UNICOS® Basic Administration Guide for Cray J90™, Cray J90se™ and Cray SV1™ Series Systems

S–2309–10008

Copyright © 2000 Cray Inc. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Cray Inc.

LIMITED AND RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in the Rights in Data clause at FAR 52.227-14 and/or in similar or successor clauses in the FAR, or in the DOD, DOE or NASA FAR Supplements. Unpublished rights reserved under the Copyright Laws of the United States.

Autotasking, CF77, Cray, Cray Ada, Cray Channels, Cray Chips, CraySoft, Cray Y-MP, Cray-1, CRInform, CRI/*Turbo*Kiva, HSX, LibSci, MPP Apprentice, SSD, SUPERCLUSTER, UNICOS, UNICOS/mk, and X-MP EA, are federally registered trademarks and Because no workstation is an island, CCI, CCMT, CF90, CFT, CFT2, CFT77, ConCurrent Maintenance Tools, COS, Cray Animation Theater, Cray APP, Cray C90, Cray C90D, Cray CF90, Cray C++ Compiling System, CrayDoc, Cray EL, Cray J90, Cray J90se, Cray J916, Cray J932, CrayLink, Cray NQS, Cray/REELlibrarian, Cray S-MP, Cray SSD-T90, Cray SV1, Cray T90, Cray T94, Cray T916, Cray T932, Cray T3D, Cray T3D MC, Cray T3D MCA, Cray T3D SC, Cray T3E, CrayTutor, Cray X-MP, Cray XMS, Cray-2, CSIM, CVT, Delivering the power . . ., DGauss, Docview, EMDS, GigaRing, HEXAR, IOS, ND Series Network Disk Array, Network Queuing Environment, Network Queuing Tools, OLNET, RQS, SEGLDR, SMARTE, SUPERLINK, System Maintenance and Remote Testing Environment, Trusted UNICOS, and UNICOS MAX are trademarks of Cray Inc.

Anaconda is a trademark of Archive Technology, Inc. DynaText and DynaWeb are trademarks of Enigma. EMASS and ER90 are trademarks of EMASS, Inc. ESCON and IBM are trademarks of International Business Machines Corporation and MVS is a product of International Business Machines Corporation. Kerberos is a trademark of the Massachusetts Institute of Technology. NFS, NGS, ONC+, Open Windows, Solaris, Sun, SunOS, Sun Soft, and Sun Workstation are trademarks of Sun Microsystems, Inc. PostScript is a trademark of Adobe Systems, Inc. Sabre is a trademark of Seagate Technology, Inc. SecurID is a trademark of Security Dynamics, Inc. SPARC and SPARCstation are trademarks of SPARC International, Inc., licensed exclusively to Sun Microsystems, Inc. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited. VxWorks is a trademark of Wind River Systems, Inc. X/Open is a trademark of X/Open Company Ltd. The X device is a trademark of the Open Group.

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

New Features

UNICOS® Basic Administration Guide for Cray J90™, Cray J90se™ and Cray SV1™ Series Systems S–2309–10008

This is the original publication of a consolidation of the content from the *UNICOS Basic Administration Guide for Cray J90se and Cray SV1 Series GigaRing based Systems* and *UNICOS Basic Administration Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems* manuals. Most information is similar; any differences between the information for GigaRing based systems and Model V based systems are clearly labeled.

Record of Revision

Version	***Description***
10008	November 2000 Original printing to support the 10.0.0.8 release of UNICOS on the Cray J90se and Cray SV1 series GigaRing based systems and Cray J90, Cray J90se, and Cray SV1 series Model V based systems.

Contents

Page

Page

Preface

This guide is written for system administrators. It includes information required for basic system administration of Cray J90se or Cray SV1 series GigaRing based systems and Cray J90, Cray J90se, or Cray SV1 series Model V based systems.

Warning: Beginning with the UNICOS 10.0 release, the term *Cray ML-Safe* replaced 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 UNICOS 10.0+ releases, the functionality of the Trusted UNICOS system has been retained, but the `CONFIG_TRUSTED` option, which enforces conformance to the strict B1 configuration, is no longer available.

