the `/etc/mount` command to receive a list of the file systems that are currently mounted, edits the list to produce a script of `/etc/umount` commands, and then executes the script. The `umount` command flushes the file system cache to the disk before actually unmounting the file system.

```
# /etc/umountem
```

Backing Up and Restoring File Systems [6]

This section describes how to maintain file systems by creating backup copies of them regularly (also called *backing up* a file system). It also describes how to restore your file systems. *Backing up* a file or file system means to create another copy of it on different storage media (using `dump`); the copy could then be used to replace the original if the original had been damaged or destroyed. *Restoring* a file or file system means to overwrite the current disk file or file system (using `restore`) with the back-up copy.

> **Note:** Backing up large file systems is a resource-consuming task. File saving procedures ideally should be performed in single-user mode with file systems unmounted; therefore, frequent backups mean less time available for user processing. You must adopt a file-backup schedule that is best for your site.

Backing up your file systems on a regular basis ensures users against the loss of time, effort, and valuable information if a file system is corrupted or disk crash occurs. New users (occasionally even experienced ones) may sometimes remove files by mistake. As the system administrator, you must develop and maintain adequate backup procedures.

The following utilities are available for partial file system backup tasks:

- Archiving and extracting files with tape (using `tar`)
- Copying file archives while maintaining status and path names (using `cpio`)

6.1 Related backup and restore documentation

The following documentation contains information covered in this section:

- *General UNICOS System Administration*, Cray Research publication SG–2301, section on file system planning
- *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `dump`(5) and `fstab`(5) man pages
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `dump`(8), `rdump`(8), `restore`(8), and `rrestore`(8) man pages

- *Tape Subsystem Administration*, Cray Research publication SG–2307: tape-related information

6.2 Tape devices referenced in `/dev/tape`

To use the various UNICOS utilities to back up and restore files not using the `tpdaemon`, the tape devices are in the `/dev/tape` directory in which the `tpdaemon` addressable devices also reside. For more information on naming tape devices, see the *Tape Subsystem Administration*, Cray Research publication SG–2307.

6.3 Backup and restore utilities

The following utilities have somewhat different capabilities to back up or restore your file systems. This subsection also recommends when to use each of the utilities. (This section of the guide describes using the `dump` and `restore` utilities for standard file system maintenance.) The `dump` and `restore` utilities are excellent utilities to use because they function based on the concepts of file systems.

6.3.1 `dump` and `restore` utilities

The `dump` and `restore` utilities are recommended for performing file system backups and restores because you can examine the contents of a tape of dumped files without actually reading the entire tape.

The `dump` utility writes a header, which lists the contents of the dump tape on a tape volume. The `restore` utility can read this tape header. The `restore` utility has a simple interactive option that allows an administrator to select some or all of the tape contents for restoration by marking desired files listed in the header.

6.3.2 `rdump` and `rrestore` utilities

The `rdump` and `rrestore` utilities are used to perform the same tasks as the `dump` and `restore` commands across a TCP/IP network.

6.3.3 dd utility

The `dd` utility is used for copying data directly from a disk partition. `dd` is a good tool for creating absolute block-by-block copies of entire file systems. The `dd` utility converts and copies a file to the specified output device (disk-to-disk backup).

6.3.4 `tar` and `cpio` utilities

The `tar` and `cpio` utilities copy regular files or directories to disk or tape. These utilities are best suited for saving portions of file systems (a series of files or directories of files) that can be written to one tape (round or cartridge). One limitation is that you must read the entire contents of the tape to determine what files reside on the media. `tar` archives files to tape, and `cpio` copies files; `cpio` uses standard input and standard output so it generally is used in conjunction with I/O redirection and/or command-line pipeline.

6.3.5 `root` and `usr` file systems

There is a minor, but significant change to the disk space allocation scheme forCRAY J90se systems. At initial installation, two sets of production disk partitions named `roota/usra/srca` and `rootb/usrb/srcb` are created. This lets you install into another set of partitions in multiuser mode without disturbing the running system. The CRAY J90se installation utility is designed to perform upgrade installations into the alternative set of partitions. These procedures makes references to `root` and `usr` disk partitions; therefore, if you are using a CRAY J90se system, replace `root` and `bkroot` with `roota` and `rootb`, respectively, and replace `usr` and `bkusr` with `usra` and `usrb`, respectively, in the creation and booting procedures.

At initial installation, the `bkroot` and `bkusr` disk partitions are created automatically for you, except for CRAY J90se systems that have extremely limited disk space. If these disk partitions are not defined, you must define them before continuing with the creation and booting procedures.

Procedure 10: Creating `bkroot` and `bkusr` file systems

You can create a bootable copy of your production `root` and `usr` file systems into file systems called `bkroot` and `bkusr`. You should perform this procedure before upgrading an operating system as a fall-back preparation measure, or when you are sure that no outstanding problems exist with your current production system. You should not run this procedure by using the `cron` utility. This procedure also is not a substitute for making regular backups to

tape. You should perform this procedure when the activity on the `root` and `usr` file systems is at a minimum.

The following are the steps for creating a bootable copy of your production `root` and `usr` file systems into file systems called `bkroot` and `bkusr`.

1. Create the directories and file structure as follows, replacing values in italics with values appropriate to your system. Use an `mkfs` big file allocation option suitable for the disk types on which `bkroot` and `bkusr` reside for *DEVTYPE*. If you use the UNICOS MLS product, you may have to add security-related options for system levels and compartments (*SECURITY_OPTIONS*).

```
# export PATH=$PATH:/etc
# mkdir -p /mnt
# mkdir -p /mnt2
# mkfs  -q -ADEVTYPE  SECURITY_OPTIONS/dev/dsk/bkroot
# mkfs  -q -ADEVTYPE SECURITY_OPTIONS  /dev/dsk/bkusr
```

2. Label the file systems, as follows:

```
# labelit  /dev/dsk/bkroot bkroot cray
# labelit  /dev/dsk/bkusr  bkusr cray
```

3. Check file system consistency, as follows:

```
# fsck -u /dev/dsk/bkroot # fsck -u /dev/dsk/bkusr
```

4. Mount the file systems, as follows:

```
# mount /dev/dsk/bkroot /mnt
# mount /dev/dsk/bkusr  /mnt2
```

5. Flush data from all logical device caches to disk and dump the file system by executing the following sequences of commands:

```
# cd /mnt
# ldsync; sync; sleep 4
# dump -t 0 -f - /dev/dsk/root | restore -r -f - &;
# cd /mnt2
# ldsync; sync; sleep 4
# dump -t 0 -f - /dev/dsk/usr | restore -r -f - &;
```

Note: Using `dump` piped to `restore` to copy the file systems mean that you will get the benefits of defragmentation and file system validity checking, but you may lose the ability to continue batch work that was checkpointed before the system backup.

6. Unmount and check the `bkroot` file system, as follows:

```
# cd /mnt
# rm restoresymtabl
# echo "root backed up to bkroot on ‘date‘" >> bkroot.log
# cd /
# umount /dev/dsk/bkroot
# fsck -u /dev/dsk/bkroot
```

7. Unmount and check the `bkusr` file system, as follows:

```
# cd /mnt2
# rm restoresymtabl
# echo "usr backed up to bkusr on ‘date‘" >> bkusr.log
# cd /
# umount /dev/dsk/bkusr
# fsck -u /dev/dsk/bkusr
```

Procedure 11: Booting `bkroot` and `bkusr` into production

The following are procedures to boot the `bkroot` and `bkusr` file systems into production.

Note: If your production `root` and/or `usr` file systems are damaged beyond the repair of `fsck`, and you have booted on a `bkroot` and `bkusr` file system to single-user mode, you can either restore your production `root` and `usr` file systems from a recent backup tape or apply the `bkroot` creation procedure in reverse order to create a new production `root` and/or `usr` file system. Boot to multiuser mode on `bkroot` and `usr`. To aid the handling of checkpointed work, you may want to disable the automatic startup of NQS while in single-user mode by editing the `/etc/config/daemons` file.

1. Shut down UNICOS by executing the following commands:

```
# /etc/shutdown
# sync
# sync
# sync
```

2. Edit the `/opt/CYRIos/`*sn9xxx*`/param` file and boot the SIO by executing the following commands (this may be done on the CRAY J90se system console, in which case the prompt will be `sn9` *xxx*`>`):

```
sn9xxx> cd /opt/CYRIos/sn9xxx
sn9xxx> cp param param.prd
sn9xxx> cp param param.bkr

sn9xxx> ed param.bkr
     1
     /rootdev/
     s/ldd root /ldd bkroot/p
        rootdev is ldd bkroot;
     w q
sn9xxx> cp param.bkr param
sn9xxx> cd /
sn9xxx> bootsys -c
```

3. Check the `bkusr` file system and mount the `bkusr` file system, as follows:

```
# fsck -u /dev/dsk/bkusr
# mount /dev/dsk/bkusr /usr
```

4. Edit the `/etc/fstab` UNICOS configuration files by using the `vi` command and change the following displayed lines:

```
# vi /etc/fstab


change root, usr, bkroot and bkusr lines from:


/dev/dsk/root     /          NC1FS  CRI_RC=NO,rw  1  1
/dev/dsk/bkroot   /mnt       NC1FS  CRI_RC=NO,rw  1  4
/dev/dsk/usr      /usr       NC1FS  CRI_RC=NO,rw  1  2
/dev/dsk/bkusr    /mnt/usr   NC1FS  CRI_RC=NO,rw  1  2


to the following:


/dev/dsk/root     /mnt       NC1FS  CRI_RC=NO,rw  1  4
/dev/dsk/bkroot   /          NC1FS  CRI_RC=NO,rw  1  1
/dev/dsk/usr      /mnt/usr   NC1FS  CRI_RC=NO,rw  1  2
/dev/dsk/bkusr    /usr       NC1FS  CRI_RC=NO,rw  1  2
```

5. Edit the `/etc/config/rcoptions` file by using the `vi` command and change the following displayed lines:

```
# vi /etc/config/rcoptions


change ROOTDEV, PIPEDEV, and USRDEV lines from:


   ROOTDEV='root'
   PIPEDEV='root'
   USRDEV='usr'


to the following:


   ROOTDEV='bkroot'
   PIPEDEV='bkroot'
   USRDEV='bkusr'
```

6. Unmount the `/usr` file system by executing the following command:

```
# umount /usr
```

7. Enter multiuser mode by executing the following command:

```
# /etc/init 2
```

Procedure 12: Backing up the SWS

To create a backup copy of the SWS database, see the "SWS Database and File System Backups" section of the *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204.

6.4 `/etc/dump` utility

The `/etc/dump` utility provides either full or incremental file system dumps. Dump level numbers 0 through 9 are used to determine the files that will be dumped. Dump level 0 causes the entire file system to be dumped. You can arbitrarily assign levels 1 through 9 (9 is considered the lowest level). A description of some important options follows. For a complete description of all the options, see the `dump`(8) man page.

Option	Description
`-A` *altfile*	Specifies the name of a file to contain a second copy of the output from the beginning of `dump`.
`-c`	Writes to cartridge tape.
`-f` *file*	Places the dump in a disk file, rather than on tape.
`-t` *dump_level*	Specifies the dump level. Default is level 9.
`-u`	Writes the date and time of the beginning of dump in the `/etc/dumpdates` file. A separate date is maintained for each file system and dump level.
`-v` *vsn_list*	Specifies a list of volume serial numbers (VSNs) to use for output. If you omit this option, `dump` prompts the operator for a list of VSNs.

`-w` Prints the file systems that must be dumped. This information is gathered from the dump frequency field in the `/etc/fstab` and `/etc/dumpdates` files.

Note: The `/etc/dump` utility is slow. Files are dumped (written) to tape in inode number order. The `/etc/dump` utility begins by traversing across and down the directory hierarchy of the file system, creating an index. This index is written to the first tape preceding data. The `restore` utility uses this index information.

6.5 Routine backup (dump) strategy

You can make two different types of backups: *full backups* or *partial backups*. Which type of backup you choose to use depends on your site, the time involved to make the backups, and the amount of media you can use for the backups. Perform file system dumps when the system is as quiet as possible. You do not have to be in single-user mode to perform file system backups. A *full backup* copies all user areas, UNICOS files, and any other special files. Full backups are often done to document the system status at a particular point in time (for example, immediately before a software update). A *partial backup* is usually more appropriate for copying everyday work; it is easily customized to individual sites.

The `dump` utility dumps all file system files that have been modified since the most recent dump that was performed at a lower level. For example, if a level dump was performed on Sunday, a level 9 dump was performed on Monday, a level 8 dump was performed on Tuesday, and a level 9 dump was performed on Wednesday, then Monday's level 9 dump tapes would contain all changes since Sunday's level dump. Tuesday's level 8 dump tapes also would contain all changes since Sunday's level dump (Sunday's level dump is the most recent dump with a dump level value less than 8). Wednesday's level 9 dump tapes would contain all changes since Tuesday (Tuesday's level 8 dump is the most recent dump with a dump level less than 9).

You should save all of the tapes that would be required to recover a given week's work for at least two weeks. Some sites use five different sets of tapes, one set for each week of a month, and the fifth set for the first week of the following month. For the second week of the following month, the first of the five sets of tapes is overwritten. With this strategy, only five sets of back-up tapes are required, and a one-month rolling window of file system contents is preserved.

Most sites perform a full file system dump (level 0) once a week and level 9 dumps every day until the next week's full level dump.

The following is the recommended routine back-up (dump) strategy. It involves performing a full (level 0) dump to cartridge tape on a weekly basis and incremental (level 9) dumps on a daily basis. If you follow this plan, you need only two sets of tapes to reload a file system: the weekly dump and the most recent daily dump.

- Once per week: You should do a full (level 0) dump.
  - Repeat for each file system you want to copy.
  - Because the `dump` command can read unmounted file systems, you can unmount the file system to be dumped before you begin.
- Daily: You should perform an incremental (level 9) dump:
  - Dump everything that has been modified since the last dump performed with a lower dump level.
  - Repeat for each file system you want to copy.
  - Do this each day that you do not perform a level 0 dump.

6.6 Restoring file systems

The `restore` utility (`/etc/restore`) processes tapes produced by `/etc/dump`. The main options are as follows (for a complete description of all options, see the `restore`(8) man page):

Options	Description
`-c`	Reads from cartridge tape.
`-f` *file*	Reads the dump from a disk file, rather than tape.
`-i`	Initiates interactive restoration. A shell-like interface is provided that lets the user traverse through the directory tree and select files to be restored.
`-r`	Reads entire tape and loads into current directory. Do this only if you run `mkfs` on the file system first. You usually should do a full `dump` after a full `restore`.
`-t`	Lists the specified file names if the files are on the tape. If no file names are specified, all files on the tape are displayed.

`-v` *vsn* Causes `restore` to type the name of each file it treats, preceded by its file type.

`-x` Extracts specified files from the tape (creates subdirectories as necessary).

Note: Because of the use of synchronous write operations, `restore` is slow. `restore` wants to ensure that directory files were created before trying to write files into directories. Because the interface on the `restore` utility also is rather limited, be sure to use the `-i` (interactive) option when possible.

6.7 Increasing and decreasing file system space

Reorganizing file systems can involve one or more of the following activities: increasing and decreasing file system space, and/or reducing file system fragmentation.

A file system can become fragmented. The amount and occurrence of fragmentation occurs with a combination of factors: changes in a file system, low free space in a file system, and amount of time a newly created file system is in use.

Not all file systems suffer from fragmentation. For example, `/root` and `/usr` contain many directories and commands that never change; but the user and spooling (if separated) file systems are in constant change. When you want to decrease file system fragmentation, perform a full dump and restore of that file system.

6.8 Tape devices

As of the UNICOS 9.0.2 release, the following tape device names are no longer supported: `/dev/rss00`, `/dev/rmt00`, `/dev/rpe02`, `/dev/rpq01`, `/dev/rpd03`. The new naming convention uses tape daemon character devices as configured in the `/etc/config/text_tapeconfig` file. An example would be `/dev/tape/t120`, where `t`=tape, `2`=SCSI bus number, `1`=SCSI target number, `0`=logical unit number.

The list of supported tape device names changes periodically, and to remain up-to-date on supported tape devices, refer to Product Availability Notice (PAN) #111. See your Cray service representative for details on how to access this information.

6.9 Procedures included in this section

This subsection includes the following procedures:

- Backing up (dumping) a file system without `tpdaemon`
- Restoring a file system without `tpdaemon`
- Backing up (dumping) a file system by using `tpdaemon`
- Restoring a full file system by using `tpdaemon`
- Restoring a partial file system by using `tpdaemon`

  Note: To back up and restore in batch requires you to set limits on the user database (UDB) account being used and on the Network Queuing System (NQS) queue. You also must use the `qsub -lU` command.

Procedure 13: Backing up (dumping) a file system without `tpdaemon`

In this method of doing a dump, you will use the UNIX-accessible logical tape devices that are defined in the `/dev` directory, as opposed to the `tpdaemon` -accessible devices defined in the `/dev/tape` directory.

Note: For this example, a square (`CART`) tape device, with the name `/dev/tape/CART`, is used.

For the rest of this example, a square-tape (`CART`) device with the name `/dev/tape/CART` is used.

1. To determine whether the tape is in a physically writable condition, load the device and enter the following command:

```
sn5111# mt -f device status
```

If necessary, physically alter the tape so that you can write to it.

For round (`TAPE`) tapes, if a plastic ring is clipped to the inner diameter of the tape, you can write to the tape. If no ring is clipped to the inner diameter of the tape, you cannot write to the tape.

For square (`CART`) tapes, you can roll a small plastic wheel back and forth. If the wheel is rolled so that the dot shows, you cannot write to the tape. If you roll the wheel so that the dot does not show, you can write to the tape.

For quarter-inch cartridge (`QIC`) tapes, a small black dial near one corner of the tape has a raised piece of plastic in the shape of a >. If the point of the shape

points at the word SAFE (for example, > SAFE), you cannot write to the tape. If you twist the dial so that the point of the shape points away from the word SAFE (for example, < SAFE), you can write to the tape.

For EXABYTE tapes (type EXB), on the edge of the tape you can pull a small red piece of plastic along the length of the tape so that it covers a small hole. If the piece of red plastic shows and the hole is covered, you cannot write to the tape. If you slide the piece of red plastic back so that it cannot be seen and the hole is exposed, you can write to the tape.

For digital audio tapes (DAT), a small white plastic slide covering the hole indicates you can write to the tape. If this white plastic slide exposes the hole, you cannot write to the tape.

2. Physically mount a tape in the tape drive that matches the logical tape device you want to use (for this example, a square (CART) tape was mounted in a drive that corresponds to the logical device `/dev/tape/CART`).

3. Rewind the tape. Round-type tape drives rewind the tape automatically when you push the button to select the tape to be loaded. It is a good practice to be cautious and rewind other types of tapes when they are used. Because round tapes are used in this example and they rewind automatically, you probably would exclude this step; however, to rewind other types of tapes, enter the `mt -f /dev/` *tapename* `rew` command and specify the tape device you are rewinding (`-f /dev/` *tapename* specifies the raw tape device to be activated). For example, to rewind an EXABYTE (`/dev/rpe02`) tape, type the following command line:

```
sn5111# mt -f /dev/tape/CART rew
```

   Note: When dumping an open file, the updated file will not be dumped until the file is written to disk. If you want to ensure that all files are dumped, you should unmount the file system.

4. Dump the file system to tape. In this case, a full level (`-t 0`) dump (as opposed to a partial dump) is performed. The `-u` option is highly recommended. If you invoke this option, the date and time of the beginning of the dump will be written to a file called `/etc/dumpdates`, and a separate entry for each file system and each dump level will be recorded.

If this is the first time the `dump` command has been used on your system with the `-u` option, the `/etc/dumpdates` file probably does not exist. This causes the following error message at the end of the `dump` command screen output:

```
dump (/src to /tmp/dumpfile): dump has completed, 23618 blocks
dump (/src to /tmp/dumpfile): cannot open an existing /etc/dumpdates file
dump (/src to /tmp/dumpfile): The dump is aborted.
```

To prevent this error, you must create an empty file named `/etc/dumpdates` before executing the `/etc/dump` command. One way to do this follows:

```
sn5111# touch /etc/dumpdates
```

The following example shows a full file system dump (`-t 0`) to a square tape (`-f /dev/tape/rss00`):

```
sn5111# /etc/dump -t 0 -u -f /dev/tape/CART /dev/dsk/src
```

Note: You may want to append an & symbol to the end of the `/etc/dump` command line so that this command operates as a background process and you can still perform other operations (such as responding to operator messages) while the `dump` command is running.

5. Physically alter the tape to prevent the tape from being overwritten (see information included in step 2).

6. Attach a physical label to the tape that states the file systems that have been dumped to the tape and the date the tape was written. It may be useful to add the command that was used to write the tape. It also may be useful to add the commands necessary to restore the tape.

Your file system backup (dump) is now complete.

Procedure 14: Restoring a file system without `tpdaemon`

In this method of doing a restore, you will use the UNIX-accessible logical tape devices that are defined in the `/dev` directory, as opposed to the `tpdaemon` -accessible devices defined in the `/dev/tape` directory. A *full file system restore* means that the entire contents of a file system will be read in from tape and will overwrite the current disk version of that file system. A *partial file system restore* restores only a file or directory or some subset of a file system to the logical device; the rest of the file system remains untouched.

1. If it was not already unmounted, unmount the file system to be restored by using the `/etc/umount` command. The `/dev/dsk/src` file system is used in this sample procedure.

Caution: Before you can unmount it, the file system must be idle. To determine whether the file system is idle, you can use the `/etc/fuser` utility.

To determine whether the file system in question is currently mounted, examine the output of the `/etc/mount` command:

```
sn5111# /etc/mount
/ on /dev/dsk/root read/write on Fri Feb 11 10:38:15 1994
/tmp on /dev/dsk/tmp read/write on Fri Feb 11 10:40:56 1994
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Fri Feb 11 10:40:59 1994
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Fri Feb 11 10:41:02 1994
/usr/src on /dev/dsk/src read/write,rw,CRI_RC="YES" on Fri Feb 11 10:41:04 1994
```

In this case, the last line of the output from the `/etc/mount` command shows that the `/dev/dsk/src` file system is currently mounted on the mount point `/usr/src`.

The `/etc/umount` command unmounts the file system. You can specify either the mount point or the file system logical device name after the `umount` command for it to be effective. In the following example, the logical device name was used:

```
sn5111# /etc/umount /dev/dsk/src
```

Now the output of the `mount` command shows that the `/dev/dsk/src` file system is no longer mounted:

```
sn5111# /etc/mount

/ on /dev/dsk/root read/write on Fri Feb 11 10:38:15 1994
/tmp on /dev/dsk/tmp read/write on Fri Feb 11 10:40:56 1994
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Fri Feb 11 10:40:59 1994
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Fri Feb 11
10:41:02 1994
```

Warning: The following step deletes all information on this file system.

2. **Complete this step only if you are doing a full file system restore. If you are doing a partial file system restore, skip to step 4.** Remake the file system structure on the `/dev/dsk/src` logical device by using the `/etc/mkfs` command.

The `-q` option on the `mkfs` command shown in the following example is optional syntax. Using this option bypasses the disk surface check and speeds the `mkfs` process, but it is not the most thorough way to prepare the disk for a file system structure. The first time you use the `/etc/mkfs` command to format a logical disk area for a file system structure, do not use the `-q` option. For more information about the `mkfs` command, see section Chapter 5, page 35, and the `mkfs`(8) man page.

```
sn5111# /etc/mkfs -q /dev/dsk/src
```

3. Check the file system by using the `/etc/fsck` command. Before mounting the file system, you **must** perform this command:

```
sn5111# /etc/fsck /dev/dsk/src
```

4. Make sure that no other file system is mounted on the directory in which you intend to mount your file system. You must mount the file system being restored on a mount point where you can perform the remaining administrative tasks without users being affected or interfering. Traditionally, the mount point that administrators use for such tasks is `/mnt`, because `/mnt` is not a directory users are likely to access. If the system is in multiuser mode, users probably will not interrupt administrative tasks being performed in that directory.

To check that no other file system is mounted on the directory in which you intend to mount your file system, examine the output of the `etc/mount` command, which lists all file systems currently mounted and their mount points, as shown in the following example:

```
sn5111# /etc/mount
/ on /dev/dsk/root read/write on Mon Feb 14 19:09:25 1994
/tmp on /dev/dsk/tmp read/write on Mon Feb 14 19:10:01 1994
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Mon Feb 14 19:10:04 1994
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Mon Feb 14 19:10:07 1994
```

The `mount` command output shows that no file systems are mounted on the `/mnt` mount point.

5. Mount the file system on the mount point you have selected by using the `/etc/mount` command. In this example, the mount point `/mnt` is used. If any user is in the directory or any of its subdirectories, the `mount` command will not be successful (to remove users from a file system forcibly, see the `fuser`(8) man page). If you are the one in the directory or a subdirectory, change to a directory that is not part of the directory tree, including the mount point directory or any of its subdirectories, as follows:

```
sn5111# cd /
sn5111# /etc/mount /dev/dsk/src /mnt
```

6. Change directories to the mount point directory on which the file system is mounted. In step 6, `/mnt` was selected to be used for this directory:

```
sn5111# cd /mnt
```

7. Physically mount a tape in the tape drive that matches the logical tape device you want to use. In this example, a square (`CART`) tape was mounted in a drive that corresponds to the logical device `/dev/rss00`. The contents of this tape should include the dumped file system you want to restore, in this case, `/dev/dsk/src`.

8. Rewind the tape. Round-type tape drives rewind the tape automatically when you push the button to select the tape to be loaded. You should be cautious and rewind other types of tapes when they are used. However, to rewind other types of tapes, enter the `mt -f /dev/` *tapename* `rew` command and specify the tape device you are rewinding (`-f /dev/` *tapename* specifies the raw tape device to be activated). For example, to rewind a square (`/dev/tape/CART`) tape, type the following command line:

```
sn5111# mt -f /dev/tape/CART rew
```

9. Restore either the full file system or, if you are doing a partial restore, restore the files and/or directories of the file system that are needed (in this case, `/dev/dsk/src`).

There are two methods of restoring file systems. This step does not discuss the interactive method in detail, but it is highly effective and very easy to use. It is invoked by using the `-i` option. For a good explanation of how to interact with the interactive shell interface to select files and directory contents for restoration, see the `restore`(8) man page.

The `-f` option addresses the logical device on which the dump tape is mounted.

To invoke the interactive method at this point, type the following command line:

```
sn5111# /etc/restore -i -f /dev/tape/CART
```

The other method of restoring a file system follows for doing a full or a partial file system restore.

To do a full file system restore :

The `-r` option invokes a full file system restore (this example is for a square tape). You may want to run the `restore` command as a background process by appending an & symbol to the end of the command line.

```
sn5111#  /etc/restore -r -f /dev/rss00
```

To do a partial file system restore :

Two examples are given. One example restores the `/src/uts/Nmakefile` file to the `/src` file system. The other example restores the `/src/uts/c1/sys` subdirectory and all of its contents to the `/src` file system. You may want to run the `restore` command as a background process by appending an `&` symbol to the end of the command line.

The `-x` option invokes a partial file system restore. You should list the files or directories that you want extracted from tape as the last arguments of the command line. You should specify the path name for each file you want to restore relative to the topmost directory of the file system in which it resides.

In the following example, the `Nmakefile` file is restored. The file's full path name in the `/src` file system is `/src/uts/Nmakefile`. The file's path name is relative to the topmost directory of the `/src` file system; that is, relative to `/src`, it is `/uts/Nmakefile`.

```
sn5111# /etc/restore -x  -f /dev/tape/CART /uts/Nmakefile
```

The following example restores the /src/uts/c1/sys directory and all of its files and subdirectories and their contents:

```
sn5111# /etc/restore -x  -f /dev/tape/CART /uts/c1/sys
```

10. Unmount the file system when the restore has completed, as follows:

```
sn5111# /etc/umount /dev/dsk/src
```

11. Remount the restored file system on its normal mount point and check the file system, as in the following example:

```
sn5111# /etc/fsck /dev/dsk/src
```

If the file system was unmounted cleanly, this step is optional.

12. Mount the restored file system on its normal mount point. The file system is now ready for users to access. The /src file system usually is mounted on the /usr file system, as follows:

```
sn5111# /etc/mount /dev/dsk/src /usr/src
```

The file system restoration of /src is now complete.

Procedure 15: Restoring a partial file system by using tpdaemon

Assumptions

For this procedure, it is assumed that the tpdaemon is up and that all tape hardware (devices, controllers, and so on) are configured to be up and available to the user. In the following example, the restore is done from a square (CART) tape. That square tape was written as unlabeled (-l nl), and it had a volume name of JON1.

A partial file system restore is being performed. This means that only a file or a directory or some subset of the file system is restored to the logical device. The rest of the file system remains untouched.

The file system restored in the example is the same file system backed up (dumped) in the backup procedure (/src).

As with the `dump` command, you should execute the `restore` command on as quiet a system as possible. Tell the user who owns the files being restored to stay out of that directory until the restore is completed.

In the backup (dump) example, `/dev/dsk/src` was dumped using the following command:

```
sn5111# /etc/dump -t 0 -u -g CART -v JON1 -l nl /dev/dsk/src
```

When doing a restore, the value for the `-l` (label) option and the `-v` (volume) option must match the label and volume that you used on the `/etc/dump` command.

Procedure 15(a): Partial file system restore

The following are the steps for performing a partial file system restore:

1. Determine which tape device group that you plan to use for your restore. This will be the same as the preceding backup (see step 1 of the backup procedure).
2. Verify that the tape(s) written from the preceding backup are set to prevent tape contents from being overwritten (see step 2 of the backup procedure).
3. Change directories to the mount point directory on which the file system is mounted. Because you are restoring files in `/usr/src`, enter the following command:

   ```
   sn5111# cd /usr/src
   ```

4. Restore the files and/or directories of the file system that are needed. In this case, the `/dev/dsk/src` file system is used.

There are two methods of doing file system restores. This procedure does not discuss the interactive method in detail, but it is highly effective and very easy to use. To invoke it, use the `-i` option. The `restore`(8) man page gives a good explanation of how to interact with the interactive shell interface to select files and directory contents for restoration.

To invoke the interactive method, type the following command line:

```
sn5111# /etc/restore -i -V JON1 -l nl -g CART
```

A description of the other method of performing file system restoration follows.

Two examples follow. One example restores the `/src/uts/Nmakefile` file to the `/src` file system. The other example restores the `/src/uts/c1/sys` subdirectory and all of its contents to the `/src` file system.

The `-x` option invokes a partial file system restore. It also is used for a full file system restore if the `-r` option fails. You should list the files or directories that you want extracted from tape as the last arguments of the command line. You must specify the path name for each file you want to restore relative to the topmost directory of the file system in which it resides.

In the following example, the `Nmakefile` file is restored. The file's full path name in the `/src` file system is `/src/uts/Nmakefile`. The file's path name relative to the topmost directory of the `/src` file system (that is, relative to `/src`) is `/uts/Nmakefile`.

```
sn5111# /etc/restore -x -V JON1 -l nl -g CART /uts/Nmakefile
```

The following example restores the `/src/uts/c1/sys` directory, including all of its files and subdirectories and their contents:

```
sn5111# /etc/restore -x -V JON1 -l nl -g CART /uts/c1/sys
```

5. If no `-g` (tape type group name) option is supplied on the `/etc/restore` command to specify a particular kind of tape device to reserve, `restore` will use the default device group name set by the `DEV_DGN` argument in the `/etc/config/tapeconfig` file. By default, when your system arrives, the `DEV_DGN` argument is set to round (`TAPE`) tapes, as shown in the following excerpt from the `/etc/config/tapeconfig` file:

```
#
#       default device group name
#       Specify a decimal number
#       This must be one of the device types specified in the CNT
#       configuration
#
DEF_DGN         TAPE
```

To change this default, change your tape configuration either by using the menu system `Configure System ==> Tape Configuration` menu or by editing the `/etc/config/tapeconfig` file.

Because the `/etc/restore` command also defaults to `DEF_DGN` tapes, you can specify other tape types by using the `-g` option.

6. The tape mount request initiated by the `restore` command causes the system to inform you that operator messages exist. The following message repeats on the console until you open up a window to reply to operator messages:

```
There are operator messages that require attention
```

To open up a window to reply to the tape mount operator messages, type the following command:

```
sn5111# /usr/lib/msg/oper
```

Your entire screen now shows a display that looks something like the following:

```
Command: msgd  Page: 1 [delay 10] Thu Oct  7 17:09:02 1993

  Msg #   Time    System Messages
  =====   =====   ===============

  1    17:07   TM122 - mount tape JON1(nl) on a CART device for jones
                  14,  () or reply cancel / device name
:: display truncated:
Enter '?' for help.
>
```

At this point, you can respond in one of three ways:

- Mount the tape in the device
- Specify a different device on which you want to mount the tape
- Cancel the request

In the preceding example, assume that an unlabeled CART tape was mounted in a CART drive. This causes another message to appear that will require a response.

7. Respond to the next operator message by specifying the volume serial number (VSN) of the tape.

In this case, the message number was 2 and the volume serial number was JON1, so /usr/lib/msg/rep 2 JON1 was typed at the > prompt, as follows:

```
Command: msgd  Page: 1 [delay 10] Thu Oct  7 17:11:10 1993
  Msg #   Time    System Messages
  =====   =====   ===============
  1    17:07   TM122 - mount tape JON1(nl) on a CART device for jones
                  14,  () or reply cancel / device name
  2    17:10   TM011 - enter vsn for tape on device CART:: display truncated:
> /usr/lib/msg/rep 2 JON1
```

All messages related to your tape job have now disappeared.

8. Exit the operator window, and return to the system prompt by typing exit at the > prompt, as follows:

```
Command: msgd  Page: 1 [delay 10] Mon Oct 11 14:02:23 1993
  Msg #   Time    System Messages
  =====   =====   ===============:: display truncated:
Enter '?' for help.
> exit
sn5111#
```

The partial file system restoration of /src is now complete.

Note: In some cases (you must specify tape block size, for example) it may be necessary to perform the tape daemon interface manually. The following manual example does the same thing as the preceding "automatic" restore example:

```
sn5111# rsv CART
sn5111# tpmnt -l nl -r in -n -v JON1 -g CART -p /tmp/dumpfile
sn5111# /etc/restore -r -f /tmp/dumpfile
sn5111# rls -a
```

Maintaining Users [7]

UNICOS user account information is stored in a user database (UDB). This chapter describes the following topics:

- Brief descriptions of the UDB and the `/etc/xadmin`, `/etc/nu`, and `/etc/udbgen` utilities
- A brief summary of the procedure for adding a user record to the UDB
- Principal UDB files and commands
- Creating a user login
- Modifying user login information in the UDB
- Deleting a user record
- Maintaining user environment files
- Transferring user records to another file system

For information on ways to communicate with users, see Chapter 8, page 173.

7.1 Related user accounts documentation

The following documentation contains detailed information covered in this chapter and additional information about the UDB:

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `chgrp`(1), `chown`(1), `passwd`(1), `su`(1), and `udbsee`(1) man pages
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `nu`(8), `udbgen`(8), and `udbpl`(8) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `acid`(5), `cshrc`(5), `group`(5), `passwd`(5), `profile`(5), `shells`(5), and `udb`(5) man pages
- *General UNICOS System Administration*, publication SG-2301, the section on UDB

7.2 The user database (UDB)

The user database (UDB), which is unique to the UNICOS operating system, contains entries for each user who is allowed to log in and to run jobs on your system. The UNICOS system maintains encrypted login passwords in the UDB, rather than in a separate password file. However, the traditional UNIX `/etc/passwd` and `/etc/group` files are still supported; when the UDB is updated, they are updated automatically.

The only way that the UNICOS system can identify an individual user is by that person's user ID. The system maps the user ID to your user record in the UDB. The system administrator assigns this unique user ID number. The user ID is also a field in the `/etc/passwd` file.

You can modify the UDB in the following ways:

- If you have access to a windowing environment, you can use the `/etc/xadmin` command, which provides a graphical user interface (GUI) for managing user login accounts. This command has all the functionality of the `/etc/nu` utility. This X Window System based interface is self-explanatory and requires no prior knowledge of the `nu` command. It contains a tutorial for an overview of the command and context-sensitive help on specific topics. `xadmin` uses the UNICOS message system to generate its error and help messages. For more information, see the `xadmin`(8) man page.

- If you do not have a windowing environment, you can use the `/etc/nu` utility (see Section 7.5, page 140).

  The `nu` utility is a full-screen, prompt-driven utility that prompts you for the user information that you want to create or modify (for example, login ID, password, and name). The `nu` utility then creates or otherwise modifies the appropriate directories, makes entries in a log file, or (for updates) merges the changes into the `/etc/udb` file. If you have configured the `nu` utility to skip prompting for specific UDB fields, you must use `udbgen` to access these fields.

- You also can use the `/etc/udbgen` utility (see Section 7.6, page 155). The `udbgen` utility is actually the program underlying the `/etc/nu` utility. You can access this underlying utility directly by issuing the `udbgen` utility and its associated directives. The `udbgen` utility does not prompt you for user information. Although using `udbgen` to update the UDB involves a more complicated syntax than `nu`, it can give you more control over the update process. The `udbgen` utility also can enable you to perform batch updates and to update many user accounts at one time. If you have configured the

`nu` utility to skip prompting for specific UDB fields, you must use `udbgen` to access these fields.

7.3 Adding user records to the UDB

The following is a summary of the procedures that you should use to add a user record to the UDB (`etc/udb`):

- Learn about the UDB fields and decide which values to assign to the UDB fields (more than 80 fields exist). Section 7.4, page 131, describes a suggested subset of UDB fields. For a full listing and explanation of all possible fields in the UDB, see the `udbgen`(8) man page. Some of the values you select will affect other factors on the system (for example, the login directory field determines in which file system the user is placed). You must make sure sufficient disk space is available to meet the user's needs in this file system.
- If the user will be placed in a new group that you will reference by name, add the new entry in `/etc/group` (see Procedure 17, page 138).
- If the user will be placed in a new account group that you will reference by name, add the new entry in `/etc/acid` (see Procedure 18, page 139).

Then, if you are using `/etc/nu`, do the following action:

- Follow the procedures in Section 7.5, page 140, to make the new entry in `/etc/udb` by using the `/etc/nu -a` command. (Section 7.5, page 140, also includes procedures for modifying and deleting user records.)

Or, if you are using `/etc/udbgen`, do the following action:

- Follow the procedures in Section 7.6, page 155, to make the desired entry in `/etc/udb` by using the `/etc/udbgen` command. (Section 7.6, page 155, also includes procedures for modifying and deleting user records.)

To help determine when you would use the `/etc/nu` and `/etc/udbgen` utilities, see Section 7.2, page 130.

7.4 UDB files and commands

The following are the principal files related to the UDB:

File	Description
`/etc/acid`	Account name/ID map file for accounting billing group.
`/etc/group`	Group name/ID map and membership file. This file is common to all UNIX environments.
`/etc/nu.cf60`	The `nu` utility configuration file.
`/etc/passwd`	UNIX password file used for compatibility with existing commands. The * symbol replaced the encrypted password field. This file is common to all UNIX environments. Encrypted passwords are stored in the `/etc/udb` file.
`/etc/udb`	Primary user database file; binary file. It stores the user's password.
`/etc/udb.public`	Public version of `/etc/udb` with read permission for "world." All sensitive information has been set to 0.
`/etc/udb_2/udb.index`	Public index file with read permission for "world." All user IDs (UIDs) and user names found in the UDB extension files along with their associated `udb_priva` and `udb_pubva` record offsets appear in this file.
`/etc/udb_2/udb.pubva`	Public file with read permission for "world." New fields that would have been publicly accessible had they been added to

	`/etc/udb.public` appear in this file.
`/etc/udb_2/udb.priva`	Private file that can be read only by privileged users. The same rules that prevent certain information from appearing in `udb.public` are applied to new fields appearing in this file.

Note: The following scripts are **not**, as released, intended to be used as is; they are only examples that you must modify for your specific site requirements.

Script	Description
`/etc/nulib/nu1.sh`	The `nu` and `xadmin` utilities uses this script to create a user directory and to change the permissions on this directory.
`/etc/nulib/nu2.sh`	The `nu` and `xadmin` utilities uses this script to initialize the contents of the user's directory.
`/etc/nulib/nu3.sh`	The `nu` and `xadmin` utilities uses this script to purge a login account.
`/etc/nulib/nu4.sh`	The `nu` and `xadmin`utilities uses this script to purge a login without removing the account from the UDB. This action is performed to preserve accounting information.
`/etc/nulib/nu5.sh`	The `xadmin` utility uses this script to try and move a user's files when the home directory is changed.
`/etc/nulib/nu6.sh`	The `xadmin` utility uses this script when a user's entry is modified in the user database.
`/etc/nulib/nu7.sh`	The `xadmin` utility uses this script when a nis+ exit is taken for a user entry in the user database.

The following are the principal commands related to the UDB:

Command	Description
`/etc/xadmin`	Graphical user interface that has all of the functionality of the `/etc/nu` command and uses the above list of scripts.
`/etc/nu`	Adds, deletes, and modifies login records and uses the above list of scripts.
`/bin/passwd`	Creates or changes a user's password
`/bin/udbsee`	Converts information from the user database into an ASCII format
`/etc/udbgen`	Generates and maintains the user database
`/etc/udbpl`	Writes to `stdout` administrative information for designated users
`/etc/udbchain`	Verifies and diagnoses problems with the user database files.

The remainder of this chapter includes information and procedures about using the `/etc/nu` and `/etc/udbgen` utilities to maintain your user records.

Procedure 16: Determining settings for UDB fields

The UDB (`/etc/udb`) contains information for each user who is allowed to log in and run jobs on your system. The UDB also contains many other fields that are specific to the UNICOS environment. Fields that you can specify for each user include settings that specify limits for batch processing, interactive processing, account security, the data migration facility, CPU access, disk quotas, the fair-share scheduler, and many others. You must provide the appropriate settings for the fields and resource limits in the UDB for each user record.

For a full listing and explanation of all possible fields in the UDB, see the `udbgen`(8) man page, which includes several examples.

Note: The following UDB fields are a suggested minimum subset of the UDB fields that you should define for each user. The `keyword:` *value* `:` syntax of each entry that follows reflects the format accepted by the UDB if you use the `/etc/udbgen` utility; however, when using the `/etc/nu` utility, you do not use this format (see Section 7.5).

Basic user account definition fields

You should define the following UDB fields for each user (for all possible fields, see the `udbgen`(8) man page).

Note: The global default table contains entries for some of the UDB fields; for a list of these fields, see the `udbgen`(8) man page. The release defaults are applied by `udbgen` when it updates a UDB that has a default table that contains all zeros. To create a default table in an existing UDB, execute the `udbgen -c'#'` command. This command is an empty modification request, but it causes the default table to be created with the released defaults. To change one or more entries, write the appropriate directive line (see Section 7.6).

user_name:	The user's login name must be a unique 1- to 8-character alphanumeric representation, in which the first character is alphabetic.
`passwd`: *encryption*:	Encrypted password to be stored in the user's record. The password content is not validated.
`pwage :force`, `superuser`, *max, min, time*`:`	Manipulate password age control fields by using `pwage`. If you omit a keyword, also omit its separating comma. The *max*, *min*, and *time* fields control how old a password can become (*max*), how long it must exist before being changed (*min*), and when it was changed (*time*). Neither *max* nor *min* may exceed 64 weeks.
`pwage :force`, `superuser`, *max, min, +age*`:`	In the second form of `pwage`, a + symbol preceding the last numeric value causes *age* to be interpreted as the amount of time to subtract from the time "now" to result in a value of the time-of-day clock that is age units in the past and then stores that value in the time field. Usually, this is intended to make it easy to set the current time in the field by using the value +0 as the age. You must precede the time with two commas if you do not specify *max* and *min* ages, because this part of the directive is position dependent, reading from the left to determine the meaning of the value string.
`pwage :` *max* `,` *min*`:`	The third form of `pwage` alters the *max* and *min* age fields.
`pwage ::`	The fourth form of `pwage` removes only age control from a record. All age control fields are

	set to a 0 or null state, which totally removes age control. After this has been done, all historical information is lost from the record. When the `YP` permbit is set (see `permbits`) and the password is being accessed from the database, password aging is disabled.
`uid:` *n*`: uid: next:`	Unique number that represents the user ID. If the value is `next`, the next highest user ID from the UDB is assigned to this user. The value 0 indicates a super-user login. The highest value that you may use is defined in `sys/param.h` as `UID_MAX-1`. You can reset the user database maximum user ID value without rebuilding the entire UDB from source by executing the `udbgen -m` command.
`gids = | + | -:` *n1* `,` *n2* `,... ,` *nn* `: gids = | + | -:` *g1* `,` *g2* `,... ,` *gn* `:`	Comma-separated list of numeric group IDs or group names to which the user belongs. The group limit is 64. If group names are used, they must be found in the `/etc/group` file before executing the `udbgen` command.
`comment:`*text*`:`	Comments that consist of a maximum of 39 characters; white space is not removed. This field is often used for indicating the user's full name, although a site may have other uses for this optional field.
`dir:` *directory* `:`	Default login (`home`) directory for this user relative to the `root` directory. The `dir:` *directory*`:` consists of a string of up to 63 characters. Typically, root is `/`; therefore, `dir` is based on the `root` (`/`) directory, but this does not have to be true. If you do not specify a value, the user is logged in under the `/` directory.
`shell:`*sh_name*`:`	Default login shell. You can specify a maximum of 63 characters. Default value for *sh_name* is `/bin/sh`.
`acids = | + | -:` *nl* `,` *n2* `,... ,` *nn* `: acids = | + | -:` *al* `,` *a2* `,... ,` *an* `:`	Account IDs. This is a list of up to 64 numeric account IDs or account names separated by commas. If you use account names, they must be found in the `/etc/acid` file as it existed before `udbgen` was executed.

`root`: *directory* :	Login `root` directory. You can specify a string of up to 63 characters. `root` specifies the directory to which the base of the user's directory tree is set. (For further information, see the `chroot`(2) man page.)
`nice[b] :` *n* `:` `nice[i] :` *n* `:`	The `nice` value bias in the range 0 < *n* < 19 for batch (`[b]`) or interactive (`[i]`) processes. If you do not specify this field, the value from the default table or the released default value of 0 is used. This field is useful for getting different interactive versus batch and NQS scheduling priority.

User resource limit fields

You can specify user resource limits for both batch and interactive processing in the UDB. The following is a list of some user limits that you may want to set; for a complete list of available limits, see the `udbgen`(8) man page.

Note: A UDB field setting of 0 means "infinite," except for tape access, where 0 means the user has no tape privileges.

`jcpulim[b]:` *n*`:jcpulim[i]:` *n*`:`	Job CPU time limit (in seconds) for batch (`[b]`) or interactive (`[i]`) jobs. The default is unlimited.
`pcpulim[b]:` *n*: `pcpulim[i]`: *n*`:`	Per-process CPU limit (in seconds) for batch (`[b]`) or interactive (`[i]`) processes. The default is unlimited.
`jmemlim[b]:` *n*: `jmemlim[i]:`*n*`:`	Job memory limit in 4096-byte blocks for batch (`[b]`) or interactive (`[i]`) jobs. The default is unlimited.
`pmemlim[b]:` *n*: `pmemlim[i]:` *n*`:`	Per-process memory limit in 4096-byte blocks for batch(`[b]`) or interactive (`[i]`) processes. The default is unlimited.
`jproclim[b]:` *n*`:` `jproclim[i]:`*n*`:`	Job process limit for batch (`[b]`) or interactive (`[i]`) jobs. If you do not specify a value for this field, the default is the value of `/MAXUP` in `sys/param.h`.
SDS limits	Secondary data segments (SDS) are not supported on CRAY J90systems; CRAY J90 system administrators should ignore these fields and use the default setting.

`jtapelim[b][t]:` *n* `:jtapelim[i][t]:` *n*`:`	Job tape unit limit for batch (`[b]`) or interactive (`[i]`) jobs. The integer value `t` represents the tape type. The default tape types are defined in the `DEVICE_GROUPS` section of the `/etc/config/tapeconfig` file. The first type defined in that section is represented by `t=0`, the second is `t=1`, and so on. If *n* is 0, the user is denied tape access.
`pfilelim[b]:` *n* `:pfilelim[i]:` *n*`:`	Per-process file allocation limit in 4096-byte blocks for batch (`[b]`) or interactive (`[i]`) processes. If *n* is 0, the user's file allocation is unlimited.

Procedure 17: Adding a group to `/etc/group`

Note: An important step in adding a user record to the UDB is to assign the user to a group or groups. You may have to add group definitions so that you can make group assignments when you add user records.

As system administrator, you maintain the `/etc/group` text file, which contains the names of groups to which users belong. Groups are created to gather together users who have common needs for accessing files or programs.

You may have to edit the `/etc/group` file to add new group names to the file. The `/etc/nu` command does not allow you to enter a group name in the `gids` field until it has been entered in the `/etc/group` file; however, you may use group ID numbers even if no entry line for that group ID number is in the `/etc/group` file. In this case, a group name is created with the form `G-` *nnnnn*; *nnnnn* is the group ID number. The `/etc/nu` utility updates this file by adding login names to the group login name field. The file contains one entry for each UNICOS group. To delimit an entry line for a group, use a newline character.

To add a group to your system, edit the `/etc/group` file by adding an entry in the following format; **you must separate fields with a colon**:

group_name`:`*unused_password_field_string*`:`*group_id*`:`

Example:

```
ops:*:62:
```

group_name	Name that you choose to reflect the group of users. The group name consists of 1 to 8 alphanumeric characters. The first character must be alphabetic. By convention, lowercase characters are used for group names.

password	This field is not implemented under the UNICOS system. Place an unmatchable character string, such as `*`, in this field.
group_id:	The values 0 to 99 are reserved, by convention, for system-related groups; therefore, you can use group ID values 100 to `UID_MAX-1` for user groups. You should select the next available group ID number for the new group. A colon must follow the *group_id* field.
user	When you create a new group, this field remains blank. Ensure that a colon follows the *group_id* field. The `/etc/udbgen` and `/etc/nu` utilities maintain this field. The list of login names from the group ID field (`gids`) is placed automatically in this *user* field.

To see all group names to which a specified user belongs, use the `groups` command.

To complete adding user records to the UDB, use either the `/etc/nu` or `/etc/udbgen` utility.

See Section 7.5, page 140, for information on how to use `/etc/nu`, and Section 7.6, page 155, for information on how to use `/etc/udbgen`. To determine which utility will work best for you in a given situation, you may want to read both sections.

Procedure 18: Adding an accounting group to `/etc/acid`

As system administrator, you maintain the `/etc/acid` file, which contains the names of accounts associated with users. Accounting groups are implemented for the accounting subsystem, allowing reports to generate information through accounting groups.

Just like the `/etc/group` file, you may have to edit this file to add new account names to the file. The `/etc/nu` command does not allow the use of account names in the `acids` field until an entry has been made in the `/etc/acid` file; however, you may use account ID numbers even if an entry line for that account ID number is not in the `/etc/acid` file. In this case, an account name is created of the form `A-` *nnnnn*; *nnnnn* is the account ID number. The file contains one entry for each UNICOS accounting group. A newline character delimits an entry line for each account.

To add an accounting group to your system, edit the `/etc/acids` file by adding an entry in the following format; **you must separate fields with a colon**:

account_name`:`*account_id*

account_name	Name that you select to reflect the accounting group (for easy identification in accounting reports). The name must consist of 1 to 79 alphanumeric characters. The first character must be alphabetic. Typically, lowercase characters are used for account names.
account_id	(Account identifier) You can use account ID values to `UID_MAX-1`.

Example:

```
markting:93
```

To complete adding user records to the UDB, use either the `/etc/nu` or `/etc/udbgen` utility.

See Section 7.5, page 140, for information on how to use `/etc/nu`, and Section 7.6, page 155, for information on how to use `/etc/udbgen`. To determine which utility will work best for you in a given situation, you may want to read both sections.

7.5 Using the `/etc/nu` utility

The `/etc/nu` utility is a prompt-driven utility for interactively adding, deleting, and modifying user records. It uses a configuration file called `/etc/nu.cf60`. This section describes the following topics:

- Procedure for changing `/etc/nu` configuration parameters
- Procedure for creating a file system to use with `/etc/nu`
- Procedure for adding user records to `/etc/udb` by using the `/etc/nu` utility
- Procedure for modifying user records by using the `/etc/nu` utility
- Procedure for deleting user records by using the `/etc/nu` utility

Procedure 19: Changing /etc/nu configuration parameters

Default values for the /etc/nu utility are in the /etc/nu.cf60 configuration file. To change several parameters in this configuration file, either edit the file or use the menu system. You also can turn off or "hide" prompts for UDB values that you want the /etc/nu program to accept automatically.

The following are common parameters you may want to change; for a complete list of changeable parameters and a description of each, see the nu(8) man page:

Parameter	Description
DefaultAcids	String that contains the default account IDs assigned to the UDB acids field.
DefaultDr	Default login root string assigned to the UDB root field.
DefaultGids	Default group IDs assigned to the UDB gids field.
DefaultHome	Default directory used to create the login directory for a new user if no GroupHome declaration exists for the user's assigned group ID. The directory created will be $DefaultHome/*username*; *username* is the user login name.
DefaultShell	String that contains the default login shell assigned to the UDB shell field.
GroupHome	Default directory used to create the login directory for a new user in the associated group if no DefaultHome declaration exists for the user. Multiple GroupHome lines should exist for each group ID number. For example, if the declaration is GroupHome = 100 "/user1", user john (who is assigned to group 100) will have, by default, a login directory of /user1/john.
Security feature variables	Security feature parameters are used only when you turn on the security feature or use the -p option of nu.

Note: If you have configured the nu utility to skip prompting for specific UDB fields, you must use udbgen to access these fields.

If you are using the menu system, select the Configure System->NU Configuration menu and its submenus. You can import the default

configuration file by executing the `Import nu configuration` line; then modify the parameter settings and activate your changes. A sample `NU Configuration` menu screen follows:

```
Configure System
    ->NU Configuration
```

```
   NU Configuration

M-> Group home directories ==>
    Password aging ==>
    Tape Limits ==>
    Miscellaneous Limits ==>
    Ask about setting up Cray Station           1
    Login shell                                 /bin/csh
    Maximum login id length                     8
    nu log file                                 /usr/adm/nu.log
    Directory creation script                   /etc/nulib/nu1.sh
    Directory initialization script             /etc/nulib/nu2.sh
    Account removal script                      /etc/nulib/nu3.sh
    Account disabled script                     /etc/nulib/nu4.sh
    Default account ids                         0
    Default group ids                           0
    Default permbits                            none
    Default security level                      0
    Default maximum security level              0
    Default minimum security level              0
    Default default integrity class             0
    Default maximum integrity class             0
    Default valid compartment name string       none
    Default active compartment name string      none
    Default category name string                none
    Default valid category name string          none
    Default permission name string              none
    Default resource group UID                  0
    Default allocation shares                   100
    Default login root                          /
    Default home directory                      /u     Import nu
configuration ...
    Activate nu configuration ...
```

If you are not using the menu system, edit the `/etc/nu.cf60` configuration file.

Procedure 20: Creating a file system to use with `/etc/nu`

The `/etc/nu` command defaults, which are set in the `/etc/nu.cf60` file, expect a default home directory path of `/usr/home`.

To use the `/etc/nu` command , you must do **one** of the following:

- Change the `DefaultHome` parameter in the `/etc/nu.cf60` file to match what your site will use for a default `/home` path. (See Procedure 19, page 141.)

or

- Invoke the `mkdir` command to create a new directory called `home` in the `/usr` directory. This means that `home` will not be a separate file system, but just a subdirectory with files in the `/usr` file system.

or

- Create `/usr/home` as a file system and ensure that it is mounted on `/usr` when in multiuser mode.

If you are using the menu system, select the `Configure System->File System (fstab) Configuration->Standard File Systems` menu, add your entries and update the form file. Then activate your changes through the `File Systems (fstab) Configuration` menu. A sample `Standard File System Configuration` menu screen follows:

```
Configure System
    ->..File System (fstab) Configuration
        ->.......Standard File Systems
```

```
   Standard File System Configuration

    Device Name        Mount Point     FsType  Freq  Pass R/O  Quota  User
    -----------        -----------     ------  ----  ---- ---  -----  ---- >
E-> /dev/dsk/home      /usr/home       NC1FS    1     2   rw
```

If you are not using the menu system, enter the following commands to accomplish this:

1. `/bin/mkdir /usr/home`

2. `/etc/mkfs -q /dev/dsk/home`
3. `/etc/labelit /dev/dsk/home home vol1` (optional step)
4. `/etc/fsck /dev/dsk/home`
5. `/etc/mount /dev/dsk/home /usr/home`
6. To ensure that `/home` is always both checked and then mounted on `/usr` when running at multiuser level, edit the `/etc/fstab` file to check and mount `/home` automatically. Such an entry would look like the following:

```
/dev/dsk/home   /usr/home      NC1FS  rw,CRI_RC="YES"  1     2
```

   At this point, the `/etc/nu` command will work as intended.

Procedure 21: Adding a user record to /etc/udb by using /etc/nu

Note: Before you begin this procedure, make sure you have completed the following:

- Determined the UDB field settings for the user account.
- Added needed group(s) to `/etc/group` to which the user will belong.
- Added needed account(s) that will be associated with the user if you have accounting implementation on your system.
- Created a `/home` or site-specific file system for user with `/etc/nu`.

To use the `nu` utility to add a user to the `/etc/udb` file, use the following form of the `nu` utility:

```
/etc/nu  -a
```

When you use the nu utility, the `/etc/udb`, `/etc/udb.public`, `/etc/group`, `/etc/acid`, and `/etc/passwd` files are updated, or you can maintain private (testing) versions. When you maintain private versions, you can move or copy them to `/etc` to install the updates.

The `nu` utility queries you for values to UDB fields; it also lets you accept default values that have been specified in the program's configuration file (`/etc/nu.cf60`).

Note: A UDB field setting of 0 means infinite, except for tape access, where 0 means the user has no tape privileges. For more information about user account field settings, see Section 7.4, page 131.

The user's login name is something you provide when the `/etc/nu` utility prompts you. The user's login name must be a unique 1 to 8 alphanumeric representation, in which the first character is alphabetic. Typically, the name is made up of lowercase alphabetic characters.

You may want to change the UDB password aging field to `force` so that the user must change the initial password when logging in for the first time. You must remove the `off` setting for the `DefaultAge` variable in the `/etc/nu.cf60` file (either manually or by using the menu system) so that the password aging field shows up and can be set when executing the `/etc/nu` script.

The `nu` utility has other options that you might want to use when adding a user, such as the following `-p` and `-c` options:

`-p` *dirname*	Modifies UDB files found under the *dirname* directory; lets you maintain private copies or test versions.
`-c` *dirname*	Uses the configuration file, `nu.cf60`, under the *dirname* directory; lets you specify an alternative configuration file.

To determine whether a record exists for a user, use the `udbsee` command.

Example `/etc/nu` session that adds a user

The following `nu` session adds login name `jones` (**`bold`** indicates what you would type; otherwise, you can use the default values):

```
# /etc/nu -a
cmd/nu/nu.c
Login name? (1-8 characters) [quit] jones
Enter password:
Please re-enter password:
Enter actual user name: John R. Jones
User id number? (max = 60000)  [2] 624
Which groups? (names or numbers, use commas, ? for list) [0] cray,test,trng,usrsrc
```

(See procedure for adding groups)

```
You selected groups:
100 0
cray , 100
104 1
test , 104
105 2
trng , 105
98 3
usrsrc , 98
Are these correct? (y or n) [y]
Login directory?  [/hot/u1/jones]
```

(This will be the user's home directory.)

```
Enter shell [/bin/csh]
Which accounts? (names or numbers, use commas, ? for list) [0]
You selected accounts:
```

(See procedure for acids)

```
root , 0
Are these correct? (y or n) [y]
User default login root?  [/]
```

(This will be the user's root directory; set to other than / to restrict acess to file systems.)

```
Resource group ID? (name or number, ? for list) [0] Users
Which permissions? (names or numbers, use commas, ? for list) [none]
You selected permbits:
none
Are these correct? (y or n) [y]
Allocation shares? (min=0) [100]
DEFAULT security compartments? (name1,name2,... or none, ? for list)  [default]
```

```
VALID security compartments? (name1,name2,... or none, ? for list)  [default]
Security permissions? (name1,name2,... or none, ? for list)  [default] Security levels?
(default max min)  [0 0 0]
Integrity classes? (default max)  [0 0]
DEFAULT integrity categories? (name1,name2,... or none, ? for list)  [default]
VALID integrity categories? (name1,name2,... or none, ? for list)  [default]
Do you want this user locked? (y or n) [n]
Do you want this user trapped? (y or n) [n]
Per job process limit for batch? (min=0)  [100]
Per job process limit for inter? (min=0)  [100]
Per job MPP PE limit for batch?  [none]
Per job MPP PE limit for inter?  [none]
Per job MPP time limit for batch?  [none]
Per job MPP time limit for inter?  [none]
Per job MPP barrier limit for batch?  [none]
Per job MPP barrier limit for inter?  [none]
Per process MPP time limit for batch?  [none]
Per process MPP time limit for inter?  [none]
Will the user be using the Cray Station? (y or n) [y] n
```

(No for CRAY J90 systems.)

```
  1)  name ....... jones
  2)  passwd ..... v0u28k2K1wtX6 (encrypted)
  3)  pwage ........ force
  4)  comment .... John R. Jones
  5)  uid ........ 624
  6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
  7)  dir ........ /hot/u1/rnl/tmp/jones
  8)  shell ...... /bin/csh
  9)  acids ...... root (0)
 10)  root ....... /
 11)  resgrp ..... Users (102)
 12)  permbits ... none
 13)  shares ..... 100
 14)  deflvl ..... 0
 15)  maxlvl ..... 0
 16)  minlvl...... 0
 17)  defcomps ... default
 18)  valcomps ... default
 19)  permits .... default
```

```
  20)  intcls...... 0            21)  maxcls...... 0
  22)  intcat...... default      23)  valcat...... default
  24)  disabled ... 0            25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
  28)  pcpulim[b] ...      none       pcpulim[i] ...      none
  29)  jmemlim[b] ...      none       jmemlim[i] ...      none
  30)  pmemlim[b] ...      none       pmemlim[i] ...      none
  31)  pfilelim[b] ..      none       pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576       jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576       psdslim[i] ...   1048576
  34)  jtapelim[b][type0   ]      99      jtapelim[i][type0   ]     99
       jtapelim[b][type1   ]      99      jtapelim[i][type1   ]     99
       jtapelim[b][type2   ]      99      jtapelim[i][type2   ]     99
       jtapelim[b][type3   ]      99      jtapelim[i][type3   ]     99
       jtapelim[b][type4   ]      99      jtapelim[i][type4   ]     99
       jtapelim[b][type5   ]      99      jtapelim[i][type5   ]     99
       jtapelim[b][type6   ]      99      jtapelim[i][type6   ]     99
       jtapelim[b][type7   ]      99      jtapelim[i][type7   ]     99
  35)  jpelimit[b] ...      none       jpelimit[i] ...      none
  36)  jmpptime[b] ...      none       jmpptime[i] ...      none
  37)  jmppbarrier[b]       none       jmppbarrier[i]       none
  38)  pmpptime[b] ...      none       pmpptime[i] ...      none

Are these values OK? (y or n)  [y]
Entry looks like:
jones:co:John R. Jones
jones:ui:624:di:/hot/u1/rnl/tmp/jones:sh:/bin/csh:dr:/:pw:v0u28k2K1wtX6
jones:gi:100,104,105,98
jones:ai:0
jones:rg:102:as:100
jones:dc:default:cm:default:pm:default
jones:ic:default:vc:default
jones:pj[b]:100:pj[i]:100
jones:js[b]:1048576:js[i]:1048576:ps[b]:1048576:ps[i]:1048576
jones:tp:type0[b]:99:tp:type0[i]:99:tp:type1[b]:99:tp:type1[i]:99
jones:tp:type2[b]:99:tp:type2[i]:99:tp:type3[b]:99:tp:type3[i]:99
jones:tp:type4[b]:99:tp:type4[i]:99:tp:type5[b]:99:tp:type5[i]:99
jones:tp:type6[b]:99:tp:type6[i]:99:tp:type7[b]:99:tp:type7[i]:99+
test 1 -ne 0
```

```
+ rm -rf /hot/u1/jones
+ mkdir /hot/u1/jones
+ test /hot/u1/jones != /hot/u1/jones
+ test 0 -eq 0 -a 1 -ne 0
+ chgrp 100 /hot/u1/jones
+ chown 624 /hot/u1/jones
+ chacid -s root /hot/u1/jones
Do you wish to add more new users? (y or n)  [y] n
execing udbgen - please wait


udbgen complete (at this time, nu executes udbgen)
```

Procedure 22: Modifying user records by using /etc/nu

To update UDB fields, follow the same procedures as for adding new user logins, except use the `-m` option, rather than the `-a` option.

> **Note:** A UDB field setting of 0 means infinite, except for tape access, where 0 means the user has no tape privileges.

Example `/etc/nu` session that modifies a user's login:

The following example shows how to update user login entries in the UDB by using the `nu` utility. The example changes the account group (`acids`) for user `jones` (**`bold`** indicates what you would type):

```
# /etc/nu -m
cmd/nu/nu.c                >>> Modify mode <<<Enter user identifier
(login or uid) [quit]: jones
Entry is now:
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  pwage ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624
   6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/jones
   8)  shell ...... /bin/csh
   9)  acids ...... root (0)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... none
  18)  valcomps ... none
  19)  permits .... none
  20)  intcls...... 0            21)  maxcls...... 0
Press 'return' for the rest of the entry...
  22)  intcat...... none        23)  valcat...... none
  24)  disabled ... 0           25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
  28)  pcpulim[b] ...      none       pcpulim[i] ...      none
  29)  jmemlim[b] ...      none       jmemlim[i] ...      none
  30)  pmemlim[b] ...      none       pmemlim[i] ...      none
  31)  pfilelim[b] ..      none       pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576       jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576       psdslim[i] ...   1048576
  34)  jtapelim[b][type0   ]    99    jtapelim[i][type0   ]       99
       jtapelim[b][type1   ]    99    jtapelim[i][type1   ]       99
       jtapelim[b][type2   ]    99    jtapelim[i][type2   ]       99
       jtapelim[b][type3   ]    99    jtapelim[i][type3   ]       99
       jtapelim[b][type4   ]    99    jtapelim[i][type4   ]       99
       jtapelim[b][type5   ]    99    jtapelim[i][type5   ]       99
       jtapelim[b][type6   ]    99    jtapelim[i][type6   ]       99
       jtapelim[b][type7   ]    99    jtapelim[i][type7   ]       99
```

```
  35)  jpelimit[b] ...      none       jpelimit[i] ...      none
  36)  jmpptime[b] ...      none       jmpptime[i] ...      none
  37)  jmppbarrier[b]       none       jmppbarrier[i]       none
  38)  pmpptime[b] ...      none       pmpptime[i] ...      none
Select
field to be modified (1-38, q (discard changes), or e (make changes)):
9
Which accounts? (names or numbers, use commas, ? for list) [0] jones
You selected accounts:
jones , 624
Are these correct? (y or n) [y] Entry is now:
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  age ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624    6)  gids ....... cray (100)  test (104)
trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/jones
   8)  shell ...... /bin/csh
   9)  acids ...... jones (624)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... none
  18)  valcomps ... none
  19)  permits .... none
  20)  intcls...... 0           21)  maxcls...... 0
  22)  intcat...... none       23)  valcat...... none
  24)  disabled ... 0          25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
  28)  pcpulim[b] ...      none       pcpulim[i] ...      none
  29)  jmemlim[b] ...      none       jmemlim[i] ...      none
  30)  pmemlim[b] ...      none       pmemlim[i] ...      none
  31)  pfilelim[b] ..      none       pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576       jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576       psdslim[i] ...   1048576
```

```
  34)  jtapelim[b][type0   ]      99     jtapelim[i][type0   ]      99
       jtapelim[b][type1   ]      99     jtapelim[i][type1   ]      99
       jtapelim[b][type2   ]      99     jtapelim[i][type2   ]      99
       jtapelim[b][type3   ]      99     jtapelim[i][type3   ]      99
       jtapelim[b][type4   ]      99     jtapelim[i][type4   ]      99
       jtapelim[b][type5   ]      99     jtapelim[i][type5   ]      99
       jtapelim[b][type6   ]      99     jtapelim[i][type6   ]      99
       jtapelim[b][type7   ]      99     jtapelim[i][type7   ]      99
  35)  jpelimit[b] ...      none       jpelimit[i] ...      none
  36)  jmpptime[b] ...      none       jmpptime[i] ...      none
  37)  jmppbarrier[b]       none       jmppbarrier[i]       none
  38)  pmpptime[b] ...      none       pmpptime[i] ...      noneSelect
field to be modified (1-38, q (discard changes), or e (make
changes)): e

Do you want to modify any more ./udb entries? (y or n) [y]n
done.
execing udbgen - please wait
udbgen complete.
```

Procedure 23: Deleting a user record by using `/etc/nu`

Caution: Deleting a user from the system requires more prudence than adding a user to the system because you may be removing valuable data from the system. Before removing the user from the UDB, you should determine whether any pertinent files are needed from the account. If files are needed, you can disable the user account by setting the `PERMBITS_RESTRICTED` bit in `permbits` to prevent the user from running. Setting `PERMBITS_NOBATCH` prevents batch jobs from running, and setting `PERMBITS_NOIACTIVE` prevents interactive jobs from running.

To remove a user account completely, follow these basic steps:

1. Save any important files the user owned on the system. You may want to back up these files to tape or have someone in the deleted user's department copy necessary files to another directory.

Example:

```
# rsv
# tpmnt -l nl -p /tmp/tapedev -v vsn -b 4096
# ls -a /usr/trng/jones | cpio -o > /tmp/tapedev
```

```
# rls -a
```

2. Disable the user's entry from `/etc/udb` by using the `/etc/nu -d` command, as follows:

   ```
   # /etc/nu -d
   ```

You will be prompted to enter the login name or UID of the user you want to disable.

Note: If you want to keep accounting records in order or if you want to ensure that the user ID is not reused, you may choose not to complete step 3.

3. Remove the user from the UDB files by using the `/etc/nu -k` command, as follows. This command removes files under the user's login directory and that directory removes the user's mailbox and accounting records:

   ```
   # /etc/nu -k jones
   ```

Example 6: `/etc/nu` session that disables and removes a user's login

The following is an example `/etc/nu` session that disables and then removes user `jones` from the system:

```
# /etc/nu -d
cmd/nu/nu.c                >>> Deletion mode <<<Enter user identifier
(login or uid) [quit]: jones
Entry is now:
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  pwage ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624
   6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/jones
   8)  shell ...... /bin/csh
   9)  acids ...... jones (624)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... none
  18)  valcomps ... none
  19)  permits .... none
  20)  intcls...... 0            21)  maxcls...... 0
  22)  intcat...... none           23)  valcat...... none
  24)  disabled ... 0            25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
  28)  pcpulim[b] ...      none       pcpulim[i] ...      none
  29)  jmemlim[b] ...      none       jmemlim[i] ...      none
  30)  pmemlim[b] ...      none       pmemlim[i] ...      none
  31)  pfilelim[b] ..      none       pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576       jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576       psdslim[i] ...   1048576
  34)  jtapelim[b][type0  ]     99    jtapelim[i][type0   ]        99
       jtapelim[b][type1  ]     99    jtapelim[i][type1   ]        99
       jtapelim[b][type2  ]     99    jtapelim[i][type2   ]        99
       jtapelim[b][type3  ]     99    jtapelim[i][type3   ]        99
       jtapelim[b][type4  ]     99    jtapelim[i][type4   ]        99
       jtapelim[b][type5  ]     99    jtapelim[i][type5   ]        99
       jtapelim[b][type6  ]     99    jtapelim[i][type6   ]        99
       jtapelim[b][type7  ]     99    jtapelim[i][type7   ]        99
```

```
  35)  jpelimit[b] ...      none       jpelimit[i] ...      none
  36)  jmpptime[b] ...      none       jmpptime[i] ...      none
  37)  jmppbarrier[b]       none       jmppbarrier[i]       none
  38)  pmpptime[b] ...      none       pmpptime[i] ...      none
Do you want to delete this entry? (y or n) [y] y

Entry for user jones has been deleted.
Do you want to delete any more users? (y or n) [y] n
execing udbgen - please wait
udbgen complete.

# nu -k jonescmd/nu/nu.c     71.7    10/30/92 09:04:35 (hot:./nu.cf60)

User jones is already disabled; no directory deletion done.

Entry for user jones has been killed.
execing udbgen - please wait.
udbgen complete.
#
```

7.6 Using /etc/udbgen

The /etc/udbgen utility lets you make changes to the UDB, either interactively or as a batch job. The /etc/udbgen utility is actually the program underlying the /etc/nu utility. The batch capability of /etc/udbgen lets you add or modify many accounts at one time. When used with the udbsee command, the udbgen command with its directives can be a powerful and efficient tool in maintaining many accounts.

> **Note:** If you have configured the nu utility to skip prompting for specific UDB fields, you must use udbgen to access these fields.

When using the /etc/udbgen command to create a new user login, you must specify the create directive. You may use the /etc/udbgen program in the following three ways:

- Interactive submission: When using udbgen interactively, you must use the quit directive to exit the utility; this action updates the UDB files.
- Batch submission: You can place directives in a file and submit them all at once to the UDB.
- Individual submission: You can submit directives individually.

With all three methods, you can enter multiple UDB field names and their values. You can place more than one field on a `create` directive line or each field may be on separate lines. The recommended method for `udbgen` is the batch approach: place the directives in a file and then use that file as input to the `/etc/udbgen` command.

You may choose, for test purposes, to modify a private copy of the UDB files, rather than the ones contained in the `/etc` directory; see Example 7, page 152.

The format of the `create` directive is as follows:

`create:`*user_name*`:uid:`*uid_number*`:gids:`*group_names*`:field_name:`*field_value*`:`

Field	Description
`create:`	Adds the specified user's information to the UDB; if a UDB record already exists for this user name, a warning message is displayed and the record is not changed.
user_name `:`	User login name to be created; must be a unique name within the database.
`uid:` *uid_number* `:`	Specifies a `uid` value or `next` to have `udbgen` assign a value.
`gids:` *group_names* `:`	Specifies one or more group IDs or names.
`field_name:` *field* `_` *value* `:`	One or more field values; you can put multiple field values on one line, or you can put each field on a separate line.

Note: You **must** include the colon at the end of the directive.

The remainder of this subsection provides the following information:

- Procedure for adding users by using the `/etc/udbgen` utility
- Procedure for transferring initial files to the login directory when using the `/etc/udbgen` utility

- Procedure for updating user logins in the UDB by using the `/etc/udbgen` utility
- Procedure for deleting users from the UDB by using the `/etc/udbgen` utility

Procedure 24: Adding users to `/etc/udb` by using `/etc/udbgen`

Note: Before you begin this procedure, make sure you have completed the following:

- Determined the UDB field settings for the user account.
- Added needed group(s) to `/etc/group` to which the user will belong.
- Added needed account(s) that will be associated with the user if you have accounting implementation on your system.

For details on these procedures, see Procedure 21, page 144. You also may choose, for test purposes, to modify a private copy of the UDB files, rather than the ones contained in the `/etc` directory; to do this, see the example at the end of this procedure.

1. Complete **one** of the following three methods of using `/etc/udbgen` to add a user to your system:

   - Create a file of `/etc/udbgen` directives that has this format; you must include the colon at the end of the line:

`create:`*username*`:uid:`*uid_number*`:gids:`*group_names*`:field_name:`*field_value*`:field_name:\n`
field_value:`field_name:`*field_value*`:`

   Then submit the changes to the `/etc/udbgen` database by entering the following command line:

```
# /etc/udbgen udbgen_directives_filename
```

Example of directives submitted in a batch file (**`bold`** indicates what you type):

```
# vi udb.source
(Enterudbgen directives.)
# cat udb.source
create:john:uid:next:
comment:John Smith:
pwage:force:
gids:cray,test,trng:
acids:testing,training:
dir:/user1/trng/john:
shell:/bin/csh:
resgrp:102:
psdslim[b]:1000000:
pmemlim[i]:8000:
psdslim[i]:1000000:
shares:100:
# udbgen udb.sourceInput style: udb
Added 1 record
#
```

or

- Type `/etc/udbgen` to enter interactive mode and reply to the prompt that has the following format; you must include the colon at the end of the line:

  `create:`*user_name*`:uid:`*uid_number*`:gids:`*group_names*`:field_name:`*field_value*`:quit`

or

- If you must make only one or two changes, you can submit directives to the database individually by typing a line that has the following format:

  `/etc/udbgen -c "create:`*user_name*`:uid:`*uid_number*`:field_name:`*field_value*`:"`

Example of directives submitted individually (**bold** indicates what you would type):

```
# /etc/udbgen -c "create:john:shell:/bin/sh:uid:next:gids:cray,test,trng:"
# /etc/udbgen -c "update:john:resgrp:102:"
# /etc/udbgen -c "delete:mary:"
```

2. Verify that your entry was added by using the `udbsee`(1) command.
3. Assign the initial password for the user.

If you use the `udbgen` command, you cannot set the password field to an initial password. Instead, you must use the `/bin/passwd` command to change the

password. You and the new user must agree on an initial password for the account. Choose one that is not easy to guess. Only the super user may change another user's password. You may have chosen to set the UDB `passwd` field to * or left it empty (indicated by two contiguous colons, ::). If no assignment was made to this field during the user's login creation, the field is assigned the * symbol.

Caution: If you have left the password field empty, anyone who knows the login can use this account. Your system is open to abuse.

Example:

```
# /etc/udbgen  -c
   "create:john:uid:next:gids:cray,test,trng:" #
   /bin/passwd john New password:
```

(The password is not visible on your screen.)

```
Reenter new password: #
```

Create a login directory for the user.

The `/etc/udbgen` command does not automatically create a login (`home`) directory. The `dir` for each entry in `/etc/udb` specifies the initial working directory (`home`) for each user at login time. As the system administrator, you must create that directory by using `/bin/mkdir`. Because you currently have a user ID of 0 and a group ID of 0, the directory created also will be assigned these permissions. You must make the user's directory accessible to the user by changing the permissions, group, and ownership of the directory. This involves executing the `chmod`(1), `chgrp`(1), and `chown`(1) commands.

The following is a brief review of how UNICOS permissions are defined (see for examples).

Format:

```
/etc/chmod permissions filename
```

Permissions are set up in three groups, and they can be displayed by using the `ls -l` command:

	User	Group	Other
-	rwx	rwx	rwx
d	rwx	rwx	rwx

The - symbol indicates that the file is a regular file. The d indicates that the file is a directory file. The r, w, and x indicate permissions for read, write, and execute, respectively. If the r, w, or x is present, that permission is set for that category of users (user, group, or other). If a - symbol is in any of the fields, except for the first field, that permission is turned off for that category of users. You can represent these fields numerically, as follows:

400, 40, and 4	Readable by user, group, and other, respectively.
200, 20, and 2	Writable by user, group, and other, respectively.
100, 10, and 1	Executable by user, group, and other, respectively.

Example:

To give user, group, and others read, write, and execute permissions, calculate the permission fields to use with the `chmod` command:

Also see the following examples.

Example of creating a login directory:

1. Create the directory by using the `/bin/mkdir` command.

   Format: `/bin/mkdir` *new_login_directory_name*

   Example: `# mkdir /user1/trng/jones`

2. Change the ownership of the directory by using the `/bin/chown` command.

   Format: `/bin/chown` *new_login_name new_login_directory_name*

   Example: `# /bin/chown jon /user1/trng/jones`

3. Change the group of the directory by using the `/bin/chgrp` command.

   Format: `/bin/chgrp` *new_group new_login_directory_name*

   Example: `# /bin/chgrp swtng /user1/trng/jones`

4. Change the permissions of the directory by using the `/bin/chmod` command.

   Format: `/bin/chmod` *permissions new_login_directory_name*

Example: # `/bin/chmod 761 /user1/trng/jones`

Note: If you want to move existing files into the login directory, use the procedure to transfer initial files to the login directory (see Procedure 25, page 163).

Examples of `/bin/chown`, `/bin/chgrp`, and `/bin/chmod` follow:

```
sn1601% pwd
/sn1601/sdiv/unicos/jones%
su root
Password:
# ls -la
total 21
drwx------   3 jones   os           4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root         4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-r--r--r--   1 root    root         192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt

# /bin/chown jones mnt
# ls -la
total 21
drwx------   3 jones   os           4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root         4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-r--r--r--   1 jones   root         192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt

# /bin/chgrp tng mnt
# ls -la
total 21
drwx------   3 jones   os           4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root         4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-r--r--r--   1 jones   tng          192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt

# /bin/chmod 761 mnt
# ls -la
total 21
drwx------   3 jones   os           4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root         4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-rwxrw---x   1 jones   tng          192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt
```

Example 7: Example of using a private copy of UDB files for test purposes:

You may choose, for test purposes, to modify a private copy of the UDB files, rather than the ones contained in the `/etc` directory. After you have set up a private UDB, use the `-p` option with the `/etc/udbgen` command, as follows:

1. Create a directory to contain your private UDB, as follows:

   # **mkdir/myudb**

2. Create a `group` file in your new directory, as follows:

   # **cp /etc/group ./myudb/group**

3. Create an `acid` file in your new directory, as follows:

   # **cp /etc/acid ./myudb/acid**

To verify that the user login was created correctly, use the `udbsee` command. Then move or copy the UDB files contained in the directory specified by the `-p` option into the `/etc` directory, as shown in the following example.

Example of directives submitted interactively (**bold** indicates what you would type):

```
# /etc/udbgen -p /user1/jones/etc
/etc/udbgen: 1>create:john:uid:next:
/etc/udbgen: 2>comment:John Smith:
/etc/udbgen: 3>pwage:force:
/etc/udbgen: 4>gids:cray,test,trng:
/etc/udbgen: 5>acids:testing,training:
/etc/udbgen: 6>dir:/user1/trng/john:
/etc/udbgen: 7>shell:/bin/csh:
/etc/udbgen: 8>resgrp:102:
/etc/udbgen: 9>psdslim[b]:1000000:
/etc/udbgen: 10>pmemlim[i]:8000:
/etc/udbgen: 11>shares:100:
/etc/udbgen: 12>quit
Added 1 record
```

Procedure 25: Transferring initial files to the login directory when using `/etc/udbgen`

To transfer initial files to the login directory when using `/etc/udbgen`, follow these steps:

1. You may want to create a directory, such as `/usr/skel`, to hold templates for such files as `.profile`, `.cshrc`, `.login`, and `.exrc`. The `/etc/udbgen` command does not automatically copy skeleton files to the user's `home` directory.

   For descriptions of how to set up the `/etc/profile` and `/etc/cshrc` files, see Section 7.7, page 167.

2. After you have created `/usr/skel` and the template files, copy the desired files to the user's `home` directory by using the `cpset` command, which lets you specify the mode, owner, and group of the destination files. `cpset` installs object files in binary directories.

Example:

```
# cpset /usr/skel/.cshrc /usr/home/john 700 john trng
# cpset /usr/skel/.login /usr/home/john 700 john trng
```

Procedure 26: Updating user logins in the UDB by using /etc/udbgen

The method for updating user logins is similar to that for adding new user logins. Different directives are used, however, to accomplish the task. Some of the fields (such as the `gids` field) have editing suffixes that may be used with the `/etc/udbgen` command. These editing suffixes are as follows:

=	Indicates that the next value(s) replace the existing value.
+	Indicates that the following values will be appended to the current values of the field (see example 1).
-	Indicates that the following values will be deleted from the list of current values for the field.

You may use the following steps to change every field except the `passwd` field. To change the `passwd` field, use the `/bin/passwd` command, as described in step 3 of the procedure for adding users to `/etc/udb` by using `/etc/udbgen` (see the `passwd` man page for further information). You change the user login name by deleting the old user login and creating a new user login under the new name.

Caution: It is a **very dangerous** practice to delete the user login of a user who may be logged in on the system. This procedure should wait until a time when you know the user is not running anything on the system.

The following steps summarize the user login update process:

1. Decide which UDB fields you want to change.

2. If the user will be placed in a new group that you will reference by name, make the desired entry in `/etc/group`.
3. If the user will be placed in a new account group that you will reference by name, make the desired entry in `/etc/acid`.
4. Make the desired entry in `/etc/udb` by using the `/etc/udbgen -c` command with the `update` directive.If you change the user's login directory, create the new directory and copy over any existing files to the new directory.

The following examples show how to update user login entries in the UDB by using the `/etc/udbgen` command.

Example 8: Adding a new group ID

This example adds the new group ID (`gids`) `usrsrc` to user `john`:

```
# /etc/udbgen -c "update:john:gids+:usrsrc:"
```

Example 9: Changing the user's shell

This example changes the login shell for user `john` to the POSIX shell. Because the POSIX shell was specified, you also must create a `.profile` file.

```
# /etc/udbgen -c "update:john:shell:/bin/sh:"
# cpset /usr/skel/.profile /user1/trng/john
```

Example 10: Changing the user's login directory

This example changes the login directory for user `john` to `/usr1/john`. Formerly, user `john` 's login directory was `/user1/trng/john`. The `mkdir`, `chown`, `chgrp`, and `chmod` commands are used to create the `/usr1/john` directory and to assign proper ownership and permissions for the directory. The last three commands remove `john` 's old login directory.

Caution: If the user is running anything on the system, you should never change a home directory. This is especially critical if libraries are removed.

```
# /etc/udbgen -c "update:john:dir:/user1/john:"
# mkdir /user1/john
# chown john /user1/john
# chgrp trng /user1/john
# chmod 700 /user1/john
# cd /user1/trng/john
# find . -print | cpio -pdm /user1/john
```

```
# rm -rf /user1/trng/john
```

Example 11: Using the udbsee command as a filter to add an account ID (acid)

This example uses the `udbsee` and `udbgen` commands to add an account ID (`acid`) of 10 to all user accounts that have a group ID (`gid`) of 103. In all, 46 login accounts are affected. This is a typical example of how a large-scale update to the UDB is performed:

```
# udbsee -a -e 'gids ~ "103"' -f "name" -m 'update:%s:acids+:10:\n'|/etc/udbgen
46 entries converted to source
Input style: udb
Updated 46 records
#
```

Example 12: Changing the user's password

This example uses the `/bin/passwd` command to change the password for user `john`:

```
# /bin/passwd john
New password:
```

(The password is not visible on your screen.)

```
Reenter new password:
```

Procedure 27: Deleting a user from the UDB by using /etc/udbgen

Caution: Deleting a user from the system requires more prudence than adding a user to the system, because you may remove valuable data from the system. Before removing a user from the UDB, you should determine whether any pertinent files are needed from the account. If files are needed, you can disable the user account by setting the *passwd* field to `*`, using the `udbgen` command.

It is a **very dangerous** practice to delete users who may be logged in. This procedure should wait until a time when you know the user is not running anything on the system.

To remove a user account completely, perform the following basic steps:

1. Make it impossible for the user to log in by using the `udbgen` command, as follows:

   ```
   # /etc/udbgen -c "update:john:passwd:*:"
   ```

2. Ensure that the user has nothing running on the system.

3. Save any important files the user owned on the system. You may want to back up these files to tape or have someone in the deleted user's department copy necessary files to another directory.

Example:

```
# rsv
# tpmnt -l nl -p /tmp/tapedev -v vsn -b 4096
# ls -a /usr/trng/john | cpio -o > /tmp/tapedev
# rls -a
```

4. Delete files from the user's home directory and any other directories on the system by using multiple `rm` commands, and remove the user's `home` directory. Also remove the user's mailbox, `/usr/mail/` *username*.

Example:

```
# rm -rf /user1/trng/john
# find  /  -user  john  -exec  rm  {}\;
# rm  -f  /usr/mail/john
```

Note: If you want to keep accounting records in order or if you want to ensure that the user ID is not reused, you may choose not to complete the previous step.

Remove the user from the UDB files by using the `delete` directive of the `udbgen` command. The `delete` directive has the `delete:` *userid*: format; you must include the colon at the end of the directive:

```
# /etc/udbgen -c "delete:john:"
```

7.7 Maintaining user environment files

This subsection describes the following procedures:

- Setting up an `/etc/profile` file
- Setting up an `/etc/cshrc` file

- Transferring users to another file system

When the user logs in to the system, the `/bin/login` script executes the program in the UDB `shell` field. If you specify `/bin/sh` or `/bin/rsh`, the following files are executed (if they exist) by `/bin/sh` or by `/bin/rsh`:

```
/etc/profile
$HOME/.profile
```

If you specify `/bin/csh`, the program executes the following files in the order shown:

```
/etc/cshrc
$HOME/.cshrc
$HOME/.login
```

As the administrator, you must maintain the `/etc/profile` and `/etc/cshrc` files, which are described in this subsection.

Procedure 28: Setting up an `/etc/profile` file

When the `/bin/login` script invokes the default shell (`/bin/sh`, which is the POSIX shell, or `/bin/rsh`, which is the restricted shell), it reads and executes the commands found in the `/etc/profile` file. This lets you set up a standard environment for all users. Users may alter this setup environment through the `$HOME/.profile` file to personalize their environment.

> **Note:** If you want to change the `.profile` file, see the `sh`(1) man page, which describes the supported shells and the `shell` script syntax.

A typical system profile, `/etc/profile`, might perform the following functions (the line references refer to the example that follows):

1. Set and export the directory search path (lines 4 and 5).
2. Set the file creation mask, using the `umask` command (line 6).
3. If using one of these shells (line 8), do the following functions:
   - Display the message of the day (line10).
   - If the `.motd` file exists, display it (lines 11 through 13).
   - Check for the existence of mail (lines 15 through 17).
   - Display the names of current news items (lines 18 through 20).
   - Set the user's prompt (lines 21 through 25).

- Set effective user ID if user uses the `/bin/su` command (line 27).

An example `/etc/profile` file follows:

```
1#       SCMID@(#) /etc/profile
 2
 3 trap "" 1 2 3
 4 PATH=/bin:/usr/bin:/usr/ucb:/usr/lbin
 5 export PATH
 6 umask 022
 7 case "$0" in
 8 -sh | -rsh | -ksh)
 9      trap : 1 2 3
10      cat /etc/motd
11      if [ -f ../.motd ] ; then
12              cat ../.motd
13      fi
14      trap "" 1 2 3
15      if mail -e ; then
16           echo "You have mail."
17      fi
18      if [ -x /usr/bin/news -a -d /usr/news ] ; then
19              news -n
20      fi
21      if id | grep 'uid=0' > /dev/null ; then
22              PS1="`uname -n`# "
23      else
24              PS1="`uname -n`$ "
25      fi
26      ;;
27  -su)
28      :
29      ;;
30  esac
31  trap 1 2 3
```

Procedure 29: Setting up an `/etc/cshrc` file

If a user has chosen to run the C shell (`/bin/csh`), the commands found in `/etc/cshrc` are executed. You should set up the same environment variables found in `/etc/profile` in the C shell start-up file.

Note: If you want to change the `.profile` file, see the `csh`(1) man page, which describes the `shell` command syntax.

Users can alter this setup environment through the `$HOME/.cshrc` and `$HOME/.login` files to personalize their environment. A typical system profile, `/etc/cshrc`, might perform the following functions (the line references refer to the example that follows):

1. Set and export the shell variable (line 3).
2. Set and export the directory search path (line 4).
3. Start up the history mechanism (line 5).
4. Set the file creation mask, using the `umask` command (line 6).
5. Display the message of the day (line 7).
6. Check for the existence of mail (lines 8 and 9).
7. Display the names of current news items (lines 10 through 12).
8. Set the user's prompt (line 13).

An example `/etc/cshrc` file follows:

```
1  # SCMID@(#) etc/cshrc
 2
 3  setenv SHELL /bin/csh
 4  set path = ( /bin /usr/bin /usr/ucb /usr/lbin /usr/ucb)
 5  set history = 24
 6  umask 022
 7  cat /etc/motd
 8  mail -e
 9  if ( $status == 0 ) echo "You have mail."
10  if (-d /usr/news) then
11    news -n
12  endif
13  set prompt = "'uname -n‘$prompt"
```

Procedure 30: Transferring user accounts to another file system

If a group of users must be transferred to another file system, use the `cpio` command to copy them. If all users are copied at the same time, the `cpio` command helps preserve any links the users had among their files.

Caution: Be sure to update the user's home directory in the UDB. When you do this, also ensure that none of the users are running anything on the system.

Example:

```
# cd /user_a
# find john tom sue mike alice -print | cpio -pdm /user_b
# rm -rf john tom sue mike alice
```

Communicating with Users [8]

As a system administrator, you must communicate with your users frequently. Several methods of communication are available for you to use. The method to use in any specific instance generally is determined by the urgency of your message.

The following list describes the types of communication you will maintain with users, as well as the commands associated with that kind of communication:

Type of communication	Command or file
Issuing emergency messages only	`/etc/wall`
Issuing critical messages	`/etc/issue`
Issuing special messages (message of the day)	`/etc/motd`
Issuing normal (noncritical) communication to all users	`/usr/news`
Communicating with specific users	`write` and `mail`

This chapter describes when you should use each type of communication and gives examples of each.

8.1 Related user communication documentation

The following documentation contains detailed information covered in this chapter:

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `mail`(1), `news`(1), `su`(1), `wall`(1), and `write`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `issue`(5) and `motd`(5) man pages

8.2 Communicating with users

During the operation of a UNICOS system, it is frequently necessary for administrators to use the system to communicate information to its users. This

section discusses a number of UNICOS commands and tools that enable you to communicate with users:

- The `wall`(8) command
- The `/etc/motd` file
- The `/etc/issue` file
- The `/usr/news` directory
- The `write`(1) utility
- The `mail`(1) utility

8.2.1 The `wall` command

The `wall`(8) command broadcasts items of immediate concern to all users currently logged in to the system. Run the command by typing the following:

```
/etc/wall
```

The `wall` command responds by telling you to type your message and to press `CONTROL-d` when you are finished. To ensure that all users who are currently logged in see a message sent by `wall`, run the command while you have `root` privileges; otherwise, the message goes only to users who allow messages to be written to their terminals (see `mesg`(1)). Additionally, users who are not currently logged in will never see the message; `wall` is thus not a suitable method for communicating a message to all users who have accounts on the system.

The `wall` command is typically used to send the following messages:

- Warnings that the system will soon be brought down for scheduled downtime. Users who log in after the message is sent, however, miss the message and should be notified by the `/etc/issue` file (see `login`(1)).
- Warnings that the system must be brought down immediately to address a system emergency.
- Warnings that a particular file system has run out of disk space and that users should make an immediate effort to delete any unneeded files (see the description of the `-g` option on the `wall`(8) man page).

8.2.2 The /etc/motd file

The /etc/motd (message-of-the-day) file is displayed to users after they are logged in to the system. The /etc/motd file is an ordinary text file, and the administrator may place messages in it by using any UNICOS text editor.

Messages that should be placed in /etc/motd are those that are less immediate than those requiring the use of wall(8), but they are important enough that users should be forced to see them. The administrator should remove messages from /etc/motd as soon as they are no longer needed. Suitable items for inclusion in this file include the following:

- Warnings to users to clean up unnecessary files on a particular file system or systems
- Brief explanations of recent problems that may have affected a number of users, often with a pointer to a news item containing a more detailed explanation

8.2.3 The /etc/issue file

The /etc/issue file is displayed while a user is logging in, before the user has successfully logged in to the system. It is an ordinary text file, and you may place messages in it by using any UNICOS text editor.

Messages placed in /etc/issue should be brief and so important that users may need the information to decide whether or not to log in to the system. Possible messages include the following:

- Warnings that the system will be brought down soon (so that users who do not see a wall(8) message are not surprised when the system is brought down shortly after they log in)
- Warnings that the system is being used for dedicated time and that not all users will be able to log in

8.2.4 The /usr/news directory

When users log in to the system they are alerted to the existence of any new files placed in the /usr/news directory. When a user then runs the news(1) utility, it displays any news files that have been created or modified since the last time the user ran news. The files placed in /usr/news are ordinary text files created with any UNICOS text editor, and they are usually assigned names that give a general idea as to their contents. For instance, a news file containing

information about a modification to a system library might be given the name `new.library`.

Because users are not notified of the existence of a new news file until the next time they log in, and because there is no guarantee that any given user will see the file (a user may choose to ignore the item by not running the `news` utility), `/usr/news` is appropriate for items that are not time-sensitive or items that are of interest to only some of the system's users. These categories include the following:

- Notices regarding recent system changes, such as a newly installed version of a command or library
- Explanations of imminent system reconfigurations or changes
- Explanations of recent system problems and their possible effects on users

It is a good idea to remove any old files in `/usr/news` periodically, not only to save disk space, but also to prevent new users on the system from having to read through a long list of out-of-date news items. The `/usr/news` file may be cleaned out regularly by `cron`(8).

8.2.5 The `write` utility

The `write`(1) utility initiates immediate person-to-person communication with a logged-in user by opening that user's tty or pty for writing and copying each line of text you type to his or her screen. To write to a user with a login name of `dolores`, for example, you would issue the following command:

```
write dolores
```

If the user `dolores` happened to be logged in on more than one tty or pty, you could specify the connection:

```
write dolores ttyp001
```

If, in this example, the user `dolores` is currently logged in, a message appears on her screen indicating that you are writing to her. Typically, the user `dolores` replies by writing back to your account; each line of text she types appears on your screen.

Given the immediate nature of its communication, the `write` utility allows you to perform the following functions:

- Converse with a user

- Obtain information about what a user is doing
- Warn a specific user to stop what he or she is doing
- Instruct a specific user to clean up his or her directories

Because each typed line appears on the other user's terminal without regard for what that person may be typing at the moment, it is easy for the other user's messages to your terminal to appear to interfere with your typing. This problem is customarily solved by having the two users take turns typing, ending a message with an `o` on a line by itself (standing for "over," much as in a two-way radio conversation). To end such a session, either user then ends a message with an `oo` on a line by itself (for "over and out"). Thus, a typical "conversation" carried out by `write` might look like this (your input appears in **bold**):

```
# write dolores
Message from dolores (ttyp001) - Mon May 11 08:20:15 - ...
Yes
o
Please clean up your account, we're out of space.
o
All right, I will.
o
Thank you.
oo
<EOT>
```

Because many users either do not know of this etiquette when using `write`, or do not follow it, they think that `write` is difficult to use. In practice, it is used rather sparingly, mainly when more convenient forms of communication (such as simply calling the user on the telephone) are impossible. Taking steps to educate your user community in the proper use of the `write` utility will prove valuable when `write` is the appropriate communication method.

Note: On a UNICOS system or Cray ML-Safe configuration, for `write` to execute properly, the user's active security labels must be equal.

8.2.6 The `mail` utility

The `mail`(1) utility provides a way to leave messages for specific users, whether or not they are currently logged in to the system. The `mail` utility is used as follows:

```
mail ralph
```

Type in message

```
CONTROL-d
```

You may specify more than one account name, in which case copies of the message go to each user named. The next time users to whom you (or anyone else) have sent mail messages log in to the system, the system alerts them to the fact that they have mail messages waiting. The `mail` utility is thus particularly well suited for messages such as the following:

- Instructions to clean up directories
- Asking or responding to questions
- General communication

In theory, there is no guarantee that the recipient of a mail message will actually see the message, because the recipient may choose not to run the `mail` utility to read the message; however, in practice, most users read their mail when they log in.

> **Note:** On a UNICOS system or Cray ML-Safe configuration, the recipient of a mail message might not be authorized to read mail at the classification with which it was sent.

For more information see `mail`(1) and `mailx`(1).

Log Files [9]

This section describes several log files that are important for you to monitor. Information found in these files can help you determine appropriate actions. You can access the logs described in this section through normal file manipulation commands such as `tail`(1), `cat`(1), `pg`(1), and `more`(1).

Note: For information on the SWS/ION message logs, see *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204.

This section describes the following:

- The `/etc/boot.log` file
- The `/etc/rc.log` file
- The `/etc/syslog.conf` file and the `syslog` daemon, `/etc/syslogd`, which works with the `/etc/syslog.conf` file to record entries into the following system log files:
  - `/usr/adm/sulog`
  - `/etc/dump.log`
  - `/usr/adm/nu.log`
  - `/usr/adm/sa/sa`*DD*
  - `/usr/adm/sl/slogfile`
  - `/usr/spool/msg/msglog.log`
  - `/usr/lib/cron/cronlog`
  - `/usr/tmp/nqs.log`
  - `/usr/adm/errfile`
  - `/usr/spool/dm/*`
- Cleaning up system logs

For information about accounting logs and reports, see Chapter 10, page 195.

9.1 Related log files documentation

The following documentation contains information that you will find useful in understanding the material presented in this section:

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `brc`(8), `cron`(8), `newsys`(8), `reduce`(8), `sar`(8), and `syslogd`(8) man pages
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `at`(1), `batch`(1), `cat`(1), `crontab`(1), `date`(1), `logger`(1), `more`(1), `sar`(1), `tail`(1), and `uname`(1) man pages

9.2 `/etc/boot.log` file

The `/etc/boot.log` file records boot dates and times for a system. When the `/etc/rc` script is executed, it appends a record to the `/etc/boot.log` file. The file is composed of output from the `/bin/date` and `uname -a` commands. The format of the `/etc/boot.log` file includes the system name, node, release, version, and hardware information. To determine the last time a system was booted, see this log. The format is as follows:

`date, uname -a` *yy/mm/dd hh:mm system node release version hardware*

Example:

```
# cat /etc/boot.log
93/09/10 08:07 sn1703c cool 9.3 CRAY J90se
```

For further information, see the `date`(1) and `uname`(1) man pages.

9.3 `/etc/rc.log` file

The `/etc/rc.log` file records the events that occurred the last time the `/etc/rc` (multiuser startup) script was run.

9.4 `/etc/syslog.conf` file

The `syslog` configuration file, `/etc/syslog.conf`, defines the messages that are processed and where they are recorded. An example of the

/etc/syslog.conf file follows (for a description of the fields, see the syslogd(8) and syslog(3) man pages):

```
# (Messages processed)                          (Stored location)
#
*.debug                                         /usr/adm/syslog/debug
#
mail.debug                                      /usr/spool/mqueue/syslog
#
kern.debug                                      /usr/adm/syslog/kern
#
daemon,auth.debug                               /usr/adm/syslog/auth
#
*.err;kern.debug;daemon,auth.notice;            /usr/adm/syslog/daylog
#
*.alert;kern.err;daemon.err                     operator
#
*.alert                                         root
```

9.5 System logs

The syslog daemon , /etc/syslogd, provides UNICOS with the ability to route messages to regular disk files or to forward them to mail accounts. The /etc/syslogd daemon reads and logs messages into a set of files specified by the administrator in the /etc/syslog.conf configuration file. /etc/syslogd configures itself at start-up time and when it encounters a hang-up signal. The /etc/newsys shell script starts it.

The /usr/ucb/logger command places entries into the system log. For example, if you restart a daemon in the middle of the day, you can log this event by using the following command:

```
# /usr/ucb/logger -p user.info restarted development copy of syslog daemon
```

This subsection includes information about the following topics:

- Message sources
- Priority levels
- syslog daemon startup
- System log files

9.5.1 Message sources

Messages may be given to the `syslog` daemon, `/etc/syslogd`, from the following sources or facilities:

Source/ Facility	Description
`auth`	Messages that the authorization system (that is, `login`, `su`, or `getty`) generates.
`daemon`	Messages that system daemons (such as `telnetd`, `ftpd`, and `errdaemon`) generate.
`kern`	Messages that the kernel generates. The daemon reads kernel messages from the `/dev/klog` device.
`local0`	Reserved for local use (`local0` through `local7` are available).
`mail`	Messages that the mail system generates.
`mark`	Informational-level messages are sent; default interval is every 20 minutes (set by the `syslogd -m` command).
`user`	Messages that user processes generate. Users write messages (see the `logger`(1) man page) to the named pipe `/dev/log`.

9.5.2 Priority levels

The following eight priority levels (in order of highest to lowest priority) are defined for messages that the system log daemon handles:

Priority	Description
`emerg`	Panic condition, which is usually broadcast to all users
`alert`	Condition that you should correct immediately
`crit`	Critical condition
`err`	Errors
`warning`	Warning message
`notice`	Condition that is not an error condition, but possibly should be specially handled

`info`	Informational message
`debug`	Message useful only when debugging a program

9.5.3 `syslog` daemon startup

The `/etc/newsys` shell script starts `/etc/syslogd` and renames any existing log files. As released, the `/etc/newsys` shell script saves the 10 most recent copies of the log files and deletes the oldest. To preserve more or fewer log files, adjust this limit by editing the `/etc/newsys` shell script. Two shell functions, `quantity()` and `time_based()`, control the preservation of old log files, which are saved under the `/usr/adm/syslog/oldlogs` directory. A description of the `quantity()` and `time_based()` shell functions follows, followed by examples of their use and examples of the `/usr/adm/syslog` and the `/usr/adm/syslog/oldlogs` files.

Function	Description
`quantity()`	Preserves the specified quantity of the specified log files. `quantity()` is called with at least two arguments. The first is the number of copies to keep. The remaining arguments are the names of the files to be preserved. It will retain *x* copies of each file and delete the oldest.
`time_based()`	Preserves old log files on the basis of time, rather than system restarts. `time_based()` is passed at least two arguments. The first is the number of days to preserve files. The remaining arguments are the names of the actual files.

Note: If the base name of the log file consists of more than 6 characters, `time_based()` will not work. The pattern match in `find` will fail.

Examples:

```
#
#  Save 20 copies of daylog and debug
#
quantity 20 daylog debug
#
#  Save the last 30 days worth of kern and auth
#
time_based 30 kern auth
```

Examples of system log files follow:

```
# cd /usr/adm/syslog
# ls -lF
total 44

-rw-r--r--  1 root            0 Nov  9 11:03 auth
-rw-r--r--  1 root        15465 Nov  9 15:52 daylog
-rw-r--r--  1 root        15465 Nov  9 15:52 kern
drwxr-xr-x  2 root        11232 Nov  9 11:03 oldlogs/
```

```
# /usr/adm/syslog 5=> tail kern

Nov 9 15:42:39  unicos: NFS server sn218 not responding, giving up
Nov 9 15:42:39  unicos: NFS fsstat failed for server sn218: TIMED OUT
Nov 9 15:42:40  unicos: NFS server sn218 not responding, giving up
Nov 9 15:42:40  unicos: NFS getattr failed for server sn218: TIMED OUT
```

```
# /usr/adm/syslog 6=> cd oldlogs
# /usr/adm/syslog/oldlogs 7=> ls -CF

10-09.5.kern  10-17.6.kern    10-22.3.kern    10-29.1.kern
11-04.1.kern
10-10.0.auth  10-17.7.auth    10-23.0.auth    10-29.2.auth
11-04.2.auth
10-10.0.kern  10-17.7.kern    10-23.0.kern    10-29.2.kern
11-04.2.kern
10-11.0.auth  10-17.8.auth    10-23.1.auth    10-29.3.auth
11-04.3.auth
10-11.0.kern  10-17.8.kern    10-23.1.kern    10-29.3.kern
11-04.3.kern
10-11.1.auth  10-17.9.auth    10-23.2.auth    10-30.0.auth
11-05.0.auth...
10-16.0.auth   10-21.0.auth   10-27.0.auth
11-02.6.auth...
10-16.0.auth   10-21.0.auth   10-27.0.auth    11-02.6.auth    daylog.0
10-16.0.kern   10-21.0.kern   10-27.0.kern    11-02.6.kern    daylog.1
10-16.1.auth   10-21.1.auth   10-27.1.auth    11-03.0.auth    daylog.10
10-16.1.kern   10-21.1.kern   10-27.1.kern    11-03.0.kern    daylog.11
```

9.5.4 /usr/adm/sulog

The /usr/adm/sulog file contains a line of information for every attempted use of the /bin/su command since this version of the file was started. The line indicates whether the attempt was successful. You could monitor this log for attempted system breaching or other malicious use of a system. root should own this file, with no read or write permissions for others. The format of the log is as follows:

SU *MM/DD hh.mm flag tty olduser-newuser*

In the following sample /usr/adm/sulog file, the entry that contains a minus sign (line 6) indicates an unsuccessful attempt to use the /bin/su command:

```
# cat /usr/adm/sulog
SU 03/11 07:00 + console root-adm
SU 03/11 07:59 + ttyp000 guest-root
SU 03/11 08:13 + ttyp001 jones-root
SU 03/11 11:14 + ttyp002 jones-root
SU 03/11 11:33 + ttyp001 smith-root
SU 03/11 12:26 - ttyp001 smith-root
SU 03/11 12:26 + ttyp001 smith-root
```

9.5.5 /etc/dump.log

The /etc/dump.log file contains the time and reason for each system dump. The system supplies the time and the user supplies the reason. By default, the dump is located in /etc/dump.log and can be accessed using the normal file manipulations, such as tail, cat, and more. When the system is changing out of single-user mode, brc calls coredd to copy a dump file to a file system. To reconfigure the location of the dump, use the menu system. To change the location of this log file, use the cpdmp -l command.

> **Note:** This is a system dump log. It is **not** the log created by the dump utility (which is the /etc/dumpdates file).

An example of an /etc/dump.log follows:

```
# cat /etc/dump.log

cpdmp: 035120 blocks on dump device - waiting to be copied
01/26/94 07:27:09   coredd: Copying system dump into /core//04260727.
UNICOS dump copied to=/core//04260727/dump
   dump taken: 04/26/93 at 07:18:51
   reason: PANIC: master.s: EEX interrupt in UNICOS kernel
```

9.5.6 /usr/adm/nu.log

The new user log contains information about new user accounts on the system that are created by using `/etc/nu`. It includes entries about who created the account and the time it was added, information about the default environment settings, and the IDs. The `/etc/nu` command creates this file (for further information about `/etc/nu`, see Chapter 7, page 129).

The following types of user account transactions are recorded into `/usr/adm/nu.log`: `changed`, `added`, `deleted`, and `destroyed`.

An excerpt from a `nu.log` file follows:

```
Text goes  here
# cat /usr/adm/nu.log
jones:co:L B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
        changed to
jones:co:Lauren B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
jones:tp:type0[b]:3:tp:type0[i]:3:tp:type1[b]:3:tp:type1[i]:3
jones:tp:type2[b]:3:tp:type2[i]:3:tp:type3[b]:3:tp:type3[i]:3
        by jones on Mon Sept  13 11:51:00 1993
```

9.5.7 /usr/adm/sa/sa*DD*

The `sar` command uses the `/usr/adm/sa/da`*DD* data collection file to report system activity. The `/usr/lib/sa/sadc` and `/usr/lib/sa/sa1` commands write data to this file; they must be scheduled by `cron` to run at frequent intervals (such as every 15 minutes).

The `/usr/adm/sa/sa`*DD* file is too large and too varied to show a representative example. It is filled with multiple types of reports, each with many different output fields.

For more information about system activity reporting, see the `sar`(1) and `sar`(8) man pages and *UNICOS Resource Administration*, Cray Research publication SG–2302.

9.5.8 /usr/adm/sl/slogfile

The `/usr/adm/sl/slogfile` data file records UNICOS multilevel security (MLS) event information. The `reduce` command, executable only by the

security administrator, reads this data file. The `reduce` command extracts, formats, and outputs entries from UNICOS MLS event files. The MLS event logging daemon, `slogdemon`, collects security-relevant records from the operating system by reading the character special pseudo device `/dev/slog`. An excerpt of the output from the `reduce` command follows:

```
# /etc/reduce -s 04021300 -u jones -p

Apr  2 14:49:21 1993  Validation        o_lvl: 0  s_lvl: 0  jid:0  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]    ********
    Login uid: jones(8863)
  Login to [jones(8863),tng(146)] : Okay   via 128.162.121.20   on /dev/ttyp042
Apr  2 14:49:21 1993  Setuid Syscall    o_lvl: 0  s_lvl: 0  jid:1255  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]         ********
    Login uid: jones(8863)
   Setuid call from root (0) to jones (8863) was successful::
```

9.5.9 /usr/spool/msg/msglog.log

The `/usr/spool/msg/msglog.log` file contains messages and replies to and from the operator. Following is an excerpt from a `msglog.log` file:

```
# cat /usr/spool/msg/msglog.log

Sep 16 09:51:20  Message    1: WARNING THRESHOLD ON /nasc
Sep 16 09:58:59  Message    2: CRITICAL THRESHOLD ON /nasc::

Jun  9 23:34:02  Message daemon stopped
Jun 10 00:43:15  Message daemon started
Jun 10 04:07:49  Message    1: From ghe: How are you?
Jun 10 04:08:44  Reply    1: good::
Jun 18  12:41:52  Informative: ********** SYSTEM ACCOUNTING RESTARTED for 0618/*
Jun 18 12:48:21  Informative: *************** SYSTEM ACCOUNTING COMPLETE Thu J*:
```

9.5.10 /usr/lib/cron/cronlog

The `/usr/lib/cron/cronlog` file reports the status of all commands that `cron` executes, including `at`, `batch`, and `crontab`. When UNICOS is brought to multiuser mode, the old log file is copied to `/usr/lib/cron/OLDLOG`.

Various types of error messages may be present in the `cronlog` file, including messages about when `cron` was started and stages of job execution. The `cronlog` file has the following format:

`CMD:` *command_executed username process_id job_type start_time username process_id job_type end_time* `rc=`*error return code*

The *job_type* argument can have one of the following values:

`a` = `at`(1) job

`b` = Batch job

`c` = `cron`(8) job

An example of `/usr/lib/cron/cronlog` follows:

```
! *** cron started ***   pid = 3654 Thu Sep 16 17:47:44 1993
! new user (ce) with a crontab Thu Sep 16 17:47:45 1993
! new user (nfs) with a crontab Thu Sep 16 17:47:45 1993
! new user (root) with a crontab Thu Sep 16 17:47:46 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 3687 c Thu Sep 16 17:48:01 1993
<  root 3687 c Thu Sep 16 17:48:02 1993
>  CMD: /usr/lib/acct/ckpacct
>  root 4291 c Thu Sep 16 18:00:01 1993
>  CMD: /usr/lib/sa/sa1 600 1
>  root 4292 c Thu Sep 16 18:00:01 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 4293 c Thu Sep 16 18:00:01 1993
<  root 4293 c Thu Sep 16 18:00:02 1993
<  root 4292 c Thu Sep 16 18:00:02 1993
<  root 4291 c Thu Sep 16 18:00:04 1993
>  CMD: $HOME/scripts/runsequence cpuseq b
>  ce 4731 c Thu Sep 16 19:30:01 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 4732 c Thu Sep 16 19:30:01 1993
<  root 4732 c Thu Sep 16 19:30:01 1993
<  ce 4731 c Thu Sep 16 19:30:12 1993
```

9.5.11 /usr/tmp/nqs.log

The Network Queuing System (NQS) log, created by the NQS log daemon, contains NQS activity. Its default location is the ASCII file `/usr/spool/nqs/log` (to change the location of the log file, use the `qmgr`

`set log_file` command; to see where the current log file resides, use the `qmgr show parameters` command). Access to `/usr/spool/nqs` is restricted; however, if you have the correct permissions, you can access the NQS log file by using normal file manipulations, such as `tail`, `cat`, and `more`. If you experience problems with NQS, use a `tail -f` command on this file to observe what NQS is doing.

A sample `nqs.log` file follows:

```
# cat /usr/tmp/nqs.log

NQS(INFO): local mid = 130
I$nqs_boot(): TZ=CST6CDT
NQS(DEBUG): tra_read():0, pid 4033,  state=0, sequence#=0, tid=0
NQS(DEBUG): gen_shrpri_tree(): completed setudb.
NQS(INFO): gen_shrpri_tree(): Fair Share turned off, Share_wt sched factor set.
NQS(DEBUG): gen_shrpri_tree(): Sh_Decay_usage =  0.0000, Sh_Run_rate =  1.0000
NQS(DEBUG): gen_shrpri_tree(): Share_basis & SHAREBYACCT = 8
NQS(DEBUG): gen_shrpri_tree(): childcnt = 1, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 2, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 3, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 4, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 5, st[0].childsum = 0
NQS(INFO): nqs_ldconf(): i = 1NQS(INFO): nqs_ldconf(): Pipe queue gale; Dest count: 1
NQS(INFO): nqs_ldconf(): Creating new destination 0NQS(INFO): nqs_ldconf(): batch
NQS(INFO): nqs_upd.c(): Adding new destn batch to head of queue
NQS(INFO): upd_addquedes(): Updating queue gale destinations
NQS(INFO): upd_addquedes(): Destination 0;  832NQS(INFO): nqs_ldconf(): i = 1
NQS(INFO): upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO): upp_setlogfil(): Set/Reset command - $$/usr/spool/nqs/log
NQS(INFO): netdaemon(): Listening on TCP/IP port: nqs
NQS(INFO): nqs_rbuild(): Set flag for first time thru spawn.
NQS(INFO): nqs_boot(): BOOTDONE, Database present.
NQS(INFO): upp_setchkpntdir(): New directory = /usr/spool/nqs/private/root/chkt
NQS(INFO): upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO): upp_setlogfil(): Set/Reset command - #$/usr/spool/nqs/log
NQS(INFO): upp_setsnapfil(): New pathname = /home/swts/cjd/nqs_snapfile
```

9.5.12 `/usr/adm/errfile`

The error log is a binary file that contains error records from the operating system. `errpt` processes error reports from the data. The `/etc/errdemon`

command (see the `errdemon`(8) man page) reads `/dev/error` and places the error records from the operating system into either the specified file, or `errfile`, by default. The `/etc/rc` (see the `brc`(8) man page) script starts `/etc/errdemon`, and `/etc/mverr` starts a new `errfile`.

9.5.13 /usr/spool/dm/*

If UNICOS Data Migration Facility (DMF) software is configured on your system, the `/usr/spool/dm/dmdlog.`*YYMMDD* files record activities that pertain to data migration.

A sample `/dm/dmdlog.`*YYMMDD* file follows:

```
# cat /usr/spool/dm/dmdlog.930912

        dmdlog.930912

10:55:29 Data Migration daemon 35745 initializing, release level 6100
10:55:29 0 index entries in database
10:55:29 Command request pipe initialized, fd = 7
10:55:29 Kernel request pipe initialized, fd = 8
10:55:29 initmsp: msp fake, pid = 35751, wt_fd = 10, rd_fd = 11
10:55:29 machine id set to 2158163973
10:55:29 First available handle for assignment is 2158163973:1
10:55:30 0 incomplete MSP entries found
10:55:30 0 soft-deleted premigration files found
   .
   .
   .
10:56:35 Counts - permdel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - retrybu,     0,     0,     0,     0,  0,  0
10:56:35 Counts - krecall,    10,    20,    20,     0,  0,  1
10:56:35 Counts - kremove,    28,    28,    28,     0,  0,  1
10:56:35 Counts - kcancel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - invalid,     0,     0,     0,     0,  0,  0
10:56:35 Counts -  pclear,     0,     0,     0,     0,  0,  0
10:56:35 Current mem  = 94437
10:56:35 Stopping daemon processing
10:56:35 Data migration daemon stopped, exit=0
```

Note: The following log files also exist for each file system under data migration:

- `dmloght` (generated by the `dmhit` command)
- `dmlogct` (generated by the `dmmctl` command)
- `dmlogsm` (generated by the `dmfree` command)

9.6 Cleaning up system logs

Some log files are recycled during each reboot, some logs accumulate content slowly and must be cleaned up only occasionally, and some log files accumulate content quickly and should be monitored and cleaned up frequently. This subsection describes each group of log files.

9.6.1 Log files recycled during each reboot

The following log files are recycled during each reboot; therefore, you do not have to monitor them for space consumption. If any of the log files must be saved, however, you should copy them to a location of your choice before shutting down the system. If you forget their location, most of the log files are linked to `/usr/spool/ccflogs` .

Log files that recycle are as follows:

- `/etc/rc.log` (log file from `init 2` function)
- `/usr/adm/sulog` (including all `su` records)
- `/usr/spool/msg/msglog.log` (messages and replies from and to an operator)
- `/usr/lib/cron/log` (all `cron` entries since reboot)
- `/usr/tmp/nqs.log` (all NQS entries)

9.6.2 Small accumulative log files

The following log files accumulate content slowly, but you should clean them up occasionally so that they do not consume space needlessly:

- `/etc/boot.log` (records boot dates and times for a system)
- `etc/dump.log` (records crash dump dates and dump file locations)

- `usr/adm/nu.log` (records all `/etc/nu` output)

9.6.3 Large accumulative log files

The system activity report (`sar`) data and report log files accumulate content quickly; therefore, you should monitor these and clean them up frequently. If not managed promptly, these log files could potentially saturate the `/usr` file system. All `sar` data is saved up to 31 days in `/usr/adm/sa/sa`*DD*. At the end of each month, you should dump them to a file server or to tape; otherwise, newer collected data will overwrite them. The `sar` reports (stored in `/usr/adm/sa/sar`*DD*) are kept only up to 7 days, because the reports usually are not backed up. To change the number of days you want to keep `sar` data or `sar` reports, modify `/usr/lib/sa/sa2`.

You also should monitor the following log files:

- Email log file
- User mail files if not read and cleared
- NQS log files (these can grow quickly)
- `errpt` files when active disk errors or tape activity exists
- MLS log files, which are located in `/usr/adm/sl`

Accounting [10]

UNICOS provides two types of system accounting, standard UNIX System V accounting or Cray system accounting (CSA). You may use one or the other of these accounting packages at your site. To help you decide which accounting package to use, see Section 10.1, page 195, which describes the unique features of CSA.

For information on using standard UNIX System V accounting, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

This section describes CSA, which is the more complete and frequently used of the two accounting types. It includes the following:

- An overview of CSA, including unique CSA features, descriptions of directories and files, and the `/usr/lib/acct/csarun` primary daily accounting shell script.
- Procedures to follow so that you can set up CSA and execute daily accounting procedures that result in the generation of a variety of reports.

Note: The UNICOS Station Call Processor (USCP) does not apply to the CRAY J90se system.

Your accounting configuration file is located in `/etc/config/acct_config`. A sample file is provided at the end of this section; the sample file may differ slightly from the one included with your system.

10.1 Cray Research system accounting (CSA)

Cray Research system accounting (CSA) is designed to meet the unique accounting requirements of Cray Research sites. Like the standard UNIX accounting package, CSA provides methods to collect per-process resource utilization data, record connect sessions, monitor disk usage, and charge fees to specific logins. CSA also provides other facilities that are not available from the standard accounting package. These include the following:

- Per-job accounting
- Accounting for socket usage
- Device accounting

- Daemon accounting (for the Network Queuing System (NQS) and the UNICOS tape subsystem)
- Disk accounting by account ID
- Arbitrary accounting periods
- Flexible system billing unit (SBU) system
- One file containing all data for an accounting period
- Off-line archiving of accounting data

Sites may run either the standard UNICOS accounting programs or the CSA package by invoking the appropriate shell scripts and programs. Both packages are installed with the UNICOS 10.0 release.

UNICOS system features in the CSA package include configurable parameters located in a single file, `/etc/config/acct_config`, and a set of user-defined exits that allows sites to tailor the daily run of accounting to their specific needs.

10.1.1 Concepts and terminology

The following concepts and terms are important in CSA:

Term	Description
Daily accounting	Unlike the standard daily accounting, CSA's accounting can be run as many times as necessary during a day. However, this feature is still referred to as *daily accounting*.
Periodic accounting	Accounting similar to the standard UNICOS monthly accounting. CSA, however, lets system administrators specify the time periods for which "monthly" or cumulative accounting is to be run. Thus, periodic accounting can be run more than once a month.
Recycled data	By default, accounting data for active sessions is recycled until the session terminates. CSA reports only data for terminated sessions unless `csarun`(8) is invoked with the `-A` option. `csarun` places recycled data into data files in the `/usr/adm/acct/day` directory. These data files are suffixed with `0`; for example, per-process accounting data for active sessions from previous accounting periods is in the `/usr/adm/acct/day/pacct0` file.
Session	CSA organizes accounting data by sessions and boot times and then places the data into a session record file. For non-NQS jobs, a *session* consists of all accounting data for a given job ID during a single boot period. A *session* for an NQS job consists of the accounting data for all job IDs associated with the job's NQS sequence number/machine name identifier. NQS jobs may span multiple boot periods. If a job is restarted, it has the same job ID associated with it during all boot periods in which it runs. Rerun NQS jobs have multiple job IDs. CSA treats all phases of an NQS job as being in the same session.

Uptime period or boot period	A period delineated by the system boot times found in `/etc/csainfo`. The `csaboots`(8) command writes to this file during system boot.

10.1.2 Files and directories overview

This section provides a brief overview of the CSA file and directory structure. A more complete description of the files and directories can be found in Section 10.1.7, page 214.

10.1.2.1 Structures of the `acct` and `tmp` directories

The directory structure of `/usr/adm/acct` is set up so that it is easy to find CSA data files and reports. The `/tmp` structure is also used while `csarun`(8) is running. Figure 3 illustrates the directory structure for both directories.

Figure 3. `/usr/adm/acct` and `tmp` directory structures

Note: As distributed, only the directory `/usr/adm/acct/day` is readable by all users. Within the `day` directory, only the `pacct*` files are readable by all users. This allows any user to examine the `pacct*` files by using the `acctcom`(1) command. All other directories and files within `/usr/adm/acct` are accessible only by `root` and users in the group `adm`.

Warning: `acctcom`(1) on a Cray ML-Safe configuration of the UNICOS system is considered to be a covert channel. You may want to consider restricting access to this command to the `adm` group.

The following abbreviations have these meanings:

Abbreviation	Definition
MMDD	Month, day
hhmm	Hour, minute

10.1.2.2 Shell scripts and C binaries

The `/usr/lib/acct` directory contains virtually all of the programs and scripts used by both the standard accounting and CSA packages. The only CSA program not located here is `/etc/csaboots` (see `csaboots`(8)), which records

boot times at system startup. Programs used only by CSA begin with the characters `csa`.

10.1.2.3 Unprocessed data files

Both CSA and the standard accounting package expect most unprocessed accounting files to be located in the `/usr/adm/acct/day` directory. The use of this directory simplifies tracking of the current accounting files. The following table shows the location of the raw data files.

Accounting file	Description
`/usr/adm/acct/day/dtmp`	Disk accounting data
`/usr/adm/acct/day/nqacct*`	NQS daemon accounting data
`/usr/adm/acct/day/pacct*`	Per-process accounting data
`/usr/adm/acct/day/tpacct*`	Tape daemon accounting data
`/usr/adm/acct/day/soacct*`	Socket accounting data
`/etc/csainfo`	Boot times
`/etc/wtmp`	Connect time accounting data

Warning: On a Cray ML-Safe configuration of the UNICOS system, `/etc/wtmp` is considered a covert channel. You may want to consider restricting access to this file to the `adm` group.

Accounting files in `/usr/adm/acct/day` whose names include the suffix `0` contain data from sessions that did not complete during the previous accounting periods.

During CSA data processing, sites may select to archive the raw and/or processed data off-line. Section 10.1.5, page 207, describes how to do this. By default, all raw data files are deleted after use and are not archived.

10.1.2.4 Data files being processed

At the start of a daily accounting run, CSA moves the raw data files from `/usr/adm/acct/day` to the appropriate `/usr/adm/acct/work/`*MMDD/hhmm* directory. The files in the work directory are as follows:

File	Description
`Ever.tmp`	Data verification work file
`Pctime*`	Preprocessed connect time data
`Pnqacct*`	Preprocessed NQS data
`Puptime*`	Uptimes
`Rctime0`	Connect data to be recycled in the next accounting run
`Rnqacct0`	NQS data to be recycled in the next accounting run
`Rpacct0`	Per-process accounting data to be recycled in the next accounting run
`Rtpacct0`	Tape data to be recycled in the next accounting run
`Ruptime0`	Uptimes to be recycled in the next accounting run
`Wctime*`	Verified raw connect time data
`Wdisktacct`	Disk accounting data (`cacct.h` format)
`Wdtmp`	Disk accounting data from `diskusg`(8) or `acctdusg`(8)
`Wnqacct*`	Raw NQS accounting data
`Wpacct*`	Raw per-process accounting data
`Wsoacct*`	Raw socket accounting data
`Wtpacct*`	Raw tape accounting data
`Wwtmp`	Raw connect time data

10.1.2.5 Processed data files

CSA outputs the following data files:

File	Description
`/tmp/AC.`*MMDD*`/`*hhmm*`/Super-record`	Session record file; this file is usually deleted after it has been used by CSA.

`/usr/adm/acct/fiscal/data/`*MMDD*`/`*hhmm*`/pdacct`

Consolidated periodic data.

`/usr/adm/acct/fiscal/data/`*MMDD*`/`*hhmm*`/cms`

Periodic command usage data.

`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cacct`

Consolidated daily data; this file is deleted by `csaperiod`(8) if the `-r` option is specified.

`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cms`

Daily command usage data; this file is deleted by `csaperiod`(8) if the `-r` option is specified.

`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/dacct`

Daily disk usage data; this file is deleted by `csaperiod`(8) if the `-r` option is specified.

10.1.2.6 Reports

CSA generates daily and periodic reports. The locations of these reports are as follows:

File	Description
`/usr/adm/acct/fiscal/rpt/`*MMDD*`/`*hhmm*`/rprt`	Periodic accounting report
`/usr/adm/acct/sum/rpt/`*MMDD*`/`*hhmm*`/rprt`	Daily accounting report

10.1.3 Daily operation overview

When the UNICOS operating system is run in multiuser mode, accounting behaves in a manner similar to the following process. However, because sites may customize CSA, the following may not reflect the actual process at a particular site:

1. System boot time is written to `/etc/csainfo`. Each time the system is booted, the boot time is written to `/etc/csainfo` by the `/etc/csaboots` command, which is invoked by `rc` (see `brc`(8)) during system startup.

2. Process accounting is enabled. When the system is switched to multiuser mode, the `/usr/lib/acct/startup` (see `acctsh`(8)) script is called by `/etc/rc` and performs the following functions:

   a. Writes an `acctg on` record to `/etc/wtmp`; the `acctwtmp` program is used to write this record.

   b. Enables process accounting with the command line `/usr/lib/acct/turnacct on`; `turnacct`(8) calls the `accton` program with the argument `/usr/adm/acct/day/pacct`.

   c. Removes lock files and saved `pacct` and `wtmp` files. `/usr/lib/acct/remove` is invoked to clean up saved `pacct` and `wtmp` files in `/usr/adm/acct/sum`. Unlike the standard accounting package, CSA does not leave files in this directory. In addition, the lock files are removed from `/usr/adm/acct/nite`.

3. By default, daemon accounting for NQS, tape, and sockets is handled by the `/usr/lib/acct/startup` script. However, in order to run NQS and tape daemon accounting, you must modify the appropriate subsystem. Section 10.1.4, page 203, describes this process in detail.

4. The amount of disk space used by each user is determined periodically. `/usr/lib/acct/dodisk` (see `dodisk`(8)) is run periodically by `cron` to generate a snapshot of the amount of disk space being used by each user. `dodisk` should be run at most once for each time `/usr/lib/acct/csarun` (see `csarun`(8)) is run. Multiple invocations of `dodisk` during the same accounting period write over previous `dodisk` output.

5. A fee file is created. Sites desiring to charge fees to certain users can do so by invoking `/usr/lib/acct/chargefee` (see `chargefee`(8)). Each accounting period's fee file (`/usr/adm/acct/day/fee`) is merged into the consolidated accounting records by `/usr/lib/acct/csaperiod` (see `csaperiod`(8)).

6. Daily accounting is run. At specified times during the day, `csarun` is executed by `cron` to process the current accounting data. The output from `csarun` is a consolidated daily accounting file and an ASCII report.

7. Periodic accounting is run. At a specific time during the day, or on certain days of the month, `/usr/lib/acct/csaperiod` (see `csaperiod`(8)) is

executed by `cron` to process consolidated accounting data from previous accounting periods. The output from `csaperiod` is a consolidated periodic accounting file and an ASCII report.

8. Accounting is disabled. When the system is shut down gracefully, the script `/usr/lib/acct/shutacct` (see `shutacct`(8)) is executed by `/etc/shutdown` (see `shutdown`(8)). `shutacct` writes an "acctg off" record to `/etc/wtmp`. It then calls `/usr/lib/acct/turnacct` and `/usr/lib/acct/turndacct` to disable per-process and daemon accounting (see `turnacct`(8) and `turndacct`(8)).

10.1.4 Setting up CSA

The following is a brief description of setting up CSA. Site-specific modifications are discussed in detail in Section 10.1.10, page 229. As described in this section, CSA is run by a person with super-user permissions. CSA also can be run by users who have `acct` permissions and are in the `adm` group. See Section 10.1.10.7, page 244, for the necessary modifications.

1. Change the default system billing unit (SBU) weighting factors, if necessary. By default, no SBUs are calculated. If your site wants to report SBUs, you must modify the configuration file `/etc/config/acct_config`.

2. Modify any necessary parameters in the `/etc/config/acct_config` file, which contains configurable parameters for the accounting system. Ensure that parameters, such as `MEMINT`, reflect the needs of your site.

3. If you want daemon accounting, you must enable daemon accounting at system startup time by performing the following steps:

   a. Ensure that the variables in `/etc/config/acct_config` for the subsystems for which you want to enable daemon accounting are set to `on`. Set the `NQS_START`, `TAPE_START`, and `SOCKET_START` parameters to `on` to enable NQS, online tapes, and socket accounting, respectively.

   b. If necessary, enable accounting from the daemon's side. Specifically, NQS and tape accounting must also be enabled by the associated daemon. Use the `qmgr`(8) `set accounting on` command to turn on NQS accounting. To enable tape daemon accounting, execute `tpdaemon`(8) with the `-c` option. Socket accounting does not require any additional processing.

4. Prior to setting up the following `cron` jobs, ensure that the `/etc/checklist` file exists. By default, `dodisk`(8) performs disk accounting on the special files listed in `checklist`. For most installations,

entries similar to the following should be made in `/usr/spool/cron/crontabs/root` so that `cron`(8) automatically runs daily accounting:

```
0 4 * * 1-6 /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
0 3 * * 1-6 /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dk2log
```

`csarun`(8) should be executed at such a time that `dodisk` has sufficient time to complete. If `dodisk` does not complete before `csarun` executes, disk accounting information may be missing or incomplete.

`dodisk` must be invoked with either the `-a` or the `-A` option. If it is not, `csaperiod`(8) aborts when it attempts to merge the disk usage information with other accounting data.

5. Periodically check the size of the `acct` files. Entries similar to the following should be made in `/usr/spool/cron/crontabs/root`:

   ```
   0 * * * * /usr/lib/acct/ckdacct nqs tape socket
   0 * * * * /usr/lib/acct/ckpacct
   ```

   `cron`(8) should periodically execute the `ckpacct`(8) and `ckdacct`(8) shell scripts. If the `pacct` file grows larger than 500 blocks (default), `ckpacct` calls the command `/usr/lib/acct/turnacct switch` to start a new `pacct` file. `ckpacct` also makes sure that there are at least 500 free blocks on the file system containing `/usr/adm/acct` (`/usr` by default). If there are not enough blocks, per-process accounting is turned off. The next time `ckpacct` is executed, it turns per-process accounting back on if there are enough free blocks.

   `ckdacct` performs an analogous function for daemon accounting. If a daemon's accounting file is larger than 500 blocks (default), the command `/usr/lib/acct/turndacct switch` is executed in order to start a new accounting file. In addition, `ckdacct` also checks the amount of free blocks on the `ACCT_FS` file system (`/usr` by default).

   Ensure that the `ACCT_FS` and `MIN_BLKS` variables have been set correctly in the `/etc/config/acct_config` configuration file. `ACCT_FS` is the file system containing `/usr/adm/acct`; the default is `/usr`. `MIN_BLKS` is the minimum number of free blocks needed in the `ACCT_FS` file system. The default is 500.

   It is very important that `ckpacct` and `ckdacct` be run periodically so that an administrator is notified when the accounting file system (`/usr` by default) runs out of disk space. After the file system is cleaned up, the next

invocation of `ckpacct` and `ckdacct` enables per-process and daemon accounting. You can manually reenable accounting by invoking `turnacct`(8) and `turndacct`(8) with the `on` operand.

If `ckpacct` and `ckdacct` are not run periodically, and the accounting file system runs out of space, an error message is written to the console stating that a write error occurred and that accounting is disabled. If you do not free disk space as soon as possible, a vast amount of accounting data can be lost unnecessarily. Additionally, lost accounting data can cause `csarun`(8) to abort or report erroneous information.

6. To run periodic accounting, an entry similar to the following should be made in `/usr/spool/cron/crontabs/root`. This command generates a periodic report on all consolidated data files found in `/usr/adm/acct/sum/data/*` and then deletes those data files:

```
15 5 1 * * /usr/lib/acct/csaperiod -r 2> /usr/adm/acct/nite/pd2log
```

   This entry is executed at such a time that `csarun`(8) has sufficient time to complete. This example results in the creation of a monthly accounting file and report on the first day of each month. These files contain information about the previous month's accounting.

7. Update the `holidays` file. The `/usr/lib/acct/holidays` file contains the prime/nonprime time table for the accounting system, which should be edited to reflect your site's holiday schedule for the year.

   By default, the `holidays` file is located in the `/usr/lib/acct` directory. You can change this location by modifying the *HOLIDAY_FILE* variable in `/etc/config/acct_config`. If necessary, modify the *NUM_HOLIDAYS* variable (also located in `/etc/config/acct_config`), which sets the upper limit on the number of holidays that can be defined in *HOLIDAY_FILE*. The format of this file is composed of the following types of entries:

   - Comment lines: These lines may appear anywhere in the file as long as the first character in the line is an asterisk (*).
   - Version line: This line must be the first uncommented line in the file and must only appear once. It denotes that the new holidays file format is being used. This line should not be changed by the site.
   - Year designation line: This line must be the second uncommented line in the file and must only appear once. The line consists of two fields. The first field is the keyword YEAR. The second field must be either the current year or the wildcard character, asterisk (*). If the year is

wildcarded, the current year is automatically substituted for the year. The following are examples of two valid entries:

```
YEAR        1997
YEAR        *
```

- Prime/nonprime time designation lines: These must be uncommented lines 3, 4, and 5 in the file. The format of these lines is as follows:

  period prime_time_start nonprime_time_start

  The variable *period* is one of the following: WEEKDAY, SATURDAY, or SUNDAY. The *period* can be in either upper or lowercase.

  The prime and nonprime start time can be one of two formats:

  – Both start times are 4-digit numeric values between 0000 and 2359. The *nonprime_time_start* value must be greater than the *prime_time_start* value. For example, it is incorrect to have prime time start at 07:30 A.M. and nonprime time start at 1 minute after midnight. Therefore, the following entry is wrong and can cause incorrect accounting values to be reported.

    ```
    WEEKDAY  0730  0001
    ```

    It is correct to specify prime time to start at 07:30 A.M. and nonprime time to start at 5:30 P.M. on weekdays. You would enter the following in the holiday file:

    ```
    WEEKDAY  0730  1730
    ```

  – Start times specify that the entire period is to be either all prime time or all nonprime time. To specify that the entire period is to be considered prime time, set *prime_time_start* to ALL and *nonprime_time_start* to NONE. If the period is to be considered all nonprime time, set *prime_time_start* to NONE and *nonprime_time_start* to ALL. For example, to specify Monday through Friday as all prime time, you would enter the following:

    ```
    WEEKDAY  ALL  NONE
    ```

    To specify all of Sunday to be nonprime time, you would enter the following:

    ```
    SUNDAY  NONE  ALL
    ```

- Company holidays lines: These entries follow the year designation line and have the following general format:

  day-of-year Month Day Description of Holiday

  The *day-of-year* field is a number in the range 1 through 366, indicating the day for a given holiday (leading white space is ignored). The other three fields are commentary and are not currently used by other programs. Each holiday is considered all nonprime time.

  If the `holidays` file does not exist or there is an error in the year designation line, the default values for all lines are used.

  If there is an error in a prime/nonprime time designation line, the entry for the erroneous line is set to a default value. All other lines in the `holidays` file are ignored and default values are used.

  If there is an error in a company holidays line, all holidays are ignored.

  The default values are as follows:

`YEAR`	The current year.
`WEEKDAY`	Monday through Friday is all prime time.
`SATURDAY`	Saturday is all nonprime time.
`SUNDAY`	Sunday is all nonprime time.

  No holidays are specified

10.1.5 The `csarun` command

The `/usr/lib/acct/csarun` command is the primary daily accounting shell script. It processes connect, disk, per-process, and daemon accounting files and is normally initiated by `cron`(8) during nonprime hours.

`csarun`(8) also contains four user-exit points allowing sites to tailor the daily run of accounting to their specific needs.

The `csarun` command does not damage files in the event of errors. It contains a series of protection mechanisms that attempt to recognize an error, provide intelligent diagnostics, and terminate processing in such a way that `csarun` can be restarted with minimal intervention.

10.1.5.1 Daily invocation

The `csarun` command is invoked periodically by `cron`(8). It is very important that you ensure that the previous invocation of `csarun` completed successfully before invoking `csarun` for a new accounting period. If this is not done, information about unfinished sessions will be inaccurate.

Data for a new accounting period can also be interactively processed by executing the following:

```
nohup csarun 2> /usr/adm/acct/nite/fd2log &
```

Before executing `csarun` in this manner, ensure that the previous invocation completed successfully. To do this, look at the files `active` and `statefile` in `/usr/adm/acct/nite`. Both files should specify that the last invocation completed successfully.

10.1.5.2 Error and status messages

The `csarun` error and status messages are placed in the `/usr/adm/acct/nite` directory. The progress of a run is tracked by writing descriptive messages to the file `active`. Diagnostic output during the execution of `csarun` is written to `fd2log`. The `lock` and `lock1` files prevent concurrent invocations of `csarun`; `csarun` will abort if these two files exist when it is invoked. The `clastdate` file contains the month, day, and time of the last two executions of `csarun`.

Errors and warning messages from programs called by `csarun` are written to files that have names beginning with `E` and ending with the current date and time. For example, `Ebld.11121400` is an error file from `csabuild`(8) for a `csarun` invocation on November 12, at 14:00.

If `csarun` detects an error, it sends an informational message to the operator with `msgi`(1), sends mail to `root` and `adm`, removes the locks, saves the diagnostic files, and terminates execution. When `csarun` detects an error, it will send mail either to `MAIL_LIST` if it is a fatal error, or to `WMAIL_LIST` if it is a warning message, as defined in the configuration file `/etc/config/acct_config`.

10.1.5.3 States

Processing is broken down into separate reentrant states so that `csarun` can be restarted. As each state completes, `/usr/adm/acct/nite/statefile` is updated to reflect the next state. When `csarun` reaches the `CLEANUP` state, it removes various data files and the locks, and then terminates.

The following describes the events that occur in each state. *MMDD* refers to the month and day `csarun` was invoked. *hhmm* refers to the hour and minute of invocation.

State	Description
`SETUP`	The current accounting files are switched via `turnacct`(8) and `turndacct`(8). These files are then moved to the `/usr/adm/acct/work/`*MMDD/hhmm* directory. File names are prefaced with `W`. `/etc/wtmp` and `/etc/csainfo` are also moved to this directory.
`WTMPFIX`	The `wtmp` file in the `work` directory is checked for accuracy by `wtmpfix` (see `fwtmp`(8)). Some date changes cause `csaline`(8) to fail, so `wtmpfix` attempts to adjust the time stamps in the `wtmp` file if a date change record appears. If `wtmpfix` is unable to fix the `wtmp` file, the `wtmp` file must be manually repaired. This is described in Section 10.1.6.1, page 212.
`VERIFY`	By default, per-process and NQS accounting files are checked for valid data. In addition, tape and socket accounting files are verified. Records with invalid data are removed. Names of bad data files are prefixed with `BAD.` in the `/usr/adm/acct/work/*` directory. The corrected files do not have this prefix.
`PREPROC`	The NQS and connect time (`wtmp`) accounting files are run through preprocessors. File names of preprocessed files are prefixed with a `P` in the `/usr/adm/acct/work/`*MMDD/hhmm* directory.
`ARCHIVE1`	First user exit of the `csarun` script. If a script named `/usr/lib/acct/csa.archive1` exists, it will be executed through the shell . (dot) command. The . (dot) command will not execute a compiled program, but the user exit script can. You might use this user exit to archive the accounting files in `${WORK}`.
`BUILD`	The per-process, NQS, tape, socket, and connect accounting data is organized into a session record file.
`ARCHIVE2`	Second user exit of the `csarun` script. If a script named `/usr/lib/acct/csa.archive2` exists, it will be executed through the shell . (dot) command. The . (dot) command will not execute a compiled program, but the user exit script can. You might use this exit to archive the session record file.

`CMS`	Produces a command summary file in `cacct.h` format. The `cacct` file is put into the `/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm* directory for use by `csaperiod`(8).
`REPORT`	Generates the daily accounting report and puts it into `/usr/adm/acct/sum/rpt/`*MMDD*`/`*hhmm*`/rprt`. A consolidated data file, `/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cacct`, is also produced from the session record file. In addition, accounting data for unfinished sessions is recycled.
`DREP`	Generates a daemon usage report based on the session file. This report is appended to the daily accounting report, `/usr/adm/acct/sum/rpt/`*MMDD*`/`*hhmm*`/rprt`.
`FEF`	Third user exit of the `csarun` script. If a script named `/usr/lib/acct/csa.fef` exists, it will be executed through the shell . (dot) command. The . (dot) command will not execute a compiled program, but the user exit script can. `csarun` variables are available, without being exported, to the user exit script. You might use this exit to convert the session record file to a format suitable for a front-end system.
`USEREXIT`	Fourth user exit of the `csarun` script. If a script named `/usr/lib/acct/csa.user` exists, it will be executed through the shell . (dot) command. The . (dot) command will not execute a compiled program, but the user exit script can. `csarun` variables are available, without being exported, to the user exit script. You might use this exit to run local accounting programs.
`CLEANUP`	Cleans up temporary files, removes the locks, and then exits.

10.1.5.4 Restarting `csarun`

If `csarun`(8) is executed without arguments, the previous invocation is assumed to have completed successfully.

The following operands are required with `csarun` if it is being restarted:

`csarun [`*MMDD* `[`*hhmm* `[`*state*`]]]`

MMDD is month and day, *hhmm* is hour and minute, and *state* is the `csarun` entry state.

To restart `csarun`, follow these steps:

1. Remove all lock files by using the following command line:

   ```
   rm -f /usr/adm/acct/nite/lock*
   ```

2. Execute the appropriate `csarun` restart command, using the following examples as guides:

   a. To restart `csarun` using the time and state specified in `clastdate` and `statefile`, execute the following command:

      ```
      nohup csarun 0601 2> /usr/adm/acct/nite/fd2log &
      ```

      In this example, `csarun` will be rerun for June 1, using the time and state specified in `clastdate` and `statefile`.

   b. To restart `csarun` using the state specified in `statefile`, execute the following command:

      ```
      nohup csarun 0601 0400 2> /usr/adm/acct/nite/fd2log &
      ```

      In this example, `csarun` will be rerun for the June 1 invocation that started at 4:00 A.M., using the state found in `statefile`.

   c. To restart `csarun` using the specified date, time, and state, execute the following command:

      ```
      nohup csarun 0601 0400 BUILD 2> /usr/adm/acct/nite/fd2log &
      ```

      In this example, `csarun` will be restarted for the June 1 invocation that started at 4:00 A.M., beginning with state `BUILD`.

Before `csarun` is restarted, the appropriate directories must be restored. If the directories are not restored, further processing is impossible. These directories are as follows:

`/usr/adm/acct/work/`*MMDD/hhmm*
`/usr/adm/acct/sum/data/`*MMDD/hhmm*
`/usr/adm/acct/sum/rpt/`*MMDD/hhmm*
`/tmp/AC.`*MMDD/hhmm*

If you are restarting at state `ARCHIVE2`, `CMS`, `REPORT`, `DREP`, or `FEF`, the session file must already exist in `/tmp/AC.`*MMDD/hhmm*. If the file does not exist, `csarun` will automatically restart at the `BUILD` state. Depending on the tasks performed during the site-specific `USEREXIT` state, the session file may or may not need to exist.

10.1.6 Verifying and correcting data files

This section describes how to remove bad data from various accounting files.

10.1.6.1 Fixing wtmp errors

The wtmp files generally cause the highest number of errors in the day-to-day operation of the accounting subsystem. When the date is changed, and the UNICOS system is in multiuser mode, a set of date change records is written into the /etc/wtmp file. The wtmpfix (see fwtmp(8)) program is designed to adjust the time stamps in the wtmp records when a date change is encountered.

Some combinations of date changes and reboots, however, slip by wtmpfix and cause csaline(8) to fail. The following example shows how to repair a wtmp file:

```
$ cd /usr/adm/acct/work/MMDD/hhmm
$ /usr/lib/acct/fwtmp < Wwtmp > xwtmp
$ ed xwtmp
   (delete corrupted records)
$ /usr/lib/acct/fwtmp -ic < xwtmp > Wwtmp
   (restart csarun at the WTMPFIX state)
```

If the wtmp file is beyond repair, create a null Wwtmp file. This prevents any charging of connect time.

10.1.6.2 Verifying data files

You can verify data files with the csaedit(8), csapacct(8), and csaverify(8) commands. csaedit and csapacct verify and delete bad data records, while csaverify only flags bad records. By default, csaedit and csaverify are invoked in csarun to verify the data files.

Note that these commands may allow files that contain bad data, such as very large values, to be successfully verified.

10.1.6.3 Editing data files

You can use the csaedit(8) and csapacct(8) commands to verify and remove records from various accounting files. The following example shows how you can use csapacct to verify and remove bad records from a per-process (pacct) accounting file.

In this example, `csapacct` is invoked with verbose mode enabled (valid data records are written to the file `pacct.NEW`):

```
/usr/lib/acct/csapacct -v pacct pacct.NEW
```

The output produced by this command line is as follows:

```
Bad record - starting byte offset is 077740 (32736)
   invalid pacct record - bad base parent process id 97867
Found the next magic word at byte offset 0100130, ignored 120 bytes


Found 394 BASE records
Found 4 EOJ records
Found 1 MTASK (multitasking) records
Found 0 ERROR records
Found 0 IO records
Found 0 SDS records                    # not on CRAY EL systems
Found 0 MPP records                    # not on CRAY EL systems
Found 0 PERFORMANCE records
Outputted records for 398 processes
Ignored 120 bytes from the input file
```

You can use `csaedit` and `csapacct` in conjunction with `csaverify`, by first running `csaverify` and noting the byte offsets of the first bad record. Next, execute `csaedit` or `csapacct` and remove the record at the specified offset. The following example shows how you can verify and then edit a bad `pacct` accounting file:

1. The `pacct` file is verified with the following command line, and the following output is received:

```
$  /usr/lib/acct/csaverify -P pacct

/usr/lib/acct/csaverify: pacct: invalid pacct record - bad base parent process id 97867
  byte offset: start = 077740 (32736)  word offset: start = 07774 (4092)
/usr/lib/acct/csaverify: pacct: invalid pacct record - bad magic word 03514000
  byte offset: start = 0100070 (32824)  word offset: start = 010007 (4103)
```

2. The record found at byte offset 32736 is deleted as follows (valid records are written to `pacct.NEW`):

   ```
   /usr/lib/acct/csapacct -o 32736 pacct pacct.NEW
   ```

3. The new `pacct` file is reverified as follows to ensure that all the bad records have been deleted:

   ```
   /usr/lib/acct/csaverify -P pacct.NEW
   ```

You can use `csaedit` to produce an abbreviated ASCII version of some of the daemon accounting files and `acctcom`(1) to generate a similar ASCII version of `pacct` files.

10.1.7 Files and directories

This section describes the files and directories used by CSA.

10.1.7.1 `/usr/adm/acct` directory

The `/usr/adm/acct` directory contains the following directories:

Directory	Description
`day`	Current accounting files
`fiscal`	Periodic accounting data and reports
`nite`	Processing messages and errors
`sum`	Daily accounting data and reports
`work`	Temporary work area

The `/usr/adm/acct/day` directory contains the current accounting files, as shown in the following list. Files with names ending with `0` contain data for uncompleted sessions from previous days.

File	Description
`dtmp`	Disk accounting data (ASCII) created by `dodisk`(8)
`nqacct*`	NQS daemon accounting data
`pacct*`	Per-process accounting data
`soacct*`	Socket accounting data

`tpacct*`	Tape daemon accounting data

The `/usr/adm/acct/fiscal/data/`*MMDD*`/`*hhmm* directory contains processed, periodic, binary accounting data in the form of the following files:

File	Description
`cms`	Periodic command usage data in command summary (cms) record format
`pdacct`	Consolidated periodic data generated on *MMDD* at *hhmm*

The `/usr/adm/acct/fiscal/rpt/`*MMDD*`/`*hhmm* directory contains the periodic accounting report, `rprt`, that was generated on *MMDD* at *hhmm*.

The `/usr/adm/acct/nite` directory contains error messages and status information about the accounting runs in the following files:

File	Description
`active`	Progress and status of `csarun`
`active`*MMDDhhmm*	Progress and status of `csarun` after an error has been detected
`clastdate`	Last two times `csarun` was executed; in *MMDD hhmm* format
`disktacct`	Disk accounting records in `cacct.h` format; created by `dodisk`(8)
`dk2log`	Diagnostic output created during execution of `dodisk`
`E*`*MMDDhhmm*	Error/warning messages for an accounting run done on *MMDD* at *hhmm*
`fd2log`	Diagnostic output created during execution of `csarun`
`lineuse`	tty line usage report
`lock, lock1`	Controls simultaneous invocations of `csarun`
`pd2log`	Diagnostic output created during execution of `csaperiod`
`pdact`	Progress and status of `csaperiod`
`pdact`*MMDDhhmm*	Progress and status of `csaperiod` after an error has been detected

`reboots`	The start and ending dates from `wtmp` and a listing of reboots
`statefile`	Current state during `csarun` execution
`tmpwtmp`	The `wtmp` file corrected by `wtmpfix` (see `fwtmp`(8))
`wtmperror`	`wtmpfix` error messages

The `/usr/adm/acct/sum/data/`*MMDD/hhmm* directory contains daily, binary accounting data in the following files:

File	Description
`cacct`	Consolidated daily data generated on *MMDD* at *hhmm* in `cacct.h` format
`cms`	Command usage data in command summary (cms) record format
`dacct`	Disk usage data in `cacct.h` format

The `/usr/adm/acct/sum/rpt/`*MMDD/hhmm* directory contains the daily accounting report, `rprt`, which was generated on *MMDD* at *hhmm*.

The `/usr/adm/acct/work/`*MMDD/hhmm* directory is used as a work area during the processing of the accounting data. It contains the following files:

File	Description
`BAD.Wnqacct*`	Unprocessed NQS accounting data containing bad records (verified by `csaedit`)
`BAD.Wpacct*`	Unprocessed per-process accounting data containing bad records (verified by `csaedit`)
`BAD.Wtpacct*`	Unprocessed tape accounting data containing bad records (verified by `csaedit`)
`Ever.tmp`	Data verification work file
`Pctime*`	Preprocessed connect time data
`Pnqacct*`	Preprocessed NQS data
`Puptime*`	Uptimes
`Rctime0`	Recycled connect data to be used in the next accounting period

`Rnqacct0`	Recycled NQS data to be used in the next accounting period
`Rpacct0`	Recycled per-process accounting data to be used in the next accounting run
`Rtpacct0`	Recycled tape data to be used in the next accounting period
`Ruptime0`	Recycled uptimes to be used in the next accounting period
`Wctime*`	Verified, unprocessed connect time data
`Wdisktacct`	Disk accounting data (`cacct.h` format) created by `acctdisk`(8)
`Wdtmp`	Disk accounting report (ASCII) created by `diskusg`(8) or `acctdusg`(8)
`Wnqacct*`	Unprocessed NQS accounting data
`Wpacct*`	Unprocessed per-process accounting data
`Wtpacct*`	Unprocessed tape accounting data
`Wsoacct*`	Unprocessed socket accounting data
`Wwtmp*`	Unprocessed connect time data

The `/tmp/AC.`*MMDD*`/`*hhmm* directory contains the session record file, `Super-record`, which is generated on *MMDD* at *hhmm*.

The `/usr/lib/acct` directory contains the following programs and shell scripts used by CSA:

Program/script	Description
`csaaddc`	Merges consolidated (`cacct`) accounting files
`csabuild`	Creates a session file
`csacon`	Creates a consolidated (`cacct`) accounting file
`csaconvert`	Converts UNICOS 8.0, 8.3, 9.0, 9.1, 9.2, and 9.3 accounting file(s), both System V and CSA, to a 10.0 format
`csacrep`	Generates consolidated accounting reports
`csadrep`	Reports daemon usage based on the session file

`csaedit`	Verifies, deletes records, and prints various data files
`csafef`	Template to convert session files to an IBM front-end format
`csafef2`	Template to summarize session file records by the tuple user name, job ID, and account ID.
`csagcon`	Consolidates accounting data for `session` and `pacct` files
`csagfef`	Formats consolidated accounting data
`csaibm`	Template to convert session files to an IBM front-end format
`csajrep`	Generates job reports from a session file
`csaline`	Preprocesses connect time data (`/etc/wtmp`)
`csanqs`	Preprocesses NQS accounting data
`csapacct`	Verifies and deletes records from a `pacct` file
`csaperiod`	Performs periodic accounting
`csaperm`	Changes the group ID and permissions on the accounting files
`csarecy`	Recycles session data for unfinished sessions
`csarun`	Performs daily accounting
`csaswitch`	Enables or disables kernel and daemon accounting
`csaverify`	Verifies various data files
`getconfig`	Extracts values from the configuration file

The `/usr/lib/acct` directory may also contain the following programs if your site uses the accounting user exits:

Program/script	Description
`csa.archive1`	Site-generated user exit for `csarun`
`csa.archive2`	Site-generated user exit for `csarun`
`csa.fef`	Site-generated user exit for `csarun`

`csa.user`	Site-generated user exit for `csarun`

10.1.7.2 `/etc` directory

The `/etc` directory contains uptime and connect time data in the following files:

File	Description
`csaboots`	Captures system boot times
`csainfo`	Output file of `csaboots`
`wtmp`	Current connect time data

10.1.7.3 `/etc/config` directory

The `/etc/config` directory is the location of the `acct_config` file that contains the configurable parameters used by the accounting commands. These parameters can be changed by using the UNICOS installation and configuration menu system (the *menu system*). To see the `acct_config` file parameters, use the following menu selection:

```
UNICOS 8.0 Installation / Configuration Menu System
    ->Configure System
        ->Accounting Configuration
```

The main menu for accounting configuration is as follows:

```
    Mainframe Dependent Parameters ==>
    Accounting Start Parameters ==>
    Block Device SBUs ==>
    Character Device SBUs ==>
    Connect Time SBU ==>
    Multitasking CPU SBUs=>
    NQS SBUs ==>
    Pacct File SBUs ==>
    Tape SBUs ==>
    Miscellaneous Settings ==>
    Parameters for CSARUN and CSAPERIOD ==>
    Site Defined Settings ==>

    Import accounting configuration ...
    Activate accounting configuration ...
    Reload default accounting configuration ...
```

Online help for the `acct_config` parameters is available through the menu system.

The main menu for accounting configuration displays a table of `acct_config` parameters and the current values.

The `Import accounting configuration ...` option gets the local site accounting configuration.

The `Activate accounting configuration ...` option rewrites the `/etc/config/acct_config` file with the current values selected in the menus.

The `Reload default accounting configuration ...` option reloads the default values for the `acct_config` file from the released `/usr/src/skl/etc/config/acct_config` file.

10.1.8 CSA data processing

The flow of data among the various CSA programs is explained in this section and is illustrated in Figure 4.

Figure 4. CSA program data flow

1. Generate raw accounting files. Various daemons and system processes write to the raw accounting files. These accounting files include `pacct`, `nqacct`, `soacct`, `usacct`, `tpacct`, `wtmp`, and `csainfo`.
2. Create a fee file. Sites that want to charge fees to certain users can do so with the `chargefee`(8) command. `chargefee` creates a fee file that is processed by `csaaddc`(8).
3. Produce disk usage statistics. The `dodisk`(8) shell script allows sites to take snapshots of disk usage. `dodisk` does not report dynamic usage; it only reports the disk usage at the time the command was run. Disk usage is processed by `csaaddc`.
4. Preprocess selected raw accounting files. Generally, a data file that must be preprocessed contains multiple records for a session. These records are scattered throughout the file, and the processing of the records often depends upon other events that are logged in the accounting file (for example, system reboot). The preprocessor collapses information about a session into one output record.

NQS and connect time accounting data are preprocessed by `csanqs`(8) and `csaline`(8), respectively.

5. Organize the accounting data. `csabuild`(8) organizes the raw and preprocessed accounting data by sessions and boot times. With the exception of disk usage statistics and fees, the `csabuild` output file contains all of the accounting data available about each session.

   Sometimes data for terminated sessions is continually recycled. This can occur when accounting data is lost. To prevent data from recycling forever, edit `csarun` so that `csabuild` is executed with the `-o` *nday* option, which causes all sessions older than *nday* days to terminate. Select an appropriate *nday* value (see the `csabuild`(8) man page for more information).

6. Recycle information about unfinished sessions. Accounting data about uncompleted sessions is saved and processed again during the next accounting period. This information is recycled until the session completes or until manual intervention occurs. Accounting data for unfinished sessions is reported during each accounting period.

7. Generate the daemon usage report, which is appended to the daily report. `csadrep`(8) outputs information about interactive, NQS, tape, and socket usage.

8. Convert the session record file to a front-end format. Sites that process UNICOS accounting data on a front-end system can convert the session file to a format suitable for use on the front end by using the `csafef`(8), `csafef2`(8), or `csaibm`(8) command. These programs are templates, and you must modify them to suit your site's requirements. It is suggested that you use the user exit in the `FEF` section of `csarun` (see Section 10.1.5, page 207 and Section 10.1.10.3, page 241) to convert the session record file to your front-end format.

9. Generate command usage data. The information output by `acctcms`(8) is reported in the daily and periodic reports.

10. Consolidate the session record file. Session files are too large to retain on disk for any amount of time. Consequently, CSA consolidates the data and keeps the condensed version on disk. The accounting reports are based on the consolidated data. Data consolidation is done by `csacon`(8).

11. Generate an accounting report based on the consolidated data. `csacrep`(8) outputs the report.

12. Create the daily accounting report. The daily accounting report includes the following:

- Connect time statistics (step 4)
- Disk usage statistics (step 3)
- Unfinished session information (step 6)
- Command summary data (step 9)
- Consolidated accounting report (step 11)
- Last login information
- Daemon usage report (step 7)

13. Generate periodic accounting data. Periodic accounting data is an accumulation of the consolidated data created in step 10. `csaaddc`(8) merges condensed data files together. The resulting file contains accounting information for numerous accounting periods.
14. Generate periodic command usage data. `acctcms`(8) merges command usage data from multiple accounting periods. The usage information was created in step 9. Both an ASCII and a binary file are created.
15. Produce a periodic accounting report. `csacrep`(8) is used to generate a report based on a periodic accounting file.

Steps 4 through 12 are performed during each accounting period by `csarun`(8). Periodic accounting (steps 13 through 15) is initiated by the `csaperiod`(8) command. Daily and periodic accounting, as well as fee and disk usage generation (steps 2 through 3), can be scheduled by `cron`(8) to execute regularly. See Section 10.1.4, page 203, for more information.

10.1.9 Data recycling

A system administrator must correctly maintain recycled data in order to ensure accurate accounting reports. The following sections discuss data recycling and describe how an administrator can purge unwanted recycled accounting data.

Data recycling allows CSA to properly bill sessions that are active during multiple accounting periods. By default, `csarun`(8) reports data only for sessions that terminate during the current accounting period. Through data recycling, CSA preserves data for active sessions until the sessions terminate.

In the `Super-record` file, `csabuild`(8) flags each session as being either active or terminated. `csarecy`(8) reads the `Super-record` file and recycles data for the active sessions. `csacon`(8) consolidates the data for the terminated

sessions, which csaperiod(8) uses later. csabuild, csarecy, and csacon are all invoked by csarun.

csarun puts recycled data in the /usr/adm/acct/day directory. Data files with names suffixed with 0 contain recycled data. For example, ctime0, nqacct0, pacct0, tpacct0, usacct0, and uptime0 are generally the recycled data files that are found in /usr/adm/acct/day.

Normally, an administrator should not have to manually purge the recycled accounting data. This purge should only be necessary if accounting data is missing. Missing data can cause sessions to recycle forever and consume valuable CPU cycles and disk space.

10.1.9.1 How sessions are terminated

Interactive sessions, cron jobs, and at jobs terminate when the last process in the job exits. Normally, the last process to terminate is the login shell. The kernel writes an end-of-job (EOJ) record to the pacct file when the session terminates.

When the NQS daemon delivers an NQS request's output, the request terminates. The daemon then writes an NQ_DISP record type to the NQS accounting file, while the kernel writes an EOJ record to the pacct file.

Unlike interactive sessions, NQS requests can have multiple EOJ records associated with them. In addition to the request's EOJ record, there can be EOJ records for pipe clients, net clients, and checkpointed portions of the request. The pipe client and net client perform NQS processing on behalf of the request.

The csabuild command flags sessions in the Super-record file as being terminated if they meet one of the following conditions:

- The session is an interactive, cron, or at job, and there is an EOJ record for the job in the pacct file.
- The session is an NQS request, and there is both an EOJ record for the request in the pacct file and an NQ_DISP record type in the NQS accounting file.
- The session is an interactive, cron, or at job and is active at the time of a system crash.
- The session is manually terminated by the administrator using one of the methods described in Section 10.1.9.3, page 224.

10.1.9.2 Why recycled sessions should be scrutinized

Recycling unnecessary data can consume large amounts of disk space and CPU time. The session file and recycled data can occupy a vast amount of disk space on the file systems containing `/tmp` and `/usr/adm/acct/day`. Sites that archive data also require additional offline media. Wasted CPU cycles are used by `csarun` to reexamine and recycle the data. Therefore, to conserve disk space and CPU cycles, unnecessary recycled data should be purged from the accounting system.

Any of the following situations can cause CSA erroneously to recycle terminated sessions:

- Kernel or daemon accounting is turned off. At boot time, the `rc` command must execute `/usr/lib/acct/startup` in order to start kernel and daemon accounting.

  The kernel, `ckpacct`(8) command, or `ckdacct`(8) command can turn off accounting when there is not enough space on the file system containing `/usr/adm/acct/day`.

- Accounting files are corrupt. Accounting data can be lost or corrupted during a system or disk crash.

- Boot times are not recorded in `/etc/csainfo`. The `csaboots` command must be invoked by `rc` to write a boot time record to `/etc/csainfo`.

- Recycled data is erroneously deleted in a previous accounting period.

10.1.9.3 How to remove recycled data

Before choosing to delete recycled data, you should understand the repercussions, as described in Section 10.1.9.4, page 226. Data removal can affect billing and can alter the contents of the consolidated data file, which is used by `csaperiod`.

You can remove recycled data from CSA in the following ways:

- Interactively execute the `csarecy -A` command. Administrators can select the active sessions that are to be recycled by running `csarecy` with the `-A` option. Users are not billed for the resources used in the sessions terminated in this manner. Deleted data is also not included in the consolidated data file.

  The following example is one way to execute `csarecy -A` (which generates two accounting reports and two consolidated files):

1. Run `csarun` at the regularly scheduled time.
2. Edit a copy of `/usr/lib/acct/csarun`. Change the `-r` option on the `csarecy` invocation line to `-A`. Also, do not redirect standard output to `${CRPT}/recyrpt`. The result should be similar to the following:

   ```
   csarecy -A -s ${SESSION_FILE} \
   -N ${WORK}/Rnqacct -P ${WORK}/Rpacct \
   -T ${WORK}/Rtpacct -U ${WORK}/Ruptime \
   -C ${WORK}/Rctime -u ${WORK}/Rusacct \
   2> ${NITE}/Erec.${DTIME}
   ```

   Since both the `-A` and `-r` options write output to `stdout`, the `-r` option is not invoked and `stdout` is not redirected to a file. As a result, the recycled job report is not generated.
3. Execute the `jstat` command, as follows, to display a list of currently active jobs:

   ```
   jstat > jstat.out
   ```
4. Execute the `qstat` command to display a list of NQS requests. The `qstat` command is used for seeing whether there are requests that are not currently running. This includes requests that are checkpointed, held, queued, or waiting.

   In order to list all NQS requests, execute the `qstat` command, as follows, using a login that has either NQS manager or NQS operator privilege:

   ```
   qstat -a > qstat.out
   ```
5. Interactively run the modified version of `csarun`. If you execute `csarun` soon after the first step is complete, this invocation of `csarun` completes quickly because not very much data exists.

   For each active session, `csarecy` asks you if you want to preserve the session. Preserve the active and nonrunning NQS sessions found in the third and fourth steps. All other sessions are candidates for removal.

- Execute `csabuild` with the `-o` *ndays* option, which terminates all active sessions older than the specified number of days. Resource usage for these terminated sessions is reported by `csarun`, and users are billed for the sessions. The consolidated data file also includes this resource usage.

To execute `csabuild` with the `-o` option, edit `/usr/lib/acct/csarun`. Add the `-o` *ndays* option to the `csabuild` invocation line. Specify for *ndays* an appropriate value for your site.

Recycled data for currently active sessions will be removed if you specify an inappropriate value for *ndays*.

- Execute `csarun` with the `-A` option. It reports resource usage for both active and terminated sessions, so users are billed for recycled sessions. This data is also included in the consolidated data file.

  None of the data for the active sessions, including the currently active sessions, is recycled. No recycled data files are generated in the `/usr/adm/acct/day` directory.

- Remove the recycled data files from the `/usr/adm/acct/day` directory. You can delete data for all of the recycled sessions, both terminated and active, by executing the following command:

  ```
  rm /usr/adm/acct/day/*[a-z]0
  ```

  The next time `csarun` is executed, it will not find data for any recycled sessions. Thus, users are not billed for the resources used in the recycled sessions, and this data is not included in the consolidated data file. `csarun` recycles the data for currently active sessions.

10.1.9.4 Adverse effects of removing recycled data

CSA assumes that all necessary accounting information is available to it, which means that CSA expects kernel and daemon accounting to be enabled and recycled data not to have been mistakenly removed. If some data is unavailable, CSA may provide erroneous billing information. Sites should be aware of the following facts before removing data:

- Users may or may not be billed for terminated recycled sessions. Administrators must understand which of the previously described methods cause the user to be billed for the terminated recycled sessions. It is up to the site to decide whether or not it is valid for the user to be billed for these sessions.

  For those methods that cause the user to be billed, both `csarun` and `csaperiod` report the resource usage.

- It may be impossible to reconstruct a terminated recycled session. If a recycled session is terminated by the administrator, but the session actually terminates in a later accounting period, information about the session is lost.

If a user questions the resource billing, it may be extremely difficult or impossible for the administrator to correctly reassemble all accounting information for the session in question.

- Manually terminated recycled sessions be improperly billed in a future billing period. If the accounting data for the first portion of a session has been deleted, CSA may be unable to correctly identify the remaining portion of the job. Errors may occur, such as NQS requests being flagged as interactive sessions, or NQS requests being billed at the wrong queue rate. This is explained in detail in Section 10.1.9.5, page 228.
- CSA programs may detect data inconsistencies. When accounting data is missing, CSA programs may detect errors and abort.

The following table summarizes the effects of using the methods described in Section 10.1.9.3, page 224.

Table 16. Possible effects of removing recycled data

Method	Underbilling?	Incorrect billing?	Consolidated data file
`csarecy -A`	Yes. Users are not billed for the portion of the session that was terminated by `csarecy -A`.	Possible. Manually terminated recycled sessions may be billed improperly in a future billing period.	Does not include data for sessions terminated by `csarecy -A`.
`csabuild -o`	No. Users are billed for the portion of the session that was terminated by `csabuild -o`.	Possible. Manually terminated recycled sessions may be billed improperly in a future billing period.	Includes data for sessions terminated by `csabuild -o`.
`csarun -A`	No. All active and recycled sessions are billed.	Possible. All active and recycled sessions that eventually terminate may be billed improperly in a future billing period, because no data is recycled.	Includes data for all active and recycled sessions.
`rm`	Yes. All users are not billed for the portion of the session that was recycled.	Possible. All recycled sessions that eventually terminate may be billed improperly in a future billing period.	Does not include data for any recycled session.

By default, the consolidated data file contains data only for terminated sessions. Manual termination of recycled data may cause some of the recycled data to be included in the consolidated file. These cases are noted in the previous table.

10.1.9.5 NQS requests and recycled data

In order for CSA to identify all NQS requests, data must be properly recycled. When an administrator manually purges recycled data for an NQS request, errors such as the following can occur:

- CSA flags the NQS request as an interactive session. This causes the request to be billed at interactive rates.
- The request is billed at the wrong queue rate.
- The wrong queue wait time is associated with the request.

These errors occur because valuable NQS accounting information was purged by the administrator. Only a few NQS accounting records are written by the NQS daemon, and all of the records are needed for CSA to properly bill NQS requests.

NQS accounting records are only written under the following circumstances:

- The NQS daemon receives a request.
- A request is routed to a queue.
- A request executes. This includes executing a request for the first time, and restarting and rerunning a request.
- A request terminates. A request can terminate because it is completed, requeued, preempted, held, checkpointed, or rerun by the operator.
- Output is delivered.

Thus, for long running requests that span days, there can be days when no NQS data is written. Consequently, it is extremely important that accounting data be recycled. If the site administrator manually terminates recycled sessions, care must be taken to be sure that only nonexistent NQS requests are terminated.

10.1.10 Tailoring CSA

This section describes the following actions in CSA:

- Setting up SBUs
- Setting up daemon accounting
- Setting up user exits
- Modifying the front-end formatting templates
- Modifying the charging of NQS jobs based on NQS termination status
- Tailoring CSA shell scripts
- Using `at`(1) instead of `cron`(8) to periodically execute `csarun`
- Allowing users without super-user permissions to execute CSA
- Using an alternate configuration file

10.1.10.1 System billing units (SBUs)

A *system billing unit* (SBU) is a unit of measure that reflects use of machine resources. You can alter the weighting factors associated with each field in each accounting record to obtain an SBU value suitable for your site. SBUs are defined in the accounting configuration file, `/etc/config/acct_config`. By default, all SBUs are set to `0.0`.

The source code for the default SBU calculations is located in `/usr/src/cmd/acct/lib/acct/sbu.c`. For sites that do not have source code, the default algorithms are also defined in `/usr/src/cmd/acct/lib/acct/user_sbu.c`. By modifying `/usr/src/cmd/acct/lib/acct/user_sbu.c`, compiling, and relinking the accounting programs, your site can use local SBU calculations.

Accounting allows different periods of time to be designated either prime or nonprime time (the time periods are specified in `/usr/lib/acct/holidays`).

Following is an example of how the prime/nonprime algorithm works:

Assume a user uses 10 seconds of CPU time, and executes for 100 seconds of prime wall-clock time, and pauses for 100 seconds of nonprime wall-clock time. Therefore, elapsed time is 200 seconds (100+100). If

prime = *prime time* / *elapsed time*
nonprime = *nonprime time* / *elapsed time*
cputime[PRIME] = *prime* * *CPU time*
cputime[NONPRIME] = *nonprime* * *CPU time*

then

cputime[PRIME] == 5 seconds
cputime[NONPRIME] == 5 seconds

Under CSA, an SBU value is associated with each record in the Session record file when that file is assembled by `csabuild`(8). Final summation of the SBU values is done by `csacon`(8) during the creation of the `cacct` record file.

Billing for SBU values is intended to be a combination of all the SBU values from each record associated with a job, as follows:

*Total SBU = (NQS queue SBU value) * (sum of all pacct record SBUs*
+ sum of all tape record SBUs
+ sum of all ctmp record SBUs)

This allows a site to bill different NQS queues at differing rates. Again, if the available formulas are insufficient to achieve the site's requirements, a site can

modify the calculations found in the `sbu` library routine, `/usr/src/cmd/acct/lib/acct/user_sbu.c`.

10.1.10.1.1 `pacct` SBUs

The SBUs for `pacct` data are separated into prime and nonprime values. Prime and nonprime use is calculated by a ratio of elapsed time. If you do not want to make a distinction between prime and nonprime time, set the nonprime time SBUs and the prime time SBUs to the same value. Prime time is defined in `/usr/lib/acct/holidays`. By default, Saturday and Sunday are considered nonprime time.

The following is a list of prime time `pacct` SBU weights. Descriptions and factor units for the nonprime time SBU weights are similar to those listed here. SBU weights are defined in `/etc/config/acct_config`.

Value	Description
`P_BASIC`	Prime-time weight factor. `P_BASIC` is multiplied by the sum of prime time SBU values to get the final SBU factor for the `pacct` base record.
`P_TIME`	General-time weight factor. `P_TIME` is multiplied by the time SBUs (made up of `P_STIME`, `P_ITIME`, `P_SCTIME`, and `P_INTTIME`) to get the time contribution to the `pacct` base record SBU value.
`P_STIME`	System CPU-time weight factor. The unit used for this weight is *billing units* per second. `P_STIME` is multiplied by the system CPU time to get the system CPU factor.
`P_UTIME`	User CPU-time weight factor. The unit used for this weight is *billing units* per second. `P_UTIME` is multiplied by the user CPU time to get the user CPU factor. User time is the sum of user times after weighting for multitasking. Multitasking may affect the user CPU cost if the `MUTIME_WEIGHT` parameters have been set to values other than 1.0. See the following explanation of these values.
`P_ITIME`	This is the weight factor for the time spent waiting in the kernel for I/O while the process is

	locked in memory. The unit used for this weight is *billing units* per second. P_ITIME is multiplied by the I/O wait time.
P_SCTIME	Weight factor for system call time. The unit used for this weight is *billing units* per second.
P_INTTIME	Weight factor for interrupt time. The unit used for this weight is *billing units* per second.
P_MEM	General-memory-integral weight factor. P_MEM is multiplied by the memory SBUs (made up of P_XMEM and P_IMEM) to get the memory contribution to the pacct base record SBU value.
P_XMEM	CPU-time-memory-integral weight factor. The unit used for this weight is *billing units* per Kword-minute. P_XMEM is multiplied by the memory integral (see Section 10.1.12.1, page 252). This value is affected by your site's choice of MEMINT (in the accounting configuration file /etc/config/acct_config).
P_IMEM	The weight factor used with the I/O wait time memory integral. This integral includes the I/O wait time while the process is locked in memory. The unit used for this weight is *billing units* per Kword-minute. P_IMEM is multiplied by the I/O-wait-time memory integral.
P_IO	General-I/O weight factor. P_IO is multiplied by the I/O SBUs (made up of P_BYTEIO, P_PHYIO, and P_LOGIO) to get the I/O contribution to the pacct base record SBU value.
P_BYTEIO	Characters-transferred weight factor. The unit used for this weight is *billing units* per character transferred. P_BYTEIO is multiplied by the bytes of I/O transferred. If tape or device I/O is to be charged at a rate other than P_BYTEIO, the tape and device weight factors need to be adjusted accordingly. See Section 10.1.11.1, page 246 (field ac_io), for more information.

P_PHYIO	Physical-I/O-request weight factor. The unit used for this weight is *billing units* per physical I/O request. P_PHYIO is multiplied by the number of physical I/O requests made. Physical requests are the number of driver requests made.
P_LOGIO	Logical-I/O-request weight factor. The unit used for this weight is *billing units* per logical I/O request. P_LOGIO is multiplied by the number of logical I/O requests made. The number of logical I/O requests is the total number of read(2), write(2), reada(2), and writea(2) system calls. The number of strides, multiplied by the number of requests processed by each listio(2) call, is also added to the total.

10.1.10.1.2 Multitasking SBUs

The MUTIME_WEIGHT *i* variables define the weighting factors that are used to charge user CPU time for multitasking programs. It is used in conjunction with the ac_mutime array (see /usr/include/sys/acct.h), which defines the amount of CPU time the multitasking program spent with *i* + 1 CPUs connected.

MUTIME_WEIGHT *i* defines the marginal cost of getting the *i* + 1 CPU at one instant. If these values are set to less than 1.0, there is an incentive for multitasking. If the values are set to 1.0, multitasking programs are charged for user CPU time just as all other programs.

For more information on multitasking incentives, see Section 10.1.12, page 251.

10.1.10.1.3 SDS SBUs

(On all Cray Research systems except the CRAY EL series) Secondary data storage (SDS) system billing units are calculated from the statistics on SDS use in the pacct file. The SBU factors are defined in /etc/config/acct_config.

The values are as follows:

Value	Description
NP_SDSMEM or P_SDSMEM	SDS-memory-integral weight factor. The memory integral is based on residency time and not on execution time. P_SDSMEM

or NP_SDSMEM is multiplied by the SDS memory integral. The unit used for this weight is *billing units* per Mword-second.

NP_SDSLOGIO or P_SDSLOGIO

SDS-logical-I/O-request weight factor. P_SDSLOGIO or NP_SDSLOGIO is multiplied by the number of SDS logical I/O requests. The unit used for this weight is *billing units* per logical I/O request.

NP_SDSBYTEIO or P_SDSBYTEIO

SDS-characters-transferred weight factor. P_SDSBYTEIO or NP_SDSBYTEIO is multiplied by the number of SDS characters transferred. The unit used for this weight is *billing units* per character transferred.

The SBU values should be very small. On Cray Research systems, it is possible to submit a very large number of requests to SDS in a short time; therefore, to prevent these numbers from dominating the SBU values, small weight factors must be used. Values of 0 result in no charge.

10.1.10.1.4 MPP SBUs

Massively parallel processing (MPP) system billing units are calculated from the statistics on MPP use in the pacct file. The SBU factors are defined in /etc/config/acct_config.

The P_MPPPE or NP_MPPPE SBUs are the MPP processing elements (PEs) weight factors, prime and nonprime charges. The prime time billing units for PEs is calculated using the following equation:

$$\texttt{P\_MPPPE billing units} = \texttt{P\_MPPPE} * \sum_{0}^{\#\ of\ sessions} (no.\ MPP\ PEs\ used * MPP\ time\ used)$$

The nonprime time billing units for PEs is calculated using the following equation:

$$\texttt{NP\_MPPPE billing units} = \texttt{NP\_MPPPE} * \sum_{0}^{\#\ of\ sessions} (no.\ MPP\ PEs\ used * MPP\ time\ used)$$

The unit used for these weights is billing units per PE-second.

The P_MPPBB or NP_MPPBB SBUs are the MPP barrier bits weight factors, prime and nonprime charges.[1] The prime time billing units for barrier bits is calculated using the following equation:

$$\texttt{P\_MPPBB billing units} = \texttt{P\_MPPBB} * \sum_{0}^{\#\ of\ sessions} (no.\ MPP\ barrier\ bits\ used * MPP\ time\ used)$$

The nonprime time billing units for barrier bits is calculated using the following equation:

$$\texttt{NP\_MPPBB billing units} = \texttt{NP\_MPPBB} * \sum_{0}^{\#\ of\ sessions} (no.\ MPP\ barrier\ bits\ used * MPP\ time\ used)$$

The unit used for these weights is *billing units* per barrier bit-second.

The P_MPPTIME or NP_MPPTIME SBUs are the MPP time weight factors, prime and nonprime charges. The prime time billing units for MPP time is calculated using the following equation:

$$\texttt{P\_MPPTIME billing units} = \texttt{P\_MPPTIME} * \sum_{0}^{\#\ of\ sessions} (MPP\ time\ used)$$

The nonprime time billing units for MPP time is calculated using the following equation:

$$\texttt{NP\_MPPTIME billing units} = \texttt{NP\_MPPTIME} * \sum_{0}^{\#\ of\ sessions} (MPP\ time\ used)$$

The unit used for these weights is *billing units* per second.

The SBU values should be very small, which will prevent these numbers from dominating the SBU values. Values of 0 result in no charge.

10.1.10.1.5 Connect time SBUs

There are SBUs for both prime- and nonprime-time connect data. The SBU values should reflect the system billing units per second of connect time. The weight factors, CON_PRIME and CON_NONPRIME, are defined in /etc/config/acct_config.

10.1.10.1.6 NQS SBUs

The /etc/config/acct_config file contains the configurable parameters that pertain to NQS SBUs.

---

1 Deferred implementation.

The NQS_NUM_QUEUES parameter sets the number of queues for which you want to set SBUs (the value must be set to at least 1). Each NQS_QUEUE *x* variable in the configuration file has a queue name and an SBU pair associated with it (the total number of queue/SBU pairs must equal NQS_NUM_QUEUES). The queue/SBU pairs define weights for the queues. If an SBU value is less than 1.0, there is an incentive to run jobs in the associated queue; if the value is 1.0, jobs are charged as though they are non-NQS jobs; and if the SBU is 0.0, there is no charge for jobs running in the associated queue. SBUs for queues not found in the configuration file are automatically set to 1.0.

The NQS_NUM_MACHINES parameter sets the number of originating machines for which you want to set SBUs (the value must be at least 1). Each NQS_MACHINE *x* variable in the configuration file has an originating machine and an SBU pair associated with it (the total number of machine/SBU pairs must equal NQS_NUM_MACHINES). SBUs for originating machines not specified in /etc/config/acct_config are automatically set to 1.0.

The queue and machine SBUs are multiplied together to give an NQS multiplier. If the SBUs are set to less than 1.0, there is an incentive to run jobs in these queues or from these machines. SBUs of 1.0 indicate that jobs in the queues or from associated hosts are billed normally.

10.1.10.1.7 Socket SBUs

Currently, there is no way to charge for socket accounting. The socket accounting records produced are only processed in order to make the data available to the site-supplied user exits.

10.1.10.1.8 Tape SBUs

There is a set of weighting factors for each group of tape devices. By default, there are only two groups, tape and cart. The TAPE_SBU*i* parameters in /etc/config/acct_config define the weighting factors for each group. There are SBUs associated with the following:

- Number of mounts
- Device reservation time (seconds)
- Number of bytes read
- Number of bytes written

10.1.10.1.9 Device SBUs

Device accounting system billing units are calculated from the device statistics in the `pacct` file. SBUs can be set for both block and character devices in `/etc/config/acct_config`. The fields in the `acct_config` file that affect SBU factors for each device are as follows:

SBU factor	Description
`Logical I/O Sbu`	Weight given to each logical I/O request.
`Characters Xfer Sbu`	Weight given to the amount of data transferred.
`Device Name`	Device type name (see Section 10.1.14, page 254).

The `Logical I/O Sbu` factor is multiplied by the number of system calls that initiated I/O on a device type. The `Characters Xfer Sbu` factor is multiplied by the number of bytes of data transferred to a device type.

The SBUs for block devices are labeled `BLOCK_DEVICE` *x*, where *x* is a number from `0` to `MAXBDEVNO-1`. Character devices are labeled `CHAR_DEVICE` *x*, where *x* is a number from `0` to `MAXCDEVNO-1`. The numeric suffixes for the `CHAR_DEVICE` *x* variables must match the minor numbers in `/dev`, which are defined in `/usr/src/uts/c1/cf/devsw.c` in the `cdevsw[]` array.

`MAXBDEVNO` and `MAXCDEVNO` are located in the `/usr/include/sys/param.h` include file and have default values of 10 and 35, respectively.

Device accounting is also discussed in Section 10.1.14, page 254.

The SBU values should be very small. On Cray Research systems, it is possible to perform a very large number of I/O requests very quickly; therefore, to prevent these numbers from dominating the SBU values, a small weight factor must be used. A value of 0 results in no charge.

10.1.10.1.10 Example SBU settings

The following section provides an example showing how you could set up the SBU system. This example is restricted to `pacct` base records (you should also consider `pacct` multitasking, `pacct` I/O (device accounting), and all the daemon records). In this example, it is assumed that an SBU is equal to one dollar of charge.

The formula for calculating the whole `pacct` base record SBU value is as follows:

```
PSBU = ((P_TIME * (P_STIME * stime + P_UTIME * utime + P_ITIME *
iowtime)) + (P_MEM * (P_XMEM * cpumem + P_IMEM * iowmem)) +
(P_IO * (P_BYTEIO * bytes + P_PHYIO * phy + P_LOGIO * log)));

NSBU = ((NP_TIME * (NP_STIME * stime + NP_UTIME * utime + NP_ITIME
* iowtime)) + (NP_MEM * (NP_XMEM * cpumem + NP_IMEM * iowmem)) +
(NP_IO*(NP_BYTEIO * bytes + NP_PHYIO * phy + NP_LOGIO * log)));

SBU = P_BASIC * PSBU + NP_BASIC * NSBU;
```

The variables in this formula are as follows:

Variable	Description
stime	System CPU time in seconds.
utime	User CPU time in seconds. User CPU time is the sum of user times after weighting for multitasking.
iowtime	Time (in seconds) spent waiting in the kernel for I/O while the process is locked in memory.
cpumem	Memory integral (see Section 10.1.12.1, page 252).
iowmem	I/O-wait-time memory integral.
bytes	Number of bytes of data transferred.
phy	Number of physical I/O requests made.
log	Number of logical I/O requests made.

All time is considered prime time. Therefore, the nonprime time SBUs should be set to the same values as their prime time counterparts.

In order to produce a billing that is somewhat repeatable, this example omits various values, such as physical I/O (set `P_PHYIO` to `0.0`), that depend on the state of the machine at run time. In particular, system time varies greatly due to system load and will cause this example to be nonrepeatable. Information on which fields generate repeatable values is contained in Section 10.1.11.1, page 246.

In this example, users are charged for each logical request (`P_LOGIO`) and the total data moved (`P_BYTEIO`). This provides users with an incentive to use larger I/O requests, which may be more efficient. Processes that perform I/O

that locks them into memory are penalized (P_IMEM), because this may result in memory fragmentation.

In this example, users are charged the following amounts for time (the accounting record fields associated with the charge are also identified):

- $100 per hour of user CPU time. This is equal to $100 per 3600 seconds, which is $0.02777777 per second (P_UTIME). To produce repeatable billing, system time must be excluded. Thus, P_STIME is set to 0.0.
- $25 for each megaword of memory per hour of CPU time. The memory integrals are in units of Kword-minutes, so the weighting factor is $25/(60 minutes * 2^{10} Kwords) or 0.0004069010 (P_XMEM).
- $3 for each hour spent waiting on I/O while locked into memory. The wait time is in units of seconds. so the weighting factor is $3/3600 seconds or .0008333333 (P_ITIME).
- $25 for I/O wait time (locked in memory) per hour. This is the same value as the memory charge because the process is using memory during this time in the same way it would when executing. The weighting factor is $25/(60 seconds * 2^{10} Kwords) or 0.0004069010 (P_IMEM).
- A DD-49 disk drive can perform I/O at a maximum rate of 9.6 Mbytes per second. Assume that the original cost of the drive was $125,000, and it will be paid for in 2 years. Also assume that it is busy 5% of the time (63072000 seconds * 5% = 3153600 seconds). The amount of I/O that can be completed in 2 years is 31745177026560 bytes (9.6 Mbytes/second * 3153600 seconds). Thus, you would charge $125,000/31745177026560 bytes or $0.00000000393760 per byte, which is approximately $0.33/10 Mwords (P_BYTEIO).
- $0 for physical I/O requests. This charge makes the billing more repeatable. The byte I/O charge covers this activity (P_PHYIO).
- $0.01 per thousand logical I/O requests. This charge encourages the user to perform larger I/O requests by charging less for a lower number of larger I/O requests (instead of a lot of small I/O requests). The weighting factor is computed as $0.01/1000 I/O requests or 0.00001 (P_LOGIO).

Therefore, in this example, the pacct base record charges are as follows (the nonprime time SBUs are set to the same value as their prime time counterparts):

Weight factor	Charge
P_BASIC	1.0

P_TIME	1.0
P_UTIME	0.02777777777777
P_STIME	0.0
P_ITIME	0.00083333333333
P_MEM	1.0
P_XMEM	0.00040690104166
P_IMEM	0.00040690104166
P_IO	1.0
P_BYTEIO	0.00000000393760
P_PHYIO	0.0
P_LOGIO	0.00001000000000

P_BASIC, P_TIME, P_MEM, and P_IO are used to weight different factors of the equation; you can use these depending on how your other groups of weighting factors are picked. For example, you could change the P_IO and P_BYTEIO factors as follows and receive the same results:

Weight factor	Charge
P_BASIC	1.0
P_TIME	1.0
P_UTIME	0.02777777777777
P_STIME	0.0
P_ITIME	0.00083333333333
P_MEM	1.0
P_XMEM	0.00040690104166
P_IMEM	0.00040690104166
P_IO	0.00001
P_BYTEIO	0.000393760
P_PHYIO	0.0

P_LOGIO	1.0

10.1.10.2 Daemon accounting

Accounting information is available from NQS, online tapes, and sockets. Data is written to the `nqacct`, `tpacct`, and `soacct` files, respectively, in the `/usr/adm/acct/day` directory.

In most cases, daemon accounting must be enabled by both the CSA subsystem and the daemon. Section 10.1.4, page 203, describes how to enable daemon accounting at system startup time. You can also enable daemon accounting after the system has booted.

You can enable accounting for a specified daemon with the `turndacct`(8) command. For example, to start tape accounting, you would execute the following:

```
/usr/lib/acct/turndacct on tape
```

The NQS and online tape daemon also must enable accounting. Use the `qmgr set accounting on` command to turn on NQS accounting. Tape daemon accounting is enabled when `tpdaemon`(8) is executed with the `-c` option.

Daemon accounting is disabled by `shutacct`(8) at system shutdown (see Section 10.1.4, page 203). It can also be disabled at any time by the `turndacct`(8) command when used with the `off` operand. For example, to disable NQS accounting, execute the following command:

```
/usr/lib/acct/turndacct off nqs
```

New daemon accounting files can be started when `turndacct` is invoked with the `switch` operand. No data is lost when files are switched. For example, to start a new NQS accounting file, execute the following command:

```
/usr/lib/acct/turndacct switch nqs
```

10.1.10.3 Setting up user exits

CSA accommodates the following user exits, which can be called from certain `csarun` states:

`csarun` state	User exit
ARCHIVE1	/usr/lib/acct/csa.archive1
ARCHIVE2	/usr/lib/acct/csa.archive2

FEF	/usr/lib/acct/csa.fef
USEREXIT	/usr/lib/acct/csa.user

These exits allow an administrator to tailor the csarun procedure to the individual site's needs by creating scripts to perform additional site-specific processing during daily accounting.

While executing, csarun checks in the ARCHIVE1, ARCHIVE2, FEF, and USEREXIT states for a shell script with the appropriate name.

If the script exists, it is executed via the shell . (dot) command. If the script does not exist, the user exit is ignored. The . (dot) command will not execute a compiled program, but the user exit script can. csarun variables are available, without being exported, to the user exit script. csarun checks the return status from the user exit and, if it is nonzero, the execution of csarun is terminated.

If CSA is run by a user without super-user permissions, the user exits must be both readable and executable by this user (see page Section 10.1.10.7, page 244).

10.1.10.4 Charging for NQS jobs

By default, SBUs are calculated for all NQS jobs regardless of the job's NQS termination code. If you do not want to bill portions of an NQS request, set the appropriate NQS_TERM_ *xxxx* variable (termination code) in /etc/config/acct_config to 0, which sets the SBU for this portion to 0.0. By default, all portions of a request are billed.

The following table describes the termination codes:

Code	Description
NQS_TERM_EXIT	Generated when the request finishes running and is no longer in a queued state. At NQS shutdown time, requests that specified both the -nc (no checkpoint) and -nr (no rerun) options for qsub also have NQS_TERM_EXIT records written. In addition, this record is written for requests that specified the -nr option for qsub and were running at the time of a system crash.
NQS_TERM_REQUEUE	Written for running requests that are checkpointed and then requeued when NQS shuts down.
NQS_TERM_PREEMPT	Written when a request is preempted with the qmgr preempt request command.

NQS_TERM_HOLD	Written for a request that is checkpointed with the `qmgr hold request` command. The `hold request` command differs from the checkpoint done at daemon shutdown time because a "hold" keeps the job from being scheduled until a `qmgr release` command is executed.
NQS_TERM_OPRERUN	Written when a request is rerun with the `qmgr rerun request` command. At NQS shutdown time, jobs that cannot be checkpointed and do not have the `-nr` (no rerun) option for `qsub` specified have this type of termination record written. The requests are requeued with this status.

10.1.10.5 Tailoring CSA shell scripts and commands

Modify the following variables in `/etc/config/acct_config` if necessary:

Variable	Description
ACCT_FS	File system on which `/usr/adm/acct` resides. The default is `/usr`.
MAIL_LIST	List of users to whom mail is sent if fatal errors are detected in the accounting shell scripts. The default is `root` and `adm`.
WMAIL_LIST	List of users to whom mail is sent if warning errors are detected by the `csarun` script at cleanup time. The default is `root` and `adm`.
MIN_BLKS	Minimum number of free blocks needed in `${ACCT_FS}` to run `csarun` or `csaperiod`. The default is 500 free blocks.

10.1.10.6 Using `at` to execute `csarun`

You can use the `at`(1) command instead of `cron`(8) to execute `csarun` periodically. If your Cray Research system is down when `csarun` is scheduled to run via `cron`, `csarun` will not be executed until the next scheduled time. On the other hand, `at` jobs execute when the machine reboots if their scheduled execution time was during a down period.

You can execute `csarun` with `at` in several ways. For instance, a separate script can be written to execute `csarun` and then resubmit the job at a specified time. Also, an `at` invocation of `csarun` could be placed in a user exit script, `/usr/lib/acct/csa.user`, that is executed from the `USEREXIT` section of `csarun`. See Section 10.1.10.3, page 241, for more information.

10.1.10.7 Allowing nonsuper users to execute CSA

Your site may want to allow users without super-user permissions to run CSA accounting. CSA can be run by users who are in the group `adm` and have permission bit `acct` set in their UDB entries.

Note: If `root` has run CSA, you must execute the shell script `/usr/lib/acct/csaperm` (see `csaperm`(8)) to change the group ID and file permissions of all accounting files in `/usr/adm/acct` so they can be accessed by a nonsuper user running CSA.

The following steps describe the process of setting up CSA so it is executed automatically on a daily basis by a user without super-user permissions. In this example, the user without super-user permissions is `adm`:

1. Ensure that user `adm` is a member of group `adm` and has the permission bit `acct` set in its UDB entry (see `udbgen`(8)).
2. As `root`, execute the shell script `csaperm` to change the group ID and file permissions of all accounting files in `/usr/adm/acct` so they can be accessed by a nonsuper user.
3. Ensure that, if they exist, the user exits `/usr/lib/acct/csa.archive1`, `/usr/lib/acct/csa.archive2`, `/usr/lib/acct/csa.fef`, and `/usr/lib/acct/csa.user` have the group ID `adm` and are both readable and executable by group `adm`.
4. Follow steps 1 through 5 of Section 10.1.4, page 203, to set up system billing units, record system boot times, and turn off accounting before system shutdown.
5. Include an entry similar to the following in `/usr/spool/cron/crontabs/root` so that `cron`(8) automatically runs `dodisk`(8):

   ```
   0 3 * * 1-6 /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dk2log
   ```

   `dodisk` must be executed by `root`, because no other user has the correct permissions to read `/dev/dsk/*`.

6. Include entries similar to the following in `/usr/spool/cron/crontabs/adm` so that user `adm` automatically runs daily accounting by using `cron`:

```
0 4 * * 1-6 /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
0 * * * * /usr/lib/acct/ckdacct nqs tape
0 * * * * /usr/lib/acct/ckpacct
```

   `csarun`(8) should be executed at a time that allows `dodisk` to complete. If `dodisk` does not complete before `csarun` executes, disk accounting information may be missing or incomplete.

7. To run periodic accounting, place an entry similar to the following in `/usr/spool/cron/crontabs/adm` (this command generates a periodic report on all consolidated data files found in `/usr/adm/acct/sum/data/*` and then deletes those data files):

```
15 5 1 * * /usr/lib/acct/csaperiod -r 2>/usr/adm/acct/nite/pd2log
```

8. Update the holidays file as described in Section 10.1.4, page 203.

10.1.10.8 Using an alternate configuration file

By default, the `/etc/config/acct_config` configuration file is used when any of the CSA commands are executed. You can specify a different file by setting the shell variable `ACCTCONFIG` to another configuration file, and then executing the CSA commands.

For example, you would execute the following commands in order to use the configuration file `/tmp/myconfig` while executing `csarun`(8):

```
ACCTCONFIG=/tmp/myconfig /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
```

10.1.10.9 Disk usage reporting (`diskusg`)

The `diskusg`(8) command can be configured at your site. The `site.c` module of `diskusg` contains an example to help you in customizing a report for your site. You can delete your choice of comment-protection characters in the example, compile the routine, relink `diskusg`, then print a sample report of disk usage for your site. You can execute your modified `diskusg` command in the `USEREXIT` state in `csarun` or `runacct` scripts.

10.1.11 Per-process accounting data

This section describes some of the fields found in the `pacct` file. `/usr/include/sys/acct.h` defines the structure of this file.

10.1.11.1 Base accounting record

One base accounting record per process is written; each record contains the following fields:

Field	Description
`ac_flag`	Accounting flags; may be any of the flags defined in `sys/acct.h`. `FORK` means that the process forked but did not `exec`. `ASU` means that the process used super-user privileges.
`ac_stat`	Low-order 8 bits from the process's exit value. See the `wait`(2) man page for more information.
`ac_uid`	Real user ID.
`ac_gid`	Real group ID.
`ac_tty`	Controlling tty device.
`ac_btime`	Start time of the process (in seconds).
`ac_utime`	User CPU time used (in clocks). This number is repeatable for nonmultitasked jobs (within the limitations caused by memory conflicts).
`ac_stime`	System CPU time used (in clocks). This number is not repeatable, because it varies with system load.
`ac_etime`	Elapsed time while process executed (in clocks). This number is not repeatable.
`ac_mem`	Memory integral selected when `MEMINT = 1` (in clicks-ticks). (`MEMINT` is located in `/etc/config/acct_config`.) This is the only constant memory integral available (within the limitations caused by memory conflicts); therefore, if repeatable billing is required, this number must be used. See Section 10.1.12.1, page 252, for more information.

`ac_mem2`	Memory integral selected when `MEMINT = 2` (in clicks-ticks). (`MEMINT` is located in `/etc/config/acct_config`.) This integral is not constant and varies with machine load. See Section 10.1.12.1, page 252, for more information.
`ac_mem3`	Memory integral selected when `MEMINT = 3` (in clicks-ticks). (`MEMINT` is located in `/etc/config/acct_config`.) This integral is not constant and varies with machine load. See Section 10.1.12.1, page 252, for more information.
`ac_io`	Number of characters transferred by the process. If any tape accounting information existed for this process, the number of tape bytes read and written is included in the `ac_io` field. Thus, the amount of tape I/O is reported twice: once in the `ac_io` field and again in the tape accounting record. The `ac_io` field generally is larger, because it includes additional I/O performed by the process. This number is repeatable. Device accounting I/O information is also reported twice: by `ac_io` and in the device accounting record field `acd_ioch`. Charges for doing I/O to tape or to a particular device can be adjusted by setting the SBU weight factors for tape and device I/O. These weights are defined in `/etc/config/acct_config`. The tape SBUs are labeled `TAPE_SBU` *x*, and the device SBUs are `BLOCK_DEVICE` *x* and `CHAR_DEVICE` *x*. Set the weight factors relative to `P_BYTEIO` (see Section 10.1.10.1.8, page 236, and Section 10.1.10.1.9, page 237). The `ac_io` value is multiplied by `P_BYTEIO`. The tape or device I/O value is multiplied by the appropriate tape or device weight factor. For example, if a surcharge is to be applied to tape I/O, the read and write values for the `TAPE_SBU` *x* variables must reflect the amount over `P_BYTEIO` that should be charged.

`ac_rw`	Number of physical I/O requests initiated by the process. This number varies due to conditions in the system buffer cache. Therefore, if repeatable billing is desired, this number cannot be used.
`ac_iowtime`	I/O wait time (in clocks) measured while the process is locked in memory. This means that system buffered I/O does not appear here. Also, this is a measure of the time elapsed from when a process is removed from the run queue until the process is reconnected to a CPU; therefore it may vary due to system load.
`ac_iowmem`	I/O-wait-time memory integral measured while the process is locked in memory (in click-ticks). This number may vary due to system load.
`ac_iosw`	Swap count. This number may vary due to system load.
`ac_lio`	Logical I/O request count; this is a count of the `read`, `write`, `reada`, `writea`, and `listio` (list entries) system calls made. This number is repeatable.
`ac_pid`	Process ID.
`ac_ppid`	Parent process ID.
`ac_ctime`	Process raw connect time in clocks.
`ac_acid`	Account ID.
`ac_jobid`	Job ID.
`ac_nice`	Nice value, measured at the end of 1 second of system and user time or the most expensive value used thereafter. This allows a process to set the value at which most of its work should be done; only charges for increased cost are levied. The 1 second of system and user time is calculated from the billable time, not from the time actually connected. Therefore, because billable time is calculated each time a process is rescheduled, or one time per second, the nice value becomes fixed some time in the interval from 1 to 2 seconds of connect time. (Connect time includes semaphore

	wait, except on CRAY C90 systems; billable time does not includes semaphore wait.)
`ac_comm`	Command name (first 8 characters).
`ac_iobtim`	I/O wait time in clocks measured while the process is not locked in memory (unlike `ac_iowtime`). System buffer I/O accumulates here. This number may vary due to system load.
`ac_himem`	Memory-use high water mark in words.
`ac_sctime`	System call time in clocks.

10.1.11.2 End-of-job accounting record

There is one end-of-job record per job. The record is written when the last process of a job is terminated. The record contains the following fields:

Field	Description
`ace_jobid`	Job ID of the job to which this record belongs.
`ace_uid`	User ID from the job table.
`ace_himem`	High-water memory use of job; sum of all processes in a job at any given time (in clicks). This can vary because of scheduling differences.
`ace_sdshiwat`	Secondary data segment high-water use; sum of all processes in a job at any given time (in SDS units). This can vary because of scheduling differences.
`ace_nice`	Last nice value of the job.
`ace_fsblkused`	Sum of the file system storage used. This value may be negative if more space was freed up than was consumed.
`ace_etime`	End time of the job (in seconds).

10.1.11.3 Multitasking accounting record

If a process is multitasked, a multitasking accounting record is written when the last member of the multitasked group is terminated. The record contains the following fields:

Field	Description
`ac_smwtime`	(Not on CRAY C90 systems) Semaphore wait time (in clocks).
`ac_mutime[MUSIZE]`	Time connected to (*i* + `1`) CPUs (in compressed 1/100ths of a second format). Prior to UNICOS release 8.3, the multitasking CPU times were stored as 21-bit pseudo-floating point numbers. Beginning with release 8.3, these values are in 1/100ths of a second and are compressed as 16-bit pseudo-floating point numbers. The compression and unit changes were made so that multitasking information for a maximum of 32 CPUs can be stored in the `pacct` file without changing the size of the records.

10.1.11.4 SDS accounting record

(Not on CRAY EL systems) If a process utilizes SDS, an SDS accounting record is written. The record contains the following fields:

Field	Description
`acs_mem`	Memory integral based on residency time, not execution time (in click-ticks).
`acs_lio`	Logical I/O request count; this count is the number of `ssread` and `sswrite` system calls made.
`acs_ioch`	Number of characters transferred to and from the SDS, stored in bytes.

10.1.11.5 MPP accounting record

If a process uses a Cray MPP system, an MPP accounting record is written that contains the following fields:

Field	Description
`ac_mpppe`	Number of MPP processor elements used.
`ac_mppbe`	Number of MPP barrier bits used.

`ac_mpptime`	Number of clocks that the MPP has been used.

10.1.11.6 Performance accounting record

When the optional performance accounting feature is enabled (by using the `devacct`(8) command with the `-b` option), a performance accounting record is written at the end of each process. Each record contains the following fields:

Field	Description
`acp_rtime`	The process start time offset (in clocks) from the previous second (reported in the `ac_btime` field of the base accounting record). This field allows you to trace start times of processes that are spawned in the same second.
`acp_tiowtime`	The terminal I/O wait time (in clocks); in other words, the period of time starting when a process performing I/O to a tty or pseudo-tty is removed from the run queue and ending when the process is reconnected to a CPU. This number may vary due to system load.
`acp_srunwtime`	This field is currently disabled.
`acp_swapclocks`	The time (in clocks) that a process spends on the swap device.
`acp_rwblks`	The number of physical blocks transferred by the process using the system buffer I/O interface. This number varies due to conditions in the system buffer cache.
`acp_phrwblks`	The number of physical blocks transferred by the process using the raw I/O interface.

10.1.12 Multitasking incentives

Some sites may want to provide accounting incentives for the use of multitasking. Several of these are available through the appropriate setting of installation parameters.

10.1.12.1 Memory integrals

Three different memory integrals are available to the accounting software. The differences among them are important to those sites that want to give incentives for use of multitasking.

Memory integral #1 - At each change in memory size, memory integral #1 is incremented by the total CPU time used since the last memory change, times the memory size, as follows:

*MI #1: memory size * (total CPU time since last size change)*

Thus, a program that is connected to two CPUs for some period will pay twice the memory cost for that period. When using memory integral #1, a multitasking program incurs the same charges, no matter how many CPUs it gets. This is helpful if consistent billing is important to your site, but not as helpful if incentives for multitasking are important.

Memory integral #2 - The calculations for memory integral #2 are similar to those for #1, except that the increment is the sum of times when at least one CPU was connected, times the memory size, as follows:

*MI #2: memory size * (total time when program was connected to at least one CPU since last size change)*

A multitasking program pays (in memory charges) only for the first CPU it receives; additional CPUs do not increase the memory charge. Using memory integral #2, a multitasking program can potentially decrease its memory charge by a factor equal to the number of CPUs in the machine. This is an incentive for using multitasking. However, because the amount of time a program is connected to a number of CPUs varies from run to run, memory integral #2 is not consistent. The maximum value for #2 is equal to #1 (if no connect time overlap occurs). Note that this also means that #1 is equal to #2 for single-tasked programs.

Memory integral #3 - Some sites with multi-CPU machines may wish to allow an individual program to use a proportionally large amount of memory only if it is also able to use more than one CPU. For instance, in a four-CPU machine, allowing one program to use 90% of memory may idle some CPUs if the program uses only one CPU.

Memory integral #3 allows the site to control this aspect of CPU use by adding an extra factor into the calculation for memory integral #2. The total memory available to user programs is divided by the number of CPUs to derive the value of "one CPU worth of memory." The memory size of the program is then

divided by the "CPU worth" factor to get the extra factor in memory integral #3, as follows (this extra factor cannot be less than 1):

*MI #3: memory size * (total time when program was connected to at least one CPU since last size change) * (memory size / "one CPU worth of memory")*

Memory integral #3 provides an incentive for single-tasked programs to limit themselves to one CPU worth of memory. Multitasked programs will also pay more in memory charges for lots of memory, but they can reduce their memory charges by using multiple CPUs. However, memory integral #3 is as inconsistent as #2, and it can also affect the memory charges for single-tasked programs.

Note that the changes from #1 to #2 and #2 to #3 are, in a sense, opposite for multitasking programs. The changes from #1 to #2 reward multitasking programs by a factor of up to the number of CPUs. The changes from #2 to #3 penalize large-memory programs by up to the number of CPUs. Thus, if a multitasking program has used all memory (on a four-CPU machine), memory integrals #1 and #3 would be nearly equal, and the value of #2 would be approximately one-quarter the value of #1 or #3.

The accounting software is released with memory integral #2 as the default. The `MEMINT` variable in `/etc/config/acct_config` can be changed to match the site's needs.

10.1.12.2 Reducing charges

Another incentive you can provide for the use of multitasking is to reduce the charges for CPU time for multitasking programs. This can be accomplished with weighting factors. The operating system kernel maintains counters of the length of time spent by a user program with one processor connected, two processors connected, and so on.

By default, the charges for a multitasking program would be calculated as follows:

```
sum = 0;
for (i=0; i < ncpu; i++)
                sum += ac_mutime[i] * (i+1);
```

This calculation assumes that all CPU time is charged equally. With the weighting factors, the site can specify, for instance, that a second CPU should be only 75% as expensive as the first CPU. Therefore, a program that gets two CPUs as it executes would have its CPU time charges reduced. Note that, because this charge depends on how much overlap a program gets, charges

may vary from execution to execution. However, charges are never more than the full price for all CPUs.

The accounting software is released with all CPUs having a cost of 1. The `MUTIME_WEIGHT` *x* variables, defined in `/etc/config/acct_config`, can be changed to meet the site's needs.

Note that the user time reported by the `time`(1) command is adjusted so that there is no charge for wait-semaphore time. (This is in order to provide consistent CPU time charges.) The multitasking overlap times do not adjust for wait-semaphore times and, hence, may actually calculate to a greater CPU time than the sum of the user times. In this case, the CPU charge is limited to the sum of the user times.

10.1.13 Socket accounting

Socket accounting tracks network usage from the perspective of sockets, wherein one process may use several sockets, and several processes may use the same socket.

The recorded accounting information tracks all of a socket's usage, but it can only be linked to the process that most recently closed the socket. This information can help you resolve network problems and/or monitor system network usage.

You can use the `csasocket`(8) command to summarize and process the socket data; `csaswitch`(8) can be used to check the status of, enable, and disable accounting methods, including socket accounting. See the `csasocket`(8) and `csaswitch`(8) man pages for more information.

10.1.14 Device accounting

This section describes device accounting. On large computer systems with expensive peripheral devices, it may be useful to associate device usage with the user who initiated the I/O. Cray device accounting allows a system administrator to specify the accounting data that should be collected for device use. This system allows a site to individually label each mounted disk's partitions and so enables the site to charge each type of secondary storage at a different rate. For example, the amount of I/O on a high-speed storage device such as an SSD may be charged at a different rate than I/O on a disk device.

10.1.14.1 Categories of devices

The following three categories of devices under the UNICOS operating system are important in device accounting:

- Character special devices, which are devices such as terminals, pseudo tty devices, and the HSX channel.
- Block devices, which are devices such as disks, BMR, and the SSD.
- Logical devices, for device accounting, which are the individual file systems. Such devices do not always correspond to a single device, but are treated as such by device accounting.

The device accounting system accounts for device I/O by device type. For a character device, device type is equivalent to its major number. For example, tty devices are major number 1 (in the default system), so they are accounted for as character device 1 (ios-tty). No accounting is performed for block devices, because block devices are used to create file systems; instead, they are treated as logical devices. Logical devices consist of one or more partitions of disk, SSD, and BMR storage. Each logical device is formatted by the `mkfs`(8) command, which provides it with a superblock. The `devacct`(8) program allows you to write an accounting device type into the superblock of each logical device.

10.1.14.2 Structures and device names

The `BLOCK_DEVICE` *x* and `CHAR_DEVICE` *x* parameters in `/etc/config/acct_config` contain the SBU values and names for device accounting. Refer to Section 10.1.10.1.9, page 237, for an explanation of configuring these parameters.

Device accounting uses arbitrary ASCII names for the user interface to accounting; internally, these names are mapped by the accounting library routines `typetonam` and `namtotype`. To be useful, these names should be meaningful to even the beginning user, because the `ja`(1) (job accounting) command displays these names when invoked with the `-d` option. The ASCII names are defined in the device name field of the `BLOCK_DEVICE` *x* and `CHAR_DEVICE` *x* parameters.

Logical device accounting names are displayed to the user by `ja` and the accounting programs, and are used by `devacct`(8) to determine the numeric values the kernel uses.

Logical and character device names should not match; in fact no two names should match, because the user cannot distinguish between them.

If names contain spaces (the shell field separator (`SHELL IFS`)), double quotes must be used around the device type names during command invocation.

Device names are used as output by `ja` and the accounting programs; therefore, keeping the names fairly short (less than 40 characters) will make them more readable.

System billing units (SBUs) have the following meanings:

SBU	Description
`Logical I/O Sbu`	The total number of system calls made to this type of device is multiplied by `Logical I/O Sbu` to determine the SBU cost. This value should be nonnegative.
`Characters Xfer Sbu`	The total number of characters transferred to this device type is multiplied by `Characters Xfer Sbu` to determine the SBU cost. This value should be nonnegative.

10.1.14.3 Configuration changes

The system is released with the character devices configured to match the released configuration; any changes to `/usr/src/uts/c1/cf/devsw.c` should be reflected in the configuration file.

The block device configuration is released with a simple configuration. Several extensions are possible, although some may require altering the values of `MAXBDEVNO` and `MAXCDEVNO`, and rebuilding the system and accounting commands. First, if a site has a special temporary device that is restricted to a set of users, a special type might be placed on that device to allow the billing process to increase the cost of use, offsetting the lower rate of use. Second, SSD or BMR allocated to logical device cache may be reflected in the configuration.

10.1.14.4 System header files

The system header files discussed in this section are important in device accounting.

10.1.14.4.1 param.h header file

The values MAXBDEVNO and MAXCDEVNO are contained in the /usr/include/sys/param.h file; they set the maximum size of the accounting structures in the user structure and the maximum size of the accounting data written. It is recommended that they not be increased beyond the current values unless necessary (although making MAXCDEVNO smaller and MAXBDEVNO larger by the same amounts is acceptable).

MAXBDEVNO is the maximum number of block (logical) device accounting types. This number can be changed from the current value of 10.

MAXCDEVNO is the maximum number of character device accounting types. This number can be changed from the current value of 35.

10.1.14.4.2 acct.h header file

The /usr/include/sys/acct.h header file contains all the kernel structures for accounting and sets the following values related to device accounting:

Value	Description
NODEVACCT	The number of devio entries per accounting record. This value is the number of device accounting entries that fit into one accounting record.
ACCT_CHSP	A marker combined by an OR operation into the type field (acd_type) to indicate that the devio entry is for a character device.
_MAXDEVIOREC	The maximum number of device accounting records that can be written for any individual process.

10.1.14.5 Using device accounting (devacct(8))

Use the devacct(8) command to label file systems with accounting types while they are mounted. If a file system does not contain a device type label, device accounting ignores it.

In order to enable device accounting, the system may be built to automatically enable specific device types. However, an easier solution is to insert lines into the system startup procedure (/etc/config/daemons) to enable device accounting when bringing the system to multiuser mode. The following

example shows a line that can be added to the daemons file (`/etc/config/daemons`) to enable device accounting (remember the device type name is a single argument and so it may need to be enclosed in double quotation marks if it contains shell separators):

```
SYS1 devacct YES - /usr/lib/acct/devacct -b "device type name"
```

The `devacct` command with the `-l` option may be used to label file systems (file systems may be labeled only while mounted). The names of device types are defined in the `BLOCK_DEVICE`*x* and `CHAR_DEVICE`*x* variables located in `/etc/config/acct_config`. Some of the default names include spaces; such names must be enclosed in double quotation marks on the command line.

For example, to label the device `/dev/dsk/root` with the label `"dd49 with ldcache"`, the command would be as follows:

```
/usr/lib/acct/devacct -l "dd49 with ldcache" /dev/dsk/root
```

Device accounting for any device type may be turned on at any time by invoking the `devacct` command with the `-b` option. While device accounting is on, no records are written unless per-process accounting is enabled.

For example, to enable accounting for the devices labeled `"dd49 with ldcache"`, the command is as follows:

```
/usr/lib/acct/devacct -b "dd49 with ldcache"
```

You can turn on performance accounting using the following command:

```
/usr/lib/acct/devacct -b perf01
```

Device accounting for any device type may be turned off at any time by invoking the `devacct` command with the `-t` option. While accounting is disabled, those processes that have already accumulated data will report that data at termination if per-process accounting is enabled. For example, to disable accounting for the devices labeled `"dd49 with ldcache"`, the command is as follows:

```
/usr/lib/acct/devacct -t "dd49 with ldcache"
```

To determine the current label for a device, use the `devacct` command with the `-L` option. For example, to list the current label of `/dev/dsk/root`, you would execute the following command:

```
/usr/lib/acct/devacct -L /dev/dsk/root
```

10.1.14.5.1 Implications of device accounting

The system overhead for device accounting is fairly low. However, the amount of accounting data produced in the worst cases is more than double that produced by standard accounting. The more device accounting data kept, the more file system space that is required. If one device is accounted for, processes that use that device generate twice as much accounting data as a process that did not use the device or the same process without device accounting. However, for 1 to `NODEVACCT` device types, the maximum size of the accounting data does not increase, except that more processes may use one of the devices.

10.1.14.5.2 Tape device accounting

To enable or disable tape device accounting, use the *device type name* associated with the `CHAR_DEVICE15` parameter in `/etc/config/acct_config`. By default, this device name is `"bmx daemon"`.

The device name associated with `CHAR_DEVICE11` (the default is `"bmx tape"`) controls device accounting only for tape diagnostics.

To enable device accounting for the tapes, execute the following command:

```
/usr/lib/acct/devacct -b "bmx daemon"
```

10.1.15 Switching / and /usr file systems

Occasionally, sites run on numerous `/` and `/usr` file systems and want to maintain the same accounting files throughout. The easiest way to accomplish this is to put `/usr/adm` or `/usr/adm/acct` on a separate file system and mount this file system along with each different system.

In addition, two other files, `/etc/csainfo` and `/etc/wtmp`, must be copied from the previously booted `/`. These files must be copied to the new root file system before it is brought up. Failure to correctly copy `/etc/csainfo` to the new `/` can cause `csarun` to abort abnormally. Incorrect connect time data is reported when `/etc/wtmp` is not copied.

10.1.16 Logging information

The following sections describe log files found in the UNICOS operating system.

10.1.16.1 Boot log

The boot log contains the date and time the system was booted. It is located in `/etc/boot.log` and can be accessed through normal file manipulations such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). The `/etc/rc` (see `brc`(8)) script appends the record to the `boot.log`. The format is as follows:

date, `uname -a`
yy/mm/dd hh:*mm system node release version hardware*

Example:

```
93/05/10 08:07 sn1703c cool 8.0.0tk dev.6 CRAY Y-MP
```

See `date`(1) and `uname`(1) for further information. See also `who`(1), and the sample `wtmp` and `utmp` files in this chapter.

10.1.16.2 `cron` log

The `cron` log contains the history of all actions taken by the `cron`(8) command. It is located in `/usr/lib/cron/log` and can be accessed by using normal file manipulations such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). The format of this file is as follows:

```
CMD: command_executed username  process_id  job_type
     start_time username  process_id    job_type
     end_timerc=error return code
```

job_type can have one of the following values:

`a`	`at` job
`b`	Batch job
`c`	`cron` job

Example:

```
>  CMD: 645827040.a
>  user1 7191 a Tue Jun 19 15:24:00 1990
>  CMD: /usr/lib/sa/sa1 120 1
>  root 7192 c Tue Jun 19 15:24:00 1990
<  root 7192 c Tue Jun 19 15:24:00 1990
<  user1 7191 a Tue Jun 19 15:24:00 1990
>  CMD: 645827059.b
>  user1 7273 b Tue Jun 19 15:24:19 1990
<  user1 7273 b Tue Jun 19 15:24:20 1990 rc=1
```

10.1.16.3 Dump log

The dump log contains the time and a reason for each dump. The system supplies the time and the user supplies the reason. By default, the dump is located in `/etc/dump.log` and can be accessed using the normal file manipulations such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). When the system is changing out of single-user mode, `brc`(8) calls `coredd`(8) to copy a dump file to a file system. The location of the dump can be reconfigured by using the install tool. Note that the user may also change the location of the log file by using the `-l` option with the `cpdmp` command.

Example of `/etc/dump.log`:

```
cpdmp: 035120 blocks on dump device - waiting to be copied
04/26/93 07:27:09   coredd: Copying system dump into /core//04260727.
Unicos-E dump copied to=/core//04260727/dump
        dump taken: 04/26/93 at 07:18:51
        reason: PANIC: master.s: EEX interrupt in UNICOS kernel
```

10.1.16.4 New user log

The new user log contains information on new users given logins on the system; this data includes who added the users, the times at which they were added, and information about their environment defaults and IDs. This log is located in `/usr/adm/nu.log` and can be accessed using normal file manipulations such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). It is created by the `nu`(8) command. An example of the format follows:

```
user1:co:user login #1
user1:ui:10702:di:/j/user1:sh:/bin/csh:dr:/:pw:qQfHS6B8XYdzg
user1:gi:128,129,130,131,132
user1:ai:0
user1:dl:0:mx:0:mn:0:lk:0:tp:0
user1:dc:default:cm:default:pm:default
        added by adm1 on Wed Jul 20 08:43:20 1988
```

10.1.16.5 su log

The `su` log records `su`(1) attempts for the current day. It is located in the `/usr/adm/sulog` file and can be accessed using normal file manipulations such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). It is written by the `su`(1) command. The format of the log is as follows:

`SU` *MM/DD hh:mm flag tty olduser-newuser*

flag can have the following values:

`+` `su` was successful.

`-` `su` was not successful.

olduser is the login name of the user executing `su`, and *newuser* is the name of the user the executing user is becoming. For example:

```
SU 06/19 15:13 + console operator-root SU 06/19 15:13 + ttyp025 \n
user1-root SU 06/19 15:14 + ttyp021 user2-adm SU 06/19 15:19 - ttyp026 \n
user3-root SU 06/19 15:19 - ttyp022 user4-root
```

10.1.16.6 OLDsu log

The `OLDsu` log is a directory containing all files of daily `su`(1) attempts. It is located in `/usr/adm/OLDsu/*` and can be accessed using normal file manipulations such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). The `/etc/rc` script moved the `/usr/adm/sulog` file to the `/usr/adm/OLDsu` directory at system startup. An example of the format follows:

```
$ ls -al OLDsu

-rw-rw-rw-  1 root    2864 Oct 31 19:02 Oct31
-rw-rw-rw-  1 root   20211 Sep 12 09:15 Sep01
-rw-rw-rw-  1 root     938 Sep 12 09:15 Sep02

$ cat Sep01

SU 09/01 16:29 + tty?? root-root
SU 09/01 16:30 + tty?? root-sys
SU 09/01 16:32 + tty?? root-sys
SU 09/01 16:32 + tty?? root-root
SU 09/01 16:34 + tty?? root-sys
SU 09/01 16:35 + tty?? root-root
SU 09/01 16:36 + tty?? root-sys
```

10.1.16.7 System logs

The system logs are files into which the `syslogd`(8) command has logged messages. They are located in the `/usr/adm/syslog/*` directory. Note that these files are described by the configuration file `/etc/syslog.conf`. They can be accessed using normal file manipulations such as `tail`(1), `cat`(1),

page(1), and more(1). They are written by the /etc/syslogd command; the logger(1B) command also makes entries in the system logs.

These logs consist of ASCII messages, which may include debug messages, kernel messages, and so on.

The following example is the configuration file for /etc/syslogd (these fields are described on the syslogd(8) and syslog(3) man pages):

```
$ cat /etc/syslog.conf

# USMID @(#)man/2302/02.accounting      92.2    02/05/96 13:26:44
#
#       This is a configuration file for /etc/syslogd
#
#*.debug                                /usr/adm/syslog/debug
#
mail.debug                              /usr/spool/mqueue/syslog
#
kern.debug                              /usr/adm/syslog/kern
#
daemon,auth.debug                       /usr/adm/syslog/auth
#
#*.err;kern.debug;auth.notice           /dev/console
#
*.err;kern.debug;daemon,auth.notice;    /usr/adm/syslog/daylog
#
#*.alert;kern.err;daemon.err            operator
*.alert                                 root
```

Note: The /etc/syslogd.conf file does not work if spaces are in it; only tabs can be used to separate items in this file.

The following example shows a listing of the files in the /usr/adm/syslog directory:

```
$ ls -l /usr/adm/syslog
total 10
-rw-r--r--   1 root     root         168 Jun 19 15:35 auth
-rw-r--r--   1 root     root        5164 Jun 19 15:45 daylog
-rw-r--r--   1 root     root        4107 Jun 19 15:45 kern
drwxr-xr-x   2 root     root       16864 Jun 19 15:09 oldlogs
```

10.1.16.8 Error log

The error log is a file containing error records from the operating system. The default error file is `/usr/adm/errfile`. There are two facilities available for generating reports from the data collected by the error-logging mechanism. The first is `errpt`(8), which processes error reports from the data, and the second is `olhpa`, a hardware performance analyzer that reports the hardware errors and statuses recorded in the system error log.

Note: The `olhpa` facility is only available on IOS-E based systems. It is not available on GigaRing based systems.

The `/etc/errdemon` command (see `errdemon`(8)) reads `/dev/error` and places the error records from the operating system into either the specified file, or `errfile`, by default. The `/etc/rc` (see `brc`(8)) script starts `/etc/errdemon`, and `/etc/mverr` is used to start a new `errfile`.

The following example shows sample `errpt` output:

```
Tue Jun 7 12:01:49 1988
       Error reported from IOS 0 for device S49-0-21
       Major:0  Minor:6         Block:140868   status: Recovered
       Iop:0  Channel:21        Unit:0
       Cylinder:1156  Head:5    Sector:0
       Function:Read  Requested:344064 bytes   Received:344064 bytes

       IOS 0 ERROR LOGGING ENABLED
```

See `errpt`(8) for further information. See the *Online Maintenance Tools Guide for Cray PVP Systems*, Cray Research publication SD–1012, or the `olhpa`(8) man page for information concerning `olhpa`. [2]

10.1.16.9 Multilevel security (MLS) log

The multilevel security (MLS) log is a file containing security-relevant event loggings. The security log, `/usr/adm/sl/slogfile`, can be analyzed by using the `reduce` command. `reduce` extracts, formats, and outputs entries from UNICOS security event files. The security event logging daemon, `slogdemon`(8), collects security-relevant records from the operating system by reading the character special pseudo device `/dev/slog`. For more information regarding the format of the security log and on the UNICOS MLS feature, see

[2] CRAY RESEARCH PRIVATE. This document contains information private to Cray Research, Inc. It can be distributed to non-CRI personnel only with approval of the appropriate Cray manager.

the reduce(8) man page and *General UNICOS System Administration*, Cray Research publication SG–2301.

10.1.16.10 System activity log

The system activity report facility provides commands for generating various system activity reports. Two reporting capabilities exist (one automatic and one user-driven); however, the actual reports are created by the sar(8) program in either case. The system activity log is located in /usr/adm/sa/sa*DD* and can be accessed with sar.

Warning: The log files located in /usr/adm/sa/sa*DD* on a Cray ML-Safe configuration of the UNICOS system are considered to be covert channels. You may want to consider restricting access to these files to the adm group.

With this command, you can generate system activity reports in real time and save system activities in a file for later use. The sa1, sa2, and sadc commands (see sar(8)) generate system activity data on a routine basis, with sa2 calling sar to generate the report.

UNICOS counters are incremented as various system actions occur. These counters provide system-wide measurements. sadc accesses /dev/kmem to read these system activity counters.

Refer to the sar(8) man page for more information on the format of the system activity log.

10.1.16.11 Message log

The message log contains messages and replies to the operator. It is located in /usr/spool/msg/msglog.log and can be accessed using normal file manipulations, such as tail(1), cat(1), pg(1), and more(1). All messages and replies to and from the operator console are put into this file by the console. An example of the file format follows:

```
Apr  1 07:11:06  Message daemon stopped
Apr  1 09:36:54  Message daemon started
Apr  1 08:09:49  Message   1: TM122 - mount tape WK1102(sl) on a CART
 device for user1 50,  () or reply cancel / device name
```

Warning: The msglog.log file is considered a covert channel on a Cray ML-Safe configuration of the UNICOS system. You may want to consider restricting access to this file to the adm group.

10.1.16.12 Accounting logs

The accounting logs are files containing various accounting information, as follows:

Log	Description
`csainfo`	A file containing boot times. It can be accessed with the `od`(1) command (the `-d` option will give the seconds). Each time the system is booted, the boot time is written to `/etc/csainfo` by the `/etc/csaboots` (see `csaboots`(8)) command. `csaboots` is invoked by `/etc/rc` (see `brc`(8)). See also the description of the boot log in Section 10.1.16.1, page 260.
`utmp`	A file containing active system and terminal connection information. This log is used by `write`(1), `who`(1), `wall`(8), and `mail`(1) in getting user information. It is located in `/etc/utmp` and can be accessed using the `who`(1) and `last`(1B) commands. It is written to by `init`(8), `date`(1), `login`(1), and `getty`(8). For information on the format of `utmp`, see `utmp`(5).
	 **Warning:** On a Cray ML-Safe configuration of the UNICOS system, `utmp` and `wtmp` are considered to be covert channels. You may want to consider restricting access to these files to the `adm` group.
`wtmp`	A file containing a system and terminal connection history record. This log includes usage statistics for each terminal, date change, time stamp, boot records, reboots, shutdowns, and state changes. `wtmp` must exist; programs that access it do not create it (the `/etc/rc` script creates `/etc/wtmp` by default). Records are in the form of `utmp`(5); `acctcon`(8) and `csaline`(8) convert `/etc/wtmp` into session and charging records. This data is merged into the system accounting reports. `wtmp` can also be accessed using the `who`(1) and `last`(1) commands. `wtmp` is written by `init`(8), `date`(1), `login`(1), and `getty`(8). For information on the format of `wtmp`, see `utmp`(5).
`pacct`	Files containing per-process accounting data; these are located in `/usr/adm/acct/day/pacct*` and can be accessed using the `acctcom`(1) command. Note that these files are read by system accounting programs, and the information appears in the accounting reports. `pacct` is written by the kernel, and its format is described in `/usr/include/sys/acct.h`.

Warning: On systems running a Cray ML-Safe configuration of the UNICOS system, access to `pacct*` files should be restricted to the `adm` group.

The following data files are accessed by system accounting programs, and their information appears in the accounting reports:

Log	Description
`disktacct`	A file containing disk accounting data, located in `/usr/adm/acct/nite/disktacct`. The `/usr/lib/acct/dodisk` (see `dodisk`(8)) command writes this file.
`fee`	A file containing user fees for accounting data, located in `/usr/adm/acct/day/fee`. This file is written by `/usr/lib/acct/chargefee` (see `chargefee`(8)).
`nqacct`	A file containing NQS daemon accounting data, located in `/usr/adm/acct/day/nqacct*`. This file is written by `/usr/lib/nqs/nqsdaemon`. See `/usr/include/acct/dacct.h` for the format.
`soacct`	A file containing socket accounting data, located in `/usr/adm/acct/day/soacct*`. This file is written by the kernel. See `/usr/include/sys/acct.h` for the format.
`tpacct`	A file containing tape daemon accounting data, located in `/usr/adm/acct/day/tpacct*`. This file is written by `/usr/lib/tp/tpdaemon` (see `tpdaemon`(8)). See `/usr/include/acct/dacct.h` for the format.

10.1.16.13 NQS log

The NQS log contains NQS information. Its default location is the ASCII file `/usr/spool/nqs/log` (you can change the location of the log file with the `qmgr set log_file` command; to see where the current log file resides, use the `qmgr show parameters` command). Access to `/usr/spool/nqs` is restricted; however, if you have the correct permissions, you can access the NQS log file using normal file manipulations, such as `tail`(1), `cat`(1), `pg`(1), and `more`(1). This log is created by the NQS log daemon.

Warning: On systems running a Cray ML-Safe configuration of the UNICOS system, access to the NQS log should be restricted to the `adm` group.

An example of the log file's format is as follows:

```
05/13 08:00:00 I getpkt(): Received packet from local process: <89775>.
05/13 08:00:00 I getpkt(): Client process real UID=<900>.
05/13 08:00:00 I getpkt(): Packet type=<PKT_QUEREQVLPQ(30)>.
05/13 08:00:00 I getpkt(): Packet contents are as follows:
05/13 08:00:00 I getpkt(): Pkt_str[1] = <batnam1>.
05/13 08:00:00 I getpkt(): Pkt_int[1] = <40>.
05/13 08:00:00 I getpkt(): Pkt_int[2] = <119>.
05/13 08:00:00 T nqs_quereq(): Request <40.cool>: Attempting to read request.
05/13 08:00:00 T nqs_quereq(): Request <40.cool>: Request was read.
```

Adding Your Cray Research System to Your Network [11]

This section describes how to place your CRAY J90se system on an existing TCP/IP network. This section also includes a table that describes some of the most common TCP/IP configuration files.

After you have placed your CRAY J90se system on an existing TCP/IP network by using the information in this section, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, for additional networking configuration information; this publication also includes which menus to use for various networking tasks.

11.1 Related network information

The following publications contain additional information that will be of use to you:

- *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303
- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022 (man pages):

`arp`(8)	`inetd`(8)	`rlogind`(8)
`atmarp`(8)	`initif`(8)	`route`(8)
`enstat`(8)	`mkbinhost`(8)	`rshd`(8)
`fingerd`(8)	`netstart`(8)	`sdaemon`(8)
`ftpd`(8)	`ntalkd`(8)	`tcpstart`(8)
`gated`(8)	`ping`(8)	`traceroute`(8)
`ifconfig`(8)	`rexecd`(8)	

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011 (man pages):

`ftp`(1B)

`hostname`(1)

`netstat`(1B)

`telnet`(1B)

- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014 (man pages):

  `gated-config`(5)

  `hosts`(5)

  `hosts.equiv`(5)

  `inetd.conf`(5)

  `lo`(4)

  `networks`(5)

  `protocols`(5)

  `rhosts`(5)

  `services`(5)

  `shells`(5)

- The following publications also are recommended, but Cray Research does not provide them:
  - *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture*, Douglas Comer. Prentice Hall, 1991.
  - *UNIX Networking*, S. Kochan and P. Wood. Hayden Books, 1989.
  - *Introduction to the Internet Protocols*, Charles Hedrick, Rutgers University, 1987. Anonymous `ftp` from `cs.rutgers.edu`.
  - *Introduction to the Administration of an Internet-based Local Network*, Charles Hedrick, Rutgers University, 1988. Anonymous `ftp` from `cs.rutgers.edu`.

Procedure 31: Adding a CRAY J90se system to an existing TCP/IP network

To place a CRAY J90se system on an existing TCP/IP network, you should complete the following steps:

Note: To add your CRAY J90se system to an existing TCP/IP network, you must have several configuration files. The easiest way to create these files is to configure the CRAY J90se system to talk to another host on the network, copy the necessary files from that machine to your CRAY J90se system, and then change them. The first five steps of this procedure make the changes to allow you to talk to another host. You then copy files you need and change them for your CRAY J90se system. This procedure assumes you are either an administrator of your network or that you will have a network administrator as a resource when you add your CRAY J90se system to your existing TCP/IP network.

1. Verify that the network section of the `/opt/CYRIos/`*sn9xxx*`/param` file on the SWS contains the proper configuration for your system. If you are adding network interfaces, you must reboot the system.

2. Create a minimal `/etc/hosts` file. (Do not overwrite the existing `/etc/hosts` file.)

   The `/etc/hosts` file contains the database of all locally known hosts on the TCP/IP network. Create an `/etc/hosts` file that contains a local host entry, entries for the interfaces on the CRAY J90se system, and an entry for at least one other host on the same network as the CRAY J90se system. The entry format is as follows:

   IPaddress host_name annotations

   Example:

```
# cat /etc/hosts

127.0.0.1      localhost loghost
(local host)


456.789.16.8    cray cray-eth
(your CRAY J90se system)


456.789.16.125  cyclone cyclone-eth1
(other host)
```

3. Compile a binary hosts file.

Cray Research systems support a binary `/etc/hosts` file called `/etc/hosts.bin`. Create this file by using the `/etc/mkbinhost` command, as follows:

```
# /etc/mkbinhost
/etc/hosts.bin: 3 entries written
```

4. Update the `/etc/config/interfaces` file.

   The `/etc/config/interfaces` file defines all network interfaces on the Cray Research system. Change the host name for each interface on your system to match those you chose in step 2; for additional information, see the `initif`(8) man page and the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304. The entry format is as follows:

   interface_name family address ifconfig parameters

   Example:

```
# cat /etc/config/interfaces

...some comment lines omitted...

lo0   -     inet   localhost    -
gether0   -     inet   cray-eth     - netmask 0xffffff00
gfddi0 -    inet    cray-fddi    - netmask 0xffffff00
gatm0  -    inet    cray-atm     - netmask 0xffffff00 iftype pvc
ghippi0   /etc/ghipp0.arp inet cray-hippi - netmask 0xffffff00 hwloop
```

5. Activate the changes by executing the following`/etc/initif` script. ATM and HIPPI require additional configuration files, so you may want to comment out the lines that pertain to those network interfaces until their configuration files are complete.

```
# /etc/initif
Configuring all network interfaces: lo0 gether0 gfddi0 gatm0 ghippi0
```

6. Create a default route.

This step creates a default route to let you communicate with hosts that are on different networks than the Cray Research system. To reach hosts that are not on the same FDDI or Ethernet network as the CRAY J90se system, you must have a default route. To create a route, execute the `/etc/route`(8) command, as shown in the following example:

```
# /etc/route add default otherhost
add net default: gateway otherhost
```

The *otherhost* is the IP address of a host that is on the same network as the CRAY J90se system and connects to one or more additional networks.

This command can be placed in the `/etc/tcpstart.mid` script so that it will be run automatically at system startup.

Alternatively, the default route can be included in the `/etc/gated.conf` file. For more information on the use of the `/etc/gated.conf` file, see the `gated-config`(5), `gated`(8), and `tcpstart`(8) man pages.

7. Test the network.

Test the network connections by using the `ping` command and view the configuration by using the `netstat` command. The `ping` command tests whether you can reach another host on the network. If `ping` succeeds, you can be confident that the hardware and routing works on all hosts and gateways between you and the system to which you are sending `ping`. The `netstat` command has many options. The `-i` option lets you view a table of cumulative statistics for transferred packets, errors, and collisions for each interface that was autoconfigured. The interfaces that are statically configured into a system but are not located at boot time are not shown; the `-r` option lets you view the routing table. The network address (currently Internet-specific) of the interface and the maximum transmission unit (mtu) in bytes also are displayed. You should become familiar with how these displays look on your system so that you will recognize changes and problems immediately.

Examples:

```
# /etc/ping otherhost
PING otherhost : 56 data bytes
64 bytes from 123.123.12.13: icmp_seq=0. time=10. ms
CONTROL-c
```

```
# netstat -i
Name     Mtu   Network           Address         Ipkts       Ierrs    Opkts     Oerrs
gether0*     1496  cray-net          cray            0           0        2         0
gfddi0    4352  crau-fddi-net     cray-fddi       249466      0        57636     0
gfddi1*   4352  none              none            0           0        0         0
lo0      65535 loopback          localhost       264         0        264       0
```

Note: An * in the Name column of the `netstat -i` command output indicates that the interface is not configured up, so your CRAY J90se system cannot access that network.

8. Transfer full configuration files from another system, as shown in the following example.

Save copies of your original files and add the new entries for the CRAY J90se system to the files you transfer (for example, use ftp to transfer the `/etc/hosts` file from another system on your network).

Example:

```
# cd /etc
# cp hosts hosts.sav
# ftp cyclone
Connected to cyclone.cray.com.
220 fred FTP server (Version 5.2 Fri Feb 18 14:09:58 CDT 1994) ready.
Remote system type is UNIX.
Using binary mode to transfer files.
Name (fred:root): sam
331 Password required for sam
Password:    <-----
```

Enter your password

```
230 User sam logged in.
ftp> get /etc/hosts hosts
200 PORT command successful.
150 Opening BINARY mode data connection for /etc/hosts (328758 bytes).
226 Transfer complete.
328758 bytes received in 0.6 seconds (5.3e+02 Kbyte/s)
ftp> quit
221 Goodbye.
# vi /etc/hosts  <-----
```

Enter your password

```
# /etc/mkbinhost
/etc/hosts.bin: 2675 entries written
```

Your CRAY J90se system should now be on the network. You may want to also transfer an `/etc/networks` file to your CRAY J90se system in the same manner.

11.2 TCP/IP path between J90se and SWS

A TCP/IP path between the CRAY J90se mainframe and the SWS is required for system software installs and upgrades. In addition, you may find it necessary to transfer certain files between the CRAY J90se mainframe and the SWS (for example, the `param` UNICOS configuration file).

This TCP/IP path should be configured as part of the system installation procedures. It can consist of a simple link between a J90se Ethernet interface

and an SWS quad-board plug. The path may also be more complex; this TCP/IP traffic could use a J90se FDDI interface and then be routed between the FDDI ring and a customer Ethernet, to which the SWS is connected.

To start TCP/IP from the CRAY J90se mainframe while still in single-user mode, use one of the following procedures. (This will allow files to be transferred between the CRAY J90se mainframe and the SWS.)

Procedure 32: Start TCP/IP from mainframe

- For direct Ethernet connection between the CRAY J90se mainframe and the SWS:

```
# /etc/ifconfig gether0 CRAY_ETHERNET_IP_addr netmask 0xffffff00
# /etc/inetd
```

- When packets are routed from CRAY J90se FDDI interface to SWS Ethernet interface:

```
# /etc/ifconfig gfddi0 CRAY_FDDI_IP_address netmask 0xffffff00
# /etc/inetd
# /etc/route add default fddi_router_IP_address
```

To verify that the TCP/IP has started correctly, run the `ping`(8) command from the SWS, as follows:

`sws$ /etc/ping sn9`*xxx*

11.3 Changing the SWS host name and IP address

Use the following procedure to change the SWS host name and IP address:

Note: No example prompts are shown in this subsection; all commands are executed on the SWS.

Procedure 33: Changing the SWS host name and IP address

1. Ensure you are logged in as the root user.
2. Change the SWS host name in the following files:

   `/etc/hostname.le0`

   `/etc/nodename`

3. Change the SWS host name and IP address in the following file (you must also change your mainframe IP address in this file):

   `/etc/hosts` (a tab should separate the name from the IP address)

4. Change the SWS host name and IP address in the `/etc/hosts` file on the CRAY J90se mainframe.

11.4 Changing the CRAY J90se host name and IP address

Section 11.3, page 276, described how to change the SWS host name and IP address.

Use the following procedure to change the CRAY J90se mainframe host name and IP address:

- On the UNICOS system, edit the `/etc/hosts` file to change the CRAY J90se IP address.
- On the SWS, edit the `/etc/hosts` file to change the CRAY J90se IP address.

11.5 Backing up all changes

Remember, back up any changes you have made to files on the SWS.

11.6 Verifying the UNICOS configuration file

The UNICOS configuration file controls kernel configuration parameters. Because the CRAY J90se system is preconfigured, you do not need to make any changes to the UNICOS configuration file in order to boot a usable system.

The default configuration file is `/opt/CYRIos/`*`sn9xxx`*`/param`.

Note: After your system is running, you may want to make configuration changes, such as adding a file system. For information on changing the configuration file, see *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303.

11.7 Rebooting in multiuser mode

As a final step in initializing the UNICOS system, you must reboot your mainframe to have the changes made in the `param` configuration file take effect. Use the following procedure:

Procedure 34: Rebooting in multiuser mode

1. Ensure that the CRAY J90se mainframe is in single user mode; if it is not, execute the following commands from the console window:

   ```
   # cd /
   # /etc/shutdown
   # sync
   # sync
   # sync
   ```

2. To start the mainframe, execute the following command:

   ```
   sws$ bootsys -c
   ```

   When this command completes, you will see another system prompt.

3. By default, the CRAY J90se system is booted in single-user mode. Execute the following UNICOS `init`(8) command in the console window to go to multiuser mode:

   ```
   # /etc/init 2
   ```

   You will see the type of output that is normally displayed during system boot. When this command completes, you will see the following prompt:

   ```
   Console login:
   ```
   This means that the system has successfully booted in multiuser mode.

11.8 Domain name service (DNS)

If you want to use domain name service (DNS) to perform host name lookup, you should configure your CRAY J90se system as a caching-only server. This should be done for the following reasons:

- A caching-only server is more efficient than a remote server (resolver only) because it maintains a cache of data and, therefore, requires less frequent network access.

- A caching-only server does not have authority over a particular zone, therefore, it does not have to answer queries from other authoritative servers.
- A caching-only server does not have to load configuration files from disk like a primary master server or across the network like a secondary master server, which gives it a faster start-up time.

To configure your Cray Research system as a caching-only server, you may configure both resolver and the local name server. You can perform most of this configuration by using the UNICOS Installation / Configuration Menu System.

Procedure 35: Configuring a caching-only server by using the menu system

Note: If you have not completed the previous procedure, "Adding a CRAY J90se system to an existing TCP/IP network," you should complete steps 3, 4, and 5 of that procedure before configuring a caching-only server.

To configure your CRAY J90se system as a caching-only server by using the UNICOS Installation / Configuration Menu System, complete the following steps:

1. Select `YES` for the `"Use domain name service?"` entry in the UNICOS Installation / Configuration Menu System. The menu system creates the `/etc/hosts.usenamed` file. The existence of this file indicates that UNICOS will use DNS, rather than the `/etc/hosts` file to look up host names. A sample `TCP/IP Host/Address Lookup Configuration` menu screen follows:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
```

```
                TCP/IP Host/Address Lookup Configuration

    Use Domain Name (DN) service ?              YES
S-> DNS lookup (resolver) ==>
    Local DN server (named) ==>
```

2. Configure the resolver, which consists of creating the `/etc/resolv.conf` file, which is created if you place information in the `DNS lookup`

(resolver) menu. When you have a local name server (named process) running, you should have the local host address (127.0.0.1) as the first name server; otherwise your local named will be bypassed, resulting in decreased performance and increased network traffic.

The following is an example of the menu screen:

```
       TCP/IP Domain Name Service Lookup (resolver) Configuration
    Local domain name                          cray.com
    Address for Domain Name server #1:         127.0.0.1
S-> Address for Domain Name server #2:         128.162.19.7
    Address for Domain Name server #3:         128.162.19.13
    Address for Domain Name server #4:
    Address for Domain Name server #5:
    Address for Domain Name server #6:
    Address for Domain Name server #7:
    Address for Domain Name server #8:
    Address for Domain Name server #9:
```

3. Configuring a caching-only local name server consists of creating the named.boot, root.cache, and localhost.rev files. You can do this by using the Local DN server (named) menu. Perform the following steps to create these files:

   a. The named.boot file is read when named starts up. It tells the server what kind of server it is, over which zones it has authority, and where to get its initial data. The following is an example of the menu screens:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
```

```
                TCP/IP Host/Address Lookup Configuration

    Use Domain Name (DN) service ?              YES
    DNS lookup (resolver) ==>
S-> Local DN server (named) ==>
```

```
             TCP/IP Local Domain Name Server Configuration

S-> Directory for name server files              /etc/named.d
    Address of forwarding name server #1         128.162.19.7
    Address of forwarding name server #2         128.162.19.13
    Address of forwarding name server #3         128.162.1.1
    Slave server?                                NO
    Root name server cache file                  root.cache
    Root name server cache ==>
    Primary zones ==>
    Secondary zones ==>
```

b. The local name server also needs configuration information for the zones for which it is the primary server (the zones for which it has authority). On a caching-only server, the only zone for which the local `named` has authority is the `0.0.127.IN-ADDR.ARPA` zone. This information is stored in the `localhost.rev` file; you can configure its name, but not its contents, by using the menu system:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
                    ->TCP/IP Root Nameserver Cache Config
```

```
                TCP/IP Root Nameserver Cache Configuration

      Server name     Server address  Time To Live
      --------------  --------------  ------------
 E->  earth.cray.com  128.162.3.55    1000000
```

```
              TCP/IP Root Nameserver Cache Configuration

S-> Server name                              earth.cray.com
    Server address                           128.162.3.55
    Time to live                             1000000
```

c. The local name server must know the name of the server that is the authoritative name server for the domain. The `root.cache` file is used to "prime the cache" with this information, and you can configure it by using the menu system:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
```

```
                TCP/IP Domain Name Service Primary Zones

     Name                  File           Account  Serial #
     --------------------  -------------  -------  --------
E-> 0.0.127.IN-ADDR.ARPA   localhost.rev
```

```
               TCP/IP Domain Name Service Primary Zones

    Zone name                                  0.0.127.IN-ADDR.ARPA
    File to contain zone information           localhost.rev
S-> Account name of responsible party
    Serial number for zone
```

4. Start the `named` daemon by executing the following command:

```
# /etc/sdaemon -s named
```

Procedure 36: Configuring a caching-only server without using the menu system

Note: If you have not completed the previous procedure, "Adding a CRAY J90se system to an existing TCP/IP network," you should complete steps 3, 4, and 5 of that procedure before configuring a caching-only server.

To configure your CRAY J90se system as a caching-only server without using the UNICOS Installation / Configuration Menu System, complete the following steps:

1. Enable the domain name system by creating the `/etc/hosts.usenamed` file. The existence of this file indicates that UNICOS will use DNS, rather than the `/etc/hosts` file to look up host names.

2. Configure the resolver by creating the `/etc/resolv.conf` file.

   When you have a local name server (`named` process) running, you should have the local host address (127.0.0.1) as the first name server; otherwise your local `named` will be bypassed, resulting in decreased performance and increased network traffic. The following is an example `/etc/resolv.conf` file:

```
## Domain name resolver configuration file
#
domain cray.com
#
nameserver 127.0.0.1
nameserver 128.162.19.7
nameserver 128.162.1.1
```

3. To configure a caching-only local name server, you must complete the following steps:

   a. Create the `/etc/name.boot` file. This file is read when `named` starts up. It tells the server what kind of server it is, over which zones it has authority, and where to get its initial data.

      The following example shows a sample `named.boot` file. The `directory` line tells the server that all file names referenced are relative to the `/etc/named.d` directory. The `forwarders` line tells the server to forward requests that it cannot resolve to the server at 128.162.19.7. The `cache` line tells the server to load the `root.cache` file (in the `/etc/named.d` directory)as its initial cache entries. The `primary` line tells the server that it has primary authority for the `0.0.127.IN-ADDR.ARPA` domain. The `domain` line tells the server that its default domain is `cray.com`.

```
directory       /etc/named.d
forwarders      128.162.19.7
cache           .                           root.cache
primary         0.0.127.IN-ADDR.ARPA    localhost.rev
domain          cray.com
```

b. Create the `localhost.rev` file. The local name server also needs configuration information for the zones for which it is the primary server (the zones for which it has authority). On a caching-only server, the only zone for which the local `named` has authority is the `0.0.127.IN-ADDR.ARPA` zone. This information is stored in the `localhost.rev` file. The following is an example of a `localhost.rev` file:

```
$ORIGIN 127.IN-ADDR.ARPA.
@               IN      SOA     localhost.cray.com. tas.cray.com. (
                                86400
                                3600
                                36000000
                                86400
                                )
                IN      NS      localhost.cray.com.
1.0.0           IN      PTR     localhost.cray.com.
```

c. Create the `root.cache` file. The local name server must know the name of the server that is the authoritative name server for the domain. The `root.cache` file is used to "prime the cache" with this information. The following is an example `root.cache` file:

```
$ORIGIN .

                    1000000 IN      NS      earth.cray.com.

earth.cray.com. 1000000 IN      A       128.162.3.55
```

4. Start the `named` daemon by executing the following command:

```
# /etc/sdaemon -s named
```

11.9 Common TCP/IP configuration files

After you have the Cray Research system on the network, you should do a few additional things to make sure it is a fully functional member of your network, including configuring `inetd`, adding additional routes, and updating other configuration files. If you are using the menu system, you should update these files by using the menu system options. Table 17 describes some of the most

common TCP/IP configuration files. The *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, describes all of these files.

Table 17. TCP/IP configuration files

File (relative to `/etc`)	Description	Change
`config/atm.pvc`	If you are running your atm interfaces in Permanent Virtual Circuit mode, this file maps the Internet addresses of remote hosts on the atm network to VPI/VCI information.	Yes, if you have atm.
`config/daemons`	Lists system and network daemons to start at system boot.	Probably
`gated.conf` or `tcpstart.mid`	Contains routes to be installed at system boot.	Yes
`config/hostname.txt`	Contains text host name for TCP/IP.	Probably not
`hosts`	Maps Internet addresses to host names.	Yes
`hosts.equiv`	Lists trusted hosts for `rlogin`, `rsh`, and so on.	Optional
`ghippix.arp`	Maps HIPPI ifields to Internet addresses.	Yes, if you have HIPPI
`inetd.conf`	Lists network services to be handled by `inetd`.	Probably not
`config/interfaces`	Lists network interfaces and their characteristics.	Yes
`networks`	Maps network names to network Internet addresses.	Optional
`protocols`	Maps protocol names to protocol numbers.	No
`$HOME/.rhosts`	Lists trusted users for `rlogin`, `rsh`, and so on.	Optional
`services`	Maps protocol and port numbers to service names.	Probably not
`shells`	Lists shells allowed for `ftpd`.	Probably not

Configuring NIS [12]

12.1 Related NIS documentation

The following documentation contains information covered in this section:

- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `netstart`(8), `udbgen`(8), `ypbind`(8), `ypinit`(8), `yppasswdd`(8), `yppush`(8), `ypserv`(8), `ypstart`(8), and `ypxfr`(8) man pages
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `domainname`(1), `udbsee`(1), `yppasswd`(1), and `ypwhich`(1) man pages

12.2 What is NIS?

The Network Information Service (NIS) is a network service that allows information such as passwords and group IDs for an entire network to be held in one database. (NIS was formerly known as Yellow Pages.)

Cray Research also supports a new naming service called Network Information Service Plus (NIS+), developed by SunSoft, Inc., a Sun Microsystems company. NIS+ is one of a suite of technologies that make up Open Network Computing Plus (ONC+), a SunSoft product and technology concept. NIS+ is separately licensed (as part of ONC+). The NIS product continues to be provided with the UNICOS release under the UNICOS license. Only the new NIS+ product requires the separate ONC+ license. For more information on NIS+, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

When implemented with the Remote Procedure Call (RPC) and eXternal Data Representation (XDR) library routines, UNICOS NIS has the following features:

- Look-up service: UNICOS NIS maintains a set of databases that can be queried through the use of pointers, or "keys." Programs can request the value associated with a particular key, or all of the keys, in a database.

- Network service: Programs do not have to know the location of data or how it is stored. Instead, they use a network protocol to communicate with a database server that contains the information.
- Distributed service: Databases are fully replicated on several machines, known as "NIS servers." The servers propagate updated databases among themselves, ensuring consistency.

The UNICOS NIS environment includes at least one Cray Research system and one or more other hosts that also run NIS.

NIS databases contain maps; a *map* contains information that is usually found in an ASCII configuration file. Each map contains a set of keys and associated values. For example, the `passwd` map contains user names (the keys) and their associated `/etc/passwd` file entries (the values). The NIS maps are stored in `dbm` format. The `makedbm` command converts an ASCII file into a `dbm` format file that NIS can use. Usually, you do not have to worry about `dbm` format or the `makedbm` command. To generate the maps, you will use the makefile in the `/etc/yp` directory. For further information on the internal map format, see the `dm`(8) and `makedbm`(8) man pages.

Cray Research supports the following maps on systems running the UNICOS operating system:

Map	Description
`group`	Performs the function of the `/etc/group` file: mapping group names to group IDs.
`netgroup`	Defines networkwide groups used for permission checking when doing remote mounts, remote logins, and remote shells.
`passwd`	Performs a few selected functions of the UDB on UNICOS systems; namely, password, home directory, and shell lookup.
`publickey`	Used for secure RPC, the `publickey` map contains public key/private key pairs for users and hosts on the network. For more information about running secure RPC, see the NIS section in the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

An important distinction to make is that a Cray Research system running UNICOS can **serve** other NIS maps for its domain; however, the Cray Research system (as a client) **consults** only the preceding maps. For more information about supported maps, see the NIS section in the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

An *NIS domain* is a specified set of NIS maps. The set of maps for a given domain is stored in a directory named after the domain. To assign hosts to a particular domain, use the `domainname` command.

Servers provide resources; clients use them. There are two types of NIS servers: master servers and slave servers. The *master server* contains the NIS maps. You can change NIS maps only on the master server. A *slave server* contains copies of the NIS maps that it obtains from the master server for its domain.

Clients do not contain their own copies of the NIS maps. Instead, they request information from the servers in their domain.

Note: You should configure your CRAY J90se system as an NIS slave server.

This section contains procedures for the following:

- Using the menu system to configure your CRAY J90se system as an NIS slave server
- Configuring your CRAY J90se system as an NIS slave server without using the menu system
- Configuring user accounts to use NIS

UNICOS NIS differs from the NIS facility used on other systems based on the UNIX system. Administration of an NIS domain that includes a Cray Research system is different from administration of an NIS domain that does not. For example, UNICOS NIS does not support the broadcast feature. If you are unfamiliar with NIS, you should first read the NIS documentation for your other systems. After you have familiarized yourself with the general NIS mechanism, read the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, to familiarize yourself with UNICOS NIS before you configure NIS on your CRAY J90se system.

Procedure 37: Using the menu system to configure your CRAY J90se system as an NIS slave server

Note: This procedure assumes that another host on the network is already configured as an NIS master server and that your CRAY J90se system can communicate with that host.

To configure NIS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group. If you have an ONC+ license, start the `rpcbind` daemon command by executing the `/etc/rpcbstart` or `/etc/rpcbind` command. This daemon provides support for universal addressing. For more information on the `rpcbind` daemon, see the *Remote Procedure Call (RPC) Reference Manual*, Cray Research publication SR–2089, and the `rpcbind`(8) and `rpcbstart`(8) man pages.

Before you configure NIS on your CRAY J90se system, read the cautionary note and other important information on Section 12.2, page 287.

The following steps use the menu system to configure your CRAY J90se system as an NIS slave server:

1. Enable the menu system to configure NIS. To give the menu system permission to change the configuration for NIS, ensure that the `NIS configuration` option is set to `YES` in the `Configure System->Configurator Automation Options` menu. Also, ensure that the `Configure System->Major Software Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

2. Assign your CRAY J90se system to an NIS domain.

   Select the `Configure System->Network Configuration->NIS Configuration` menu. Enter the NIS domain name, and then activate the NIS configuration. A sample menu screen follows:

```
Configure System
    ->Network Configuration
        ->NIS Configuration
```

```
                        NIS Configuration

S-> NIS domain name
    Import the NIS configuration ...
    Activate the NIS configuration ...
```

3. Run the `/etc/yp/ypinit` script with the `-s` option to configure the CRAY J90se system as an NIS slave server. Include the host name of the NIS master for your domain on the command line, as follows:

```
# /etc/yp/ypinit -sNISmasterserver
```

Running `ypinit -s` *NIS_masterserver* causes a copy of the NIS maps to be transferred from the master to the slave server (your Cray Research system running UNICOS) and placed in the `/etc/yp/`*domainname* directory. Running this command also adds the slave server to the `ypservers` map for your domain.

Note: You must run `ypinit` only once, when you first install the host as a slave server. After that, you can perform map updates by using either the `yppush` command from the master server or the `ypxfr` command from the slave server.

4. Start the NIS daemons, `ypserv` and `ypbind`.

Note: The procedure for starting NIS daemons differs from the procedure for starting other system daemons.

An administrator (`root`) can start all daemons manually from the command line. To start the `ypbind` daemon, use the `-h` option, as follows:

```
# /etc/ypserv
# /etc/ypbind -h yourCRAY J90sehostname
```

You may start the daemons manually when you first install NIS to verify that everything is working. After that, the daemons will be started automatically each time the system boots because when you activated the menu in step 1, your NIS domain name was written into the `/etc/config/ypdomain.txt` file. At system startup, the `/etc/ypstart` script accesses the `/etc/config/ypdomain.txt` file, automatically sets the NIS domain name, and then starts the `ypserv` and `ypbind` daemons.

You **must not** add the daemons to the `/etc/config/daemons` file. The `ypstart` script assumes that you are running the Cray Research system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

5. Verify that the CRAY J90se system has bound to itself by using the `ypwhich` command. It is normal to see that the domain has not bound the first time you execute `ypwhich`; simply enter it a second time, as follows:

```
# ypwhich
Domain domainname not bound.
# ypwhich
yourCRAY J90sesystem
```

Procedure 38: Configuring your CRAY J90se system as an NIS slave server without using the menu system

Note: This procedure assumes that another host on the network is already configured as an NIS master server and that your CRAY J90se system can communicate with that host.

To configure NIS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group.

Before you configure NIS on your CRAY J90se system, read the cautionary note and other important information on Section 12.2, page 287.

The following steps explain how to configure your CRAY J90se system as an NIS slave server without using the menu system:

1. Edit the `/etc/config/rcoptions` file and set the `RC_YP=` parameter to `YES`.

2. Assign your system to an NIS domain.

   To set the NIS domain, use the `domainname` command (*domainname* is the name of your NIS domain), as follows:

```
# domainnamedomainname
```

3. Run the `/etc/yp/ypinit` script with the `-s` option to configure the system as an NIS slave server. Include the host name of the NIS master for your domain on the command line, as follows:

```
# /etc/yp/ypinit -s NISmasterserver
```

   Running `ypinit -s` *NISmasterserver* transfers a copy of the NIS maps from the master to the slave server (your Cray Research system running UNICOS) and places it in the `/etc/yp/`*domainname* directory. Running this command also adds the slave server to the `ypservers` map for your domain.

   Note: You must run `ypinit` only once, when you first install the host as a slave server. After that, you can perform map updates by using either the `yppush` command from the master server or the `ypxfr` command from the slave server.

4. Start the NIS daemons, `ypserv` and `ypbind`.

   Note: The procedure for starting NIS daemons differs from the procedure for starting other system daemons.

   An administrator (`root`) can start all daemons manually from the command line. To start the `ypbind` daemon, use the `-h` option, as follows:

```
# /etc/ypserv
# /etc/ypbind -h yourCRAY J90sehhostname
```

   You may start the daemons manually when you first install NIS to verify that everything is working. After that, you should configure the daemons so that they are started automatically each time the system boots; see "To have NIS start automatically when you start UNICOS" at the end of this procedure.

5. Verify that the CRAY J90se system has bound to itself by using the `ypwhich` command. It is normal to see that the domain has not bound the first time you execute `ypwhich`; simply enter it a second time, as follows:

```
# ypwhich
Domain domainname not bound.
# ypwhich
yourCRAY J90sesystem
```

To have NIS start automatically when you start UNICOS

To start NIS automatically when you start UNICOS, specify the NIS domain name by placing the name in the `/etc/config/ypdomain.txt` file, as follows:

```
# echo your_NIS_domain_name > /etc/config/ypdomain.txt
```

When you start UNICOS in the future, the `/etc/ypstart` script will access the `/etc/config/ypdomain.txt` file, set the NIS domain name automatically, and then start the `ypserv` and `ypbind` daemons. You **must not** add the daemons to the `/etc/config/daemons` file. The `ypstart` script assumes that you are running the Cray Research system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

Procedure 39: Configuring user accounts to use NIS

Note: This procedure assumes that your CRAY J90se system has already been configured as an NIS slave server (see the preceding procedure).

You can configure NIS so that a user's password, home directory, and default shell are obtained from the NIS `passwd` map, rather than from the UNICOS user database (UDB).

1. Set the user account `permbits` to `yp` and the `passwd`, `dir`, and `shell` fields to null by using `udbgen`.

   Example:

   ```
   # /etc/udbgen -c "update:john:permbits:yp:passwd::dir::shell::"
   ```

2. Verify that the user database entry is correct, using the `udbsee` command.

   Example:

```
# udbsee john
create  :john:   uid     :10055:
        comment :John Stephen Smith:
        passwd  ::
        gids    :175:
        acids   :10055:
        dir     ::
        shell   ::
        root    :/:
        logline         :/dev/ttyp003:
        loghost         :asbestos:
        logtime         :748554978: # Mon Jan 10 14:56:18 1994
        resgrp          :175: # uid
        permbits        :yp:
        .
        .
        .
```

3. Inform NIS users that they must use the `yppasswd` command to change their password, rather than the `passwd` command. The `passwd` command also will inform NIS users to use `yppasswd` if they forget (see Chapter 7, page 129). The `yppasswd` command works only if you have started the `yppasswdd` daemon on the NIS master server machine.

Configuring NFS [13]

13.1 Related NFS documentation

The following documentation contains information covered in this section:

- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `automount`(8), `biod`(8), `cnfsd`(8), `exportfs`(8), `mount`(8), `mountd`(8), `nfsd`(8), `nfsidmap`(8), and `sdaemon`(8) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `exports`(5) and `fstab`(5) man pages

13.2 What is NFS?

The network file system (NFS) is a Cray Research software product that allows users to share directories and files across a network of machines.

NFS users can use standard I/O system calls, commands, and permission controls to access files from any file system. Similarly, other NFS users can make use of file systems by using remote commands from anywhere in the local network environment. You can use NFS in diverse administrative environments through the use of the ID mapping facility (see Section 13.3, page 298). By default, this facility is on in the UNICOS kernel. The user interface to NFS is transparent.

NFS uses a server/client system to provide access to files on the network. A *fileserver* is any machine that allows a portion of its local disk space to be exported (made available for mounting on a host machine). A *client* is any machine that makes a request for an exported file system. When a user issues an I/O call for a file that resides on a file system mounted by NFS, the call is transmitted to the server machine. When the server receives the request, it performs the indicated operation. In the case of read or write requests, the indicated data is returned to the client or written to disk, respectively. This processing is transparent to users, and it appears that the file resides on a disk drive that is local.

NFS client operations are separate from NFS server operations. This section describes the procedures for configuring a CRAY J90se system as an NFS client and as an NFS server.

For additional information about NFS, including information about the following topics, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304:

- NFS automounter (`automount`(8) command), which is a program that runs on an NFS client that mounts and unmounts NFS file systems on demand. Using the automounter, NFS file systems are mounted only when users are accessing them.
- General security concerns; although UNICOS NFS is an excellent tool for sharing files between computer systems, it also makes the files on a server vulnerable to unauthorized access.
- Kerberos authentication, which can be required for NFS access to exported UNICOS file systems.

13.3 ID mapping and when it is used

In the UNICOS operating system, file access is controlled by checking the numeric user ID (UID) and group ID (GID) against the permissions bits for a file. These same rules apply to NFS. Therefore, in a standard NFS implementation, UNICOS NFS is designed to be used within one administrative domain, which is sometimes called a *flat administrative space*. An *administrative domain* is a set of hosts, usually managed by the same authority, in which all users share a common set of UIDs and GIDs. With the network information service (NIS), a given user or group ID always refers to the same user or group within the administrative domain. This allows all hosts in the NFS group to interpret the authentication information passed in the NFS requests in the same way. Traditional NFS environments make use of NIS (formerly called Yellow Pages) to achieve a flat administrative space. NIS is a distributed look-up service that maintains a common database of UID and GID information for members of an administrative domain. An NIS domain is one administrative domain.

If your Cray Research system resides entirely within one NIS domain and does not interact with hosts outside that domain, you probably will not have to configure NFS ID mapping. However, Cray Research systems are often shared by many different administrative domains, making the creation of a single, flat administrative space for user and group identification technically and/or organizationally difficult. Because a given ID can refer to different users or groups in different administrative domains, this would prevent NFS from being used in such an environment, or would cause serious security problems.

The Cray Research system NFS ID mapping facility allows different administrative domains to participate in cross-mounting NFS file systems without creating a single, flat administrative space.

Following is a description of circumstances in which it is desirable and circumstances in which it is necessary for the Cray Research NFS server to access ID mapping information:

- Account IDs, or ACIDs, which are unique to UNICOS, are not passed across the network as part of the NFS protocol. If ID mapping is configured, NFS servers can use the requesting user's ACID for operations such as file creation. This allows NFS-created files to be charged correctly when using ACIDs for disk accounting and/or file quotas.
- On UNICOS MLS systems (with or without using the IP security option), a UNICOS NFS server must be able to validate requests based on the user's security levels and compartments. If you want to export and serve file systems on a UNICOS MLS system, ID mapping is required.
- If you want to export file systems by using the `-krb` option (Kerberos authentication), ID mapping is required so that the kernel has a place to put a list of authenticated addresses for each Kerberos user.

For additional information on NFS ID mapping, see the Network File System (NFS) section in the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

Procedure 40: Configuring a CRAY J90se system as an NFS client

Note: If you are the administrator on the server(s) and the client(s), you should know whether you exported the file systems you want to mount. To see the file systems that are currently exported, execute the `exportfs` command without arguments on the server.

The following steps explain how to configure a CRAY J90se system as an NFS client:

1. Make entries in the `/etc/fstab` file that describe the file systems you want mounted using NFS. You can use the menu system to do this step or you can do this manually.

   If you are using the menu system, you must first enable the menu system to configure NFS. To give the menu system permission to change the configuration for NFS, change the `NFS configuration` option to `YES` in the `Configure System->Configurator Automation Options` menu. Also, ensure that the `Configure System->Major Software`

`Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

Then, select the `Configure System->File System (fstab) Configuration->NFS File Systems` menu, add your entries, and update the form file. Then activate your changes through the `File Systems (fstab) Configuration` menu. A sample `Network File System Configuration` menu screen follows:

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

```
   Network File System Configuration
     Host    Name              Mount         RW  Quota  Suid  Auto  Bg  So
    -------  --------          ------------  --  -----  ----  ----  --  -- >
E-> tngmoon  /usr/bin          /UTNA/sunbin  ro                     bg  so
```

If you are not using the menu system, edit the `/etc/config/rcoptions` file and set the `RC_NFS=` parameter to `YES`. Then make entries in the `/etc/fstab` file that describe the file systems you want mounted using NFS. The following example shows sample entries; for more information about the options, see the `fstab`(5) and `mount`(8) man pages and *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304:

```
# cat fstab
#
# Mainframe file system table (fstab)
#
#       There are six fields per line, separated by white space.
#       1. device name
#       2. filesystem name
#       3. filesystem type
#       4. mount options
#       5. dump frequency
#       6. pass number to check file system
#
/dev/dsk/root   /                   NC1FS  rw    1    1
/dev/dsk/home   /home               NC1FS  rw    1    2
/dev/dsk/core   /core               NC1FS  rw    1    2
/dev/dsk/usr    /usr                NC1FS  rw    1    2
/dev/dsk/src    /usr/src            NC1FS  rw    1    2
#
# NFS file systems
#
tngmoon:/home/tngmoon/user1     /UTNA/user1     NFS     ro,soft,bg
tngmoon:/usr/bin                /UTNA/sunbin    NFS     ro,soft,bg
```

2. Start the `biod` daemon, which is an optional client daemon that handles write-behind and read-ahead requests. Although this daemon is optional, you should run it to improve NFS performance. By default, four `biod` daemons are started; you might improve client performance by running more `biod` daemons. Ensure that the `biod` daemon is started by using the menu system or by doing it manually.

   Note: To configure NFS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group.

   If you are using the menu system, select the `Configure System->System Daemons Configuration->System Daemons Table` menu, set the `biod` daemon to `YES`, and update the form file. Then activate your change through the `System Daemons Configuration` menu. When you activate this change, the `biod` daemon will be started automatically each time you start UNICOS. A sample `System Daemons Table` menu screen follows:

```
                    Configure System
                        ->System Daemons Configuration
                            ->System Daemons Table
```

```
                      System Daemons Table

    TCP   snmpd     YES  *          /etc/snmpd                    >
    TCP   -         YES  -          /usr/bin/domainname     ""    >
    TCP   portmap   YES  *          /etc/portmap                  >
    TCP   keyserv   NO   *          /etc/keyserv                  >
    TCP   ntpd      NO   *          /etc/ntpd                     >
    NFS   nfsd      YES  *          /etc/nfsd               4     >
    NFS   exportfs  NO   *          /etc/exportfs           -av   >
    NFS   mountd    YES  *          /etc/mountd                   >
E-> NFS   biod      YES  *          /etc/biod               4     >
    NFS   pcnfsd    NO   *          /etc/pcnfsd                   >
    .
    .
    .
```

If you are not using the menu system, edit the `/etc/config/daemons` file and set the NFS `biod` daemon to be `YES`. (Editing this file will ensure that the `biod` daemon will be started automatically each time you start UNICOS in the future.) Then execute the `/etc/sdaemon` script to start `biod` now, as follows:

```
# /etc/sdaemon -s biod
```

Note: You cannot do the remaining steps to this procedure by using the menu system.

3. If you do not want to configure UNICOS NFS ID mapping at this time, you should disable this feature by executing the following command (for information on UNICOS NFS ID mapping, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304:

```
# /etc/uidmaps/nfsidmap -d
NFS ID mapping is disabled.
```

Note: To disable NFS ID mapping permanently, place the`/etc/uidmaps/nfsidmap -d` command in the `/etc/uidmaps/Set.domains` file; otherwise, if you want NFS ID mapping disabled, you must execute the previous step each time you start UNICOS NFS.

4. Create mount points (empty directories in which the NFS file systems will be accessed on your system) for the file systems you will mount using NFS.

   Example:

```
# cd /UTNA
# mkdir user1
# mkdir sunbin
```

5. If you would like the file systems to be mounted using NFS automatically when you start UNICOS, create the `/etc/mountnfs` script and include the appropriate `mount` commands. Based on the preceding examples, a sample script follows:

```
# cat /etc/mountnfs
# Script for mounting NFS file systems
#
mount /UTNA/user1 &
mount /UTNA/sunbin &
```

6. Ensure that the `/etc/mountnfs` script is executable by executing the following command:

```
# chmod +x /etc/mountnfs
```

7. Run the `/etc/mountnfs` script to mount the file systems by executing the following command; when you start UNICOS in the future, the script will be run automatically:

```
# /etc/mountnfs
[1] 85260
[2] 85261
```

8. Verify that the file systems have been mounted by using the `/etc/mount` and `df` commands, as shown in the following examples:

```
# /etc/mount
/ on /dev/dsk/root read/write on Mon Jan 10 08:45:33 1994
/tmp on /dev/dsk/tmp read/write on Mon Jan 10 08:46:11 1994
/usr on /dev/dsk/usr read/write,rw on Mon Jan 10 08:46:12 1994
/home on /dev/dsk/home read/write,rw on Mon Jan 10 08:46:13 1994
/usr/src on /dev/dsk/src read/write,rw on Mon Jan 10 08:46:13 1994
/proc on /proc read/write on Mon Jan 10 08:46:14 1994
/els_src on /dev/dsk/els_src read/write on Mon Jan 10 09:14:58 1994
/UTNA/sunbin on tngmoon:/usr/bin read only,ro,soft,bg on Mon Jan 10 18:13:41 1994
/UTNA/user1 on tngmoon:/home/tngmoon/user1 read only,ro,soft,bg on Mon Jan 10 18:13:41 1994
```

```
# df
/UTNA/user1  (tngmoon:/home/tngmoon/user1):
                                100955  1K blocks ( 48.2%)
/UTNA/sunbin (tngmoon:/usr/bin ): 65879  1K blocks ( 50.7%)
/els_src     (/dev/dsk/els_src ): 178106  4K blocks ( 59.4%)*   57677 I-nodes
/proc        (/proc            ): 119400  4K blocks ( 95.5%)      412 procs
/usr/src     (/dev/dsk/src     ): 27132  4K blocks ( 18.1%)*   25654 I-nodes
/home        (/dev/dsk/home    ): 671031  4K blocks ( 97.8%)*  169883 I-nodes
/usr         (/dev/dsk/usr     ): 152964  4K blocks ( 59.0%)*   26694 I-nodes
/tmp         (/dev/dsk/tmp     ): 495065  4K blocks ( 98.8%)*   98292 I-nodes
/            (/dev/dsk/root    ): 13417  4K blocks ( 17.9%)*   16169 I-nodes
```

Procedure 41: Configuring a CRAY J90se system as an NFS server

The following steps explain how to configure a CRAY J90se system as an NFS server:

1. Describe the file systems you want to allow other systems to mount using NFS by placing entries in the `/etc/exports` file. (For a complete list of export options, see the `exports`(5) man page.) You can use the menu system to place your entries in the `/etc/exports` file or you can do this manually.

   If you are using the menu system, ensure that the menu system has permission to change the configuration for NFS by verifying the `NFS configuration` option is set to `YES` in the `Configure System->Configurator Automation Options` menu. Also, ensure that the `Configure System->Major Software Configuration` menu

has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

Then select the `Configure System->Network Configuration->NFS Configuration->List of Exported File Systems` menu, add your entries, and update the form file. Then activate your changes through the `NFS Configuration` menu. A sample `NFS Exported File Systems Configuration` menu screen follows:

```
Configure System
    ->Network Configuration
        ->NFS Configuration
            ->List of Exported File Systems
```

```
  NFS Exported File Systems Configuration
    File System  Clients  ro  rw clients  Anon  UID  Root Clients  Sum  Ker
    -----------  -------  --  ----------  ----  ---  ------------  ---  --- >
E->  /home                 rw              anon  -2   edge
```

If you are not using the menu system, ensure that the `RC_NFS=` parameter is set to `YES` in the `/etc/config/rcoptions` file. Then edit the `/etc/exports` file to describe the file systems you want to allow other systems to mount using NFS.

2. Look at the list of these file systems by using the `cat /etc/exports` command, as shown in the following example:

```
# cat /etc/exports
/nasc   -root=edge:sn1234
/home   -rw
/tmp
/UTNA/goodstuff
```

3. Make all file systems described in the `/etc/exports` file available to NFS client systems by using the menu system or doing it manually.

   If you are using the menu system, select the `Configure System->System Daemons Configuration->System Daemons Table` menu, set the `Start up at boot time?` option to `YES`, and update the form file. Then activate your changes through the `System`

`Daemons Configuration` menu. A sample `System Daemons Table` menu screen follows:

```
Configure System
    ->System Daemons Configuration
        ->System Daemons Table
```

```
                System Daemons Table

S-> Group                                          NFS
     Name                                           exportfs
     Start up at boot time?                         YES
     Kill action                                    *
     Executable pathname                            /etc/exportfs
     Command-line arguments                         -av
     Additional command-line arguments
     Additional command-line arguments
```

If you are not using the menu system, you can make all file systems described in the `/etc/exports` file available to NFS client systems by executing the `/etc/exportfs -av` command, as follows:

```
# /etc/exportfs -av
```

Caution: If you are not using the menu system, you **must** run the `/etc/exportfs -av` command each time your system is rebooted. You should automate the execution of this command by using the menu system.

4. Enable the NFS server daemons; at a minimum, the TCP/IP `portmap` daemon and the NFS `nfsd` and `mountd` daemons must be started on an NFS server. You can use the menu system to enable the daemons or you can do this manually. The `cnfsd` daemon is necessary **only** when you have more than one Cray Research system; it is intended for use only between Cray Research systems. For more information, see the `exports`(5) and `cnfsd`(8) man pages.

If you are using the menu system, select the `Configure System->System Daemons Configuration->System Daemons Table` menu, set the NFS server daemons to `YES`, and update the form file. Then activate your changes through the `System Daemons Configuration` menu. When you activate this change, the daemons will be started automatically each time you start the UNICOS operating system. A sample `System Daemons Table` menu screen follows:

```
Configure System
    ->System Daemons Configuration
        ->System Daemons Table
```

```
                    System Daemons Table

TCP   snmpd     YES  -           /usr/bin/domainname     ""   >
TCP   -         YES  -           /usr/bin/domainname     ""   >
TCP   portmap   YES  *           /etc/portmap                 >
TCP   keyserv   NO   *           /etc/keyserv                 >
TCP   ntpd      NO   *           /etc/ntpd                    >
NFS   nfsd      YES  *           /etc/nfsd               4    >
NFS   exportfs  YES  *           /etc/exportfs           -av  >
NFS   cnfsd     NO   *           /etc/cnfsd              4    >
NFS   mountd    YES  *           /etc/mountd                  >
NFS   biod      YES  *           /etc/biod               4    >
NFS   pcnfsd    NO   *           /etc/pcnfsd                  >...
```

If you are not using the menu system, edit the `/etc/config/daemons` file to enable the NFS server daemons, as shown in the following example. (If you do not use the menu system, editing this file also will ensure that the daemons will be started automatically each time you start the UNICOS operating system.)

```
# vi /etc/config/daemons
TCP     portmap         YES     *               /etc/portmap
NFS     nfsd            YES     *               /etc/nfsd       4
NFS     cnfsd           NO      *               /etc/cnfsd      4
NFS     -               YES     -               /etc/exportfs   -av
NFS     mountd          YES     *               /etc/mountd
```

Then execute the `/etc/sdaemon` script as follows to start the NFS server daemons (you should have the TCP/IP `portmap` daemon already running):

```
# /etc/sdaemon -g NFS
```

Menu System Overview [A]

This appendix includes information about the following topics:

- Accessing and initiating the UNICOS menu system
- Selecting software components to maintain through the menu system
- Using menu prompts
- Using menu keys
- Using menu definition files
- Sample process of using a menu
- Restoring a configuration
- Viewing the `/etc/install/install.log` log file

This appendix provides a brief description of the UNICOS Installation and Configuration Menu System (hereafter referred to as the *menu system*), which, with system start-up scripts, provides a uniform way to configure and start the various system and network utilities.

If you are upgrading your Cray Research products' software and you want to run the menu system but have never run it before at your site, you must import and activate all parameters. To perform these functions, see the `Utilities` submenu of the main menu. This allows you to maintain your existing configuration files instead of using default configuration files included with a Cray Research product release.

A.1 Accessing and initiating the menu system

The files and scripts that compose the menu system are grouped in the `/etc/install` directory. To access and initiate the menu system, enter the following command lines:

```
cd /etc/install
./install
```

To execute the `./install` script, you must have super-user permission.

To eliminate the need to change (`cd`) to the `/etc/install` directory to enter the menu system, you can include `/etc/install` in your `PATH` statement in your `.profile` or `.cshrc` file.

Note: CRAY J90se systems use a different installation menu system than other Cray Research systems. To use the UNICOS installation menu system's configuration window, CRAY J90se users should first quit the CRAY J90se installation window and then enter the previous commands.

In the X Window System version, the installation tool automatically opens the X Window System version if your workstation or terminal has an X Window System display capability.

A.2 Selecting components to maintain by using the menu system

You can use the menu system to maintain all or selected portions of your system configuration files. Select the following menu and set the components you want to maintain by using the menu system to `YES` (the menu system will access the configuration files for these components):

```
UNICOS Installation / Configuration Menu
    ->SystemConfigure System
        ->Configurator Automation Options
```

If you elect to maintain a component manually, set that component to `NO`. All related menus for that component will be disabled; files cannot be imported into or activated from the menu system.

If you change any settings in the following file, you must rebuild your kernel:

```
UNICOS Installation / Configuration Menu System
    ->Configure System
        ->Major Software Configuration
```

You also should verify that the components you have set to `YES` in the `Configurator Automation Options` menu also are set to `on` in the following menu:

```
UNICOS Installation / Configuration Menu System
    ->Configure System
        ->Major Software Configuration
```

Note: You should **always** import, modify, and/or activate a configuration from the same menu. Although a component's menus may be disabled, if you execute the `Import ...` and `Activate...` lines of the `Configure System` menu, you will import and activate that component's default configuration along with all other default configurations of components that are set to `on` in the `Major Software Configuration` menu.

To import an existing configuration file into the menu system, execute the `Import ...` line of a menu. After you have imported a file, you can modify the configuration within the menu system. When you have the configuration you want, execute the `Activate...` line of the menu. The activation process writes the configuration into the menu system's internal database files to the new `root` (`/`).

A.3 Menu support for full system build

If you wish, you may perform a full system build from within the install tool. The main build menu selection `Release Type` allows you to determine which system components are installed in `/usr/src` and which components will be built:

```
Build / Install System
    ->Release type
```

The default release type for CRAY J90 systems, `Executable`, builds the `uts` component of the system. This installs only the executable installation package onto the system. Otherwise, the selections `Relocatable` and `Source` build all standards components of `/usr/src`.

A.4 Menu prompts

On the left side of a menu display you will be prompted with the following character strings:

String	Description
`M ->`	Means pressing `RETURN` displays a submenu. Each menu display corresponds to a `/etc/install/` *xxx*`.mnu` file written in menu specification language (MSL). Over 30 menu displays exist.
`E ->`	Means pressing `RETURN` takes the horizontal fields to the right of the prompt and displays them vertically as a selection list. These

values are stored in a `.cfg` file in the `/etc/install/cfdb` directory.

`S ->` Means pressing `RETURN` moves the prompt to the far right column, letting the user edit the value or tab through a list of valid selections. These values are stored in `.sav` files in the `/etc/install` directory.

`A ->` Means pressing `RETURN` invokes the action described, such as loading a tape or invoking a build.

`N/A` The current selection is not applicable because of previous selections.

A.5 Menu keys

To access information regarding a specific menu, use the keys provided at the bottom of each menu system screen.

Example:

```
                        Disk Configuration

M-> Physical devices ==>
  Physical device slices ==>
  Logical devices (/dev/dsk entries) ==>
  Mirrored devices (/dev/mdd entries) ==>
  Striped devices (/dev/sdd entries) ==>
  Logical device cache ==>
  Special system device definitions ==>
  Verify the disk configuration ...
  Review the disk configuration verification ..
  Dry run the disk configuration ...
  Review the disk configuration dry run ...
  Update disk device nodes upon activation?    YES
  Import the disk configuration ...
  Activate the disk configuration ...


 Keys: ^? Commands  H Help  Q Quit   V ViewDoc   W WhereAmI
```

Key	Description
`^?  Commands`	Displays a list of currently active command keys that you can enter from the tool.
`H Help`	Accesses the help screen for this particular menu. Provides explanations for each line.
`Q Quit`	Allows you to get out of the menu system from this screen.
`V ViewDoc`	Allows you to get information from a man page from within the menu system.
`W WhereAmI`	Displays the list of menus that you have traversed.

For information about the menu prompts, maneuvering keys, and function keys, see the *UNICOS Installation Menu System Reference Card*, Cray Research publication SQ–2411. To change the value of a `selection` item in a menu, use one of the input keys. Input keys are set to emulate the functions of one of the text editors, which you choose in the `Preferences` menu.

For detailed information about the menu system, see the *UNICOS Installation and Configuration Tool Reference Manual*, Cray Research publication SR–3090.

A.6 Menu definition files

Menu definition files use the menu specification language (MSL) to define the menus displayed by the menu system program (`inmenu`). Menu definition files are identified by the file-name suffix `.mnu`. The following are the basic directories, files, and scripts that the menu system uses:

Path/File	Description
`/etc/config`	Directory that contains the start-up configuration files that the menu system maintains.
`/etc/install`	Directory that contains all menu system interface scripts (`*.sh` and `*.mnu`) and some of the current menu system values (`*.sav`).

`/etc/install/listings`	Directory used by the menu to hold listings that reflect errors.
`/etc/install/cfdb`	Directory of form files (`.cfg`) that the menu system modifies and uses to replace the corresponding system files (`hosts`, `ldchlist`, and so on).
`/etc/install/editions`	`cpio` archives of menu system changes (all `.sav`, `cfg`, and `/dev` changes)

A.7 Sample process of using a menu

To use the menu system to make a change to the `/etc/fstab` file, you must traverse these menus:

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

Make the changes for `/etc/fstab` on this menu. When you have completed your changes, exit this submenu. You are prompted as follows:

```
Do you want to update the form file?  y/n
```

Answering `yes` updates **only** `/etc/install/cfdb/fsmf.cfg`; answering `no` eliminates any changes you made on this menu.

If you want the change dispersed into the real system source, you must execute the following action entry:

```
Activate the configuration
```

Activating the configuration overwrites `/etc/fstab` with the contents of the form file `/etc/install/cfdb/fsmf.cfg`. The menu system displays which files it will update (in this case `/etc/fstab`) and prompts you with the following:

```
continue? y/n
```

This question means that you must give permission for copying the `fsmf.cfg` file to `fstab`. If you answer `yes`, it will copy the file; if you answer `no`, it will not copy the file. This is your last chance to back out.

A `cpio` archive file also is created each time you activate a configuration and a file is actually updated. The `cpio` archive file is a snapshot of all menu system settings; therefore, you can use it at a later date to restore the menu system to a specific state.

The `cpio` archive file contains the `.sav` and. `cfg` files, and all special files in the `/dev` directory. The `cpio` archive is in `/etc/install/editions` and can be restored in `/tmp`, where you can examine or copy all or some files to the appropriate directory.

A.8 Restoring a configuration

Occasionally, you may have to restore a prior iteration of the menu system's configuration files. You can use the menu system for configuration management. Use the following menu sequence to do the following tasks:

- List each stored configuration edition
- Compare two configuration editions
- Extract either a complete edition or individual configuration files within an edition
- Compress the files
- Convert pre-UNICOS 8.0 editions
- Print the listing of available configuration editions
- Store a complete edition of the current system configuration (a snapshot of the system configuration files in their current state)

```
UNICOS Installation / Configuration Menu System

M
    ->U

M
    ->Configuration editions utility
```

A.9 Viewing the /etc/install/install.log log file

The menu system provides a log file, /etc/install/install.log, which you can use to monitor actions, including any errors and problems. To examine this file within the menu system, use the following menu sequence:

```
UNICOS Installation / Configuration Menu System
    ->Utilities
        ->Inspect the installation log
```

To view this file from outside the menu system, enter the following command:

```
$ more /etc/install/install.log
```

File Version Numbers [B]

In the course of normal system administration, occasions exist when it is important to make a new version of a file, but it is also important to keep an old file. A sequence of operations often used to replace a real/production file with a new one and keep an old version of a file is as follows:

```
cp file.REAL file.NEW
```

edit file.NEW

```
cp file.REAL file.OLD
mv file.NEW file.REAL
```

This sequence can lead to problems; if a small error was made in the generation of the new file and a subsequent version is made, reusing the sequence will cause the loss of the previous real file. Because this is a well-known problem, you should use one of the following two sequences. To correct the error, use the following command lines:

```
cp file.REAL file.NEW
```

edit file.NEW

```
cp file.NEW file.REAL
```

To start again but to keep a copy of the broken new file, use the following command lines:

```
cp file.OLD file.NEW
cp file.REAL file.OLD2
```

edit file.REAL

A better strategy is to dispense with the `.OLD` file naming convention and use the following sequences. The first time you want to alter a file, use the following sequence:

```
cp file.REAL file.000
cp file.000 file.001
```

edit file.001

Each time you are ready to save the latest version of a file, copy the highest number file to *file*`.REAL`.

This method of file version numbering has three main advantages over the `.OLD` file naming scheme:

- You can quickly see a version history.
- You can make as many versions of the file as you like without losing the real file.
- You have a backed up copy of the real file in case it gets damaged in production.

Each time you make a new version, you can add a comment in the file's history file, as follows:

`echo "`*file*`.`*00x*` version comment" >> `*file*`.HISTORY`

Disk Capacities and Transfer Rates [C]

For up-to-date information about disk device capacities and transfer rates for disk drives supported on CRAY J90se systems, see the following man pages:

`disksfcn`(7)	Physical specifications of disk devices connected to the Fibre Channel I/O Node (FCN-1).
`disksipn`(7) or `diskspec`(7)	Physical specifications of disk devices connected to the IPI-2 I/O Node (IPN-1).
`disksmpn`(7)	Physical specifications of disk devices connected to the Multipurpose I/O Node (MPN-1).

Logical Device Cache Process [D]

Logical device cache (ldcache) is an optional feature that you may enable to reduce disk I/O wait time from a user's perspective. On CRAY J90se systems, you define the logical device cache in central memory (DDRAM).

When a process issues a read request of data on a file system, the action taken to access the data depends on whether the data is currently in the UNICOS system buffer cache or logical device cache. The process is described as follows:

1. If the data is found in the system buffer cache (central memory), it is copied to the user area. If the requested data is not found, step 2 is taken.
2. If logical device cache has been allocated for the file system, the ldcache area is searched for the sector of data. If found, it is read into the system buffer cache and then copied to the user's process space. If the desired data is not found, step 3 is taken.
3. If ldcache is allocated for the file system, the sector is read from disk and cached into the ldcache area. The sector is then read from the ldcache device into the system buffer (central memory) cache and then copied to the user area.

   Note: The system buffer cache may be bypassed if the data is a multiple of 512 words, begins on a word boundary, and the file system address of the data is on a block boundary.

The system buffer cache writes only to the ldcache area. When system buffers age and require reassignment, the system buffers are written to ldcache and the ldcache segment is marked as dirty. Dirty segments in ldcache are then written to disk when the segment is needed for a different part of the file system, when the ldcache area is flushed to disk by `ldsync`(8), or when the system periodically flushes the ldcache area to disk.

D.1 Setting up ldcache by using `/etc/ldcache`

The cache for a logical device is specified as a number of units and a count of 4096-byte blocks per unit. You can easily configure the relationship between the number of cache units and the cache unit size. The `/tmp` and `root` (`/`) file systems are excellent candidates for logical device cache. If you have more ldcache area available, distribute the remaining area to other heavily used file

systems. To be effective, the ldcache hit rate should be above 97% for ldcaching; however, the main concern is the ratio of logical reads to physical reads.

The `/etc/ldcache` command assigns groups of blocks, called *units*, of an ldcache device (central memory) to a specific file system. To set the number of blocks in an ldcache unit, use the `ldcache -s` command. Choose the size that is used in the `mkfs` command to build that file system. This makes read and write operations to that physical device much faster.

If a striped file system is cached, multiply the number of blocks per cylinder for the physical device type by the number of devices in the stripe group. Larger unit sizes are good for sequential I/O, but they may cause excessive I/O when the I/O is random.

Ensure that the number of blocks assigned for ldcache for all file systems added together does not exceed the total number of blocks available on your logical cache device. To calculate this figure, use the following steps:

1. For each file system being ldcached, multiply the number of blocks in an ldcache unit by the total number of ldcache units allocated for that file system.
2. Add all such totals together.
3. Subtract that sum from the total number of blocks available on the ldcache device for ldcaching.

D.2 Assigning ldcache

When assigning logical device cache, be sure to include the type. The `MEM` type is used when assigning central memory-based logical device cache. The `LDCHCORE` value defines the number of blocks of core memory to be used for logical device cache. The configuration specification language (CSL) `NLDCH` value defines the number of cache headers that will be configured. This sets the total number of logical device cache units that can be active at one time. You must use both the CSL `LDCHCORE` and `NLDCH` statements in conjunction to define central, memory-based logical device cache.

`/etc/ldcache -l` *dev* `-n` *units* `[-s` *size*`]` `[-t` *type*`]`

`-l`	Full path name or minor device number of logical device.
`-n` *units*	Number of cache units to assign. If 0, the logical device caching is released.

`-s` *size*	Size (in 4-Kbyte blocks) of each cache unit. For best performance, set *size* as a multiple of tracks per cylinder related to the logical device and the file system used.
`-t` *type*	Type of memory for cache (`MEM`).

An example of releasing a logical device cache follows:

```
# /etc/ldcache -l /dev/dsk/user_a -n 0
```

An example of assigning a logical device cache follows:

```
# /etc/ldcache -l /dev/dsk/source-tree -s 27 -t MEM -n 500
```

You also can assign a logical device cache by creating an `/etc/config/ldchlist` file, which contains logical device cache configuration information used by `/etc/rc`. During multiuser startup, the `/etc/rc` script checks for the existence of an `/etc/config/ldchlist` file. If the file exists, `/etc/rc` will configure ldcaching according to the entries and values in the `/etc/config/ldchlist` file. There are four fields per line, separated by space; the first field is the logical device, the second field is the cache type (`MEM`), the third field is the number of cache units, and the fourth field is the size in 4-Kbyte blocks of each unit (usually a track size). The following is an example:

```
/dev/dsk/root MEM 300 27

/dev/dsk/usr  MEM 300 27

/dev/dsk/tmp  MEM 300 27

/dev/dsk/home MEM 300 27
```

The third field multiplied by the fourth field is the total cache area (in blocks) allocated for that file system. The total of the third column is the number of `NLDCH` that you must define in the UNICOS `config` file.

An example of displaying the ldcache hit rate follows:

```
# /etc/ldcache
T unit size    reads     writes     hits    misses   rate   name
- ---- ----  ---------- --------  --------  -------  -----  -----
B  300   27    16727    30354      34865     1799     95.09  /dev/dsk/root
B  300   27    1729      4703       1399      254     84.63  /dev/dsk/home
B  250   27    6702     20794       6191      263     95.93  /dev/dsk/tmp
#
M  200   10      47        11         27        4     87.10  /dev/dsk/src

# ldcache -bCache to user    Cache to disk     Cache/disk ratio
Reads  Writes    Reads  Writes    Read  Write  Total    Name
-----  ------    -----  ------    ----  -----  -----    ----
839155 334505    65016  28772    12.9  11.6  12.5    /dev/dsk/root_b
301039  26871    25616   1628    11.8  16.5  12.0    /dev/dsk/usr_b
 68947  74725    13424  13824     5.1   5.4   5.3    /dev/dsk/spool
   183   1678    18416   1743     0.0   1.0   0.1    /dev/dsk/usr_tmp

# ldcache -l /dev/dsk/tmp /dev/dsk/tmp   Fri Sep 24 14:52:12 1993
```

A hit rate of under 97% probably indicates that the file system is not a good candidate for ldcaching or that you should enlarge the size of that file system's ldcache area if possible. In the preceding display, you should examine the file system usage and ldcache configuration aspects of the `/dev/dsk/home` and `/dev/dsk/src` file systems. You also should examine the ratio of logical reads to physical reads, as shown in the preceding display of the `ldcache -b` example.

An example of displaying ldcache statistics for an individual file system follows:

```
                          Read data     Write data
                          ---------     ----------
Blocks transferred:           689            1296
Avg request length:         1 blks           1 blks
Lst transfer rate:    0.008192 Mbs     0.061236 Mbs
Max transfer rate:    0.135680 Mbs     0.208438 Mbs
Cache hits:                   597             677
Cache misses:                   0              73
Cache hit rate:        1000.000000       90.266667
```

D.3 Flushing data by using /etc/ldsync

You can use the /etc/ldsync command to flush data from all logical device caches to disk. Only data that has been written to a logical device cache, but not to disk, is affected. The /etc/ldsync command does not flush data in the system buffers to disk. During normal operation, the UNICOS system periodically flushes data from the ldcache area to disk; the /bin/sync command does this action.

During a normal UNICOS shutdown, all logical device cache data is flushed to disk. At shutdown time it is important that all ldcache is removed from all file systems. To check that all ldcache is removed, use /etc/ldcache. The command should print just a header, as in the following example:

```
# /etc/ldcache

T  Unit  Size  Reads  Writes  Hits  Misses  Rate  Name
-------------------------------------------------------
#
```

For additional information about when to execute the /etc/ldsync command when shutting down the UNICOS system, see the procedure in Chapter 4, page 25.

Power Up and Down Procedures [E]

This appendix explains how to power up and power down a CRAY J90se system mainframe cabinet

Procedure 42: Powering up a CRAY J90se system

After the system workstation (SWS) has been installed and powered up, perform the following steps in the system console window to continue with the power up procedure:

1. Log in as `crayadm`.
2. Enter `initial0` when the system asks you for the password. A command tool window opens.
3. Verify the date and time by entering `DATE`. Correct the values if necessary. (For additional information, see the *Read Me First* documentation that came with your system.)
4. Place each circuit breaker to the `ON` position.
5. Ensure that the System Ready LED on the Central Control Unit (CCU) is illuminated.
6. Close doors of all cabinets.
7. The system is now ready to boot. Enter the `bootsys -c` command at the SWS. This will load the UNICOS kernel, initiate communication between the maintenance I/O node and UNICOS software, boot UNICOS to single-user mode, start the UNICOS console, and start a console for each mainframe that is booted.

Procedure 43: Powering down a CRAY J90se system

To power down a CRAY J90se system, perform the following steps:

1. Shut down the UNICOS operating system by executing the following commands at a UNICOS prompt (You must have super-user privileges to perform the following commands):

```
# cd /
# /etc/shutdown 120

(this step executes after 120 seconds)

# /bin/sync
# /bin/sync
```

2. Open the front door of the mainframe cabinet.
3. On the control panel, locate the SYSTEM OFF button in the lower-right corner. Push in on the button and release it.

   Note: Pushing the SYSTEM OFF button removes power from all of the cabinets in the system. Verify that each cabinet circuit breaker trips to the 0 (OFF) position. All AC indicator lights on the control panel should now be off.

4. Ensure that all system lights are off.

Memory Configuration Parameters [F]

You can use the following tables, based on your system's backplane configuration, to verify your system's NBANKS, CHIPSZ, and MEMORY parameters.

Table 18. NBANKS values for CRAY J916se 2X2 backplane

Memory label	CHIPSZ	Memory boards	NBANKS value	Mwords
8	M4MCH	2	128	32
0	M4MCH	2	256	64
B	M16MCH	2	128	128
3	M16MCH	2	256	256

Table 19. NBANKS values for CRAY J916se 4X4 backplane

Memory label	CHIPSZ	Memory boards	NBANKS value	Mwords
8	M4MCH	4	256	64
0	M4MCH	4	512	128
B	M16MCH	4	256	256
3	M16MCH	4	512	512

Table 20. NBANKS values for CRAY J932se 8X8 backplane

Memory label	CHIPSZ	Memory boards	NBANKS value	Mwords
V	M4MCH	8	512	128
P	M4MCH	8	1024	256
Y	M16MCH	8	512	512
S	M16MCH	8	1024	1024

IOS and Mainframe Dump [G]

If your CRAY J90se system experiences an IOS (MPN or SPN) assertion panic, processor panic, or a UNICOS system panic, you should perform an IOS dump and mainframe dump so that the data in your system's memory is saved into a file. These dumps are very important and useful for determining probable causes for software or hardware problems. This section explains the procedures for capturing memory dumps on CRAY J90se systems from the system workstation (SWS).

G.1 Send dump results to Cray Research

Send your dump results to Software Product Support at Cray Research corporate headquarters in Eagan, Minnesota, USA for analysis. For tracking purposes, include the CRUISE ticket number and/or the SPR number for the incident. The `unicos` and crash files that are created at the same time as the dump file become the core file system and also should be forwarded for analysis.

G.2 Generating a dump

When either an assertion panic or a processor fault panic occurs, the firmware on the IOS (MPN or SPN) automatically captures a dump of that I/O node and saves it to the system console disk in the `/opt/CYRIdump` file. IOS dumps are usually 9 Mbytes. In cases where an apparent hang or unusual system behavior has occurred, you must manually perform an MPN dump.

To create a dump image of the UNICOS operating system running on a CRAY J90se mainframe over the GigaRing channel, execute the `dumpsys`(8) command. To indicate a reason for the dump, use the `-r` *reason* argument.

The `dumpsys` command dumps one or more I/O node or mainframe system components. If you do not specify any options, `dumpsys` dumps the maintenance host I/O nodes, dumps the other I/O nodes, boots the maintenance host I/O nodes, initializes the rings, boots the other I/O nodes, and dumps the mainframe for all system components specified in the `topology`(5) file. The `topology` file identifies rings, I/O nodes, and mainframes and the manner in which they are connected.

`dumpsys` uses the defaults provided by the lower-level commands, such as `dumpj90`(8), unless they are overridden by specifications in the `options`(5) file.

If you specify one or more system components on the `dumpsys` command line, only those components are dumped. The order in which components are specified on the command line is insignificant.

Caution: Do not execute the `dumpsys` command on a running system. It will cause the system to crash, possibly causing a loss of data or corrupted file systems. Before executing this command, shut down UNICOS according to the methods described in UNICOS administration documentation.

For more information on dumps, see the `dumpsys`(8) man page and the *SWS-ION Administration and Operations Guide*, Cray Research publication SG–2204.

Index

B

C

D

E

F

M

N

T