UNICOS System Administration Publications

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

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

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

Related Publications

In addition to the items in this list, see the following section called "Related Man Page Manuals".

Topics described in this guide list publications that you can read to get a greater understanding of those topics. However, the following list identifies topics not covered in this guide that you may wish to pursue to best administer your Cray J90, Cray J90se or Cray SV1 series system.

For information about	Read
File system space monitoring	*UNICOS Resource Administration*; `df`(1) and `du`(1) man pages
File system quotas	*UNICOS Resource Administration*; `qudu`(8), `quadmin`(8), `mount`(8), and `quota`(1) man pages
System activity monitoring	*UNICOS Resource Administration*; `sag`(1), `sar`(8), `sdc`(8), `tsar`(8), and `timex`(1) man pages

Automated incident reporting (AIR)	*UNICOS Resource Administration*; `aird`(8), `airdet`(8), `airprconf`(8), `airsum`(8), and `airtsum`(8) man pages
Job and process recovery	*General UNICOS System Administration*; `chkpnt`(1), `chkpnt`(2), and `crash`(8) man pages
Reinstalling your system software	*UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems* or *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems*
Updating your system software	*UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems* or *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems*
Using the `cron`(8) and `at`(8) utilities	*General UNICOS System Administration*; `at`(1) and `cron`(8) man pages
Configuring network interfaces	*UNICOS Networking Facilities Administrator's Guide*
Monitoring networks	*UNICOS Networking Facilities Administrator's Guide*
Unified Resource Manager (URM) centralizes resource allocation with a formal method of communication	*UNICOS Resource Administration*
Fair-share scheduler	*UNICOS Resource Administration*; `shradmin`(8) and `shrdist`(8) man pages
Memory scheduling	*UNICOS Resource Administration*
Multilevel security (MLS)	*General UNICOS System Administration*
UNICOS message system	*UNICOS Message Reference Manual*; `explain`(1) man page
Data migration facility (DMF)	*Cray Data Migration Facility (DMF) Administrator's Guide*`dmmode`(2), `dmofrq`(2), `dm`(4) and `dmf_offline`(3C) man pages

Tape subsystem	*Tape Subsystem Administration*

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.

Related Man Page Manuals

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

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

- *UNICOS User Commands Reference Manual*
- *UNICOS System Calls Reference Manual*

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

Obtaining Publications

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

To order a document, call +1–651–605–9100. Cray employees may send e-mail to `orderdsk@cray.com`

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, pathnames, signals, messages, programming language structures, and e-mail addresses) and items that appear on the screen.
`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.

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

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

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

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

Reader Comments

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

You can contact us in any of the following ways:

- Send e-mail to the following address:

  `pubs@cray.com`

- Send a fax to the attention of “Software Publications” at: +1–651–605–9001.

- File an SPR; use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

- Call the Software Publications Group, through the Cray Support Center, using one of the following numbers: 1–800–950–2729 (toll free from the United States and Canada) or +1–651–605–8805.

- Send mail to the following address:

  Software Publications
  Cray Inc.
  1340 Mendota Heights Road
  Mendota Heights, MN 55120

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.

Note: Before you consult any of the procedures in this document, you must first boot your SIO and UNICOS software in multiuser mode. See *General UNICOS System Administration*, and the *Open Me First* document for more information.

Procedures may be labeled with a number (for example, Procedure 7). This number is for your reference only. It does not indicate any order to perform it, nor does it indicate any relation to any other procedure. If a procedure itself contains steps or items that are numbered, that numerical order within the procedure may be significant.

1.1 The UNICOS System Administrator

The UNICOS system administrator maintains the Cray UNICOS computing environment for its users. You can see a more detailed list of these duties by looking at the topics listed in the Table of Contents of this book. This person is responsible for the following items:

- 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.
- Creating and maintaining a log book.

The titles of the major topics listed in the Table of Contents of this book serve as a list of the many system administrator duties.

1.1.1 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.

1.2 Major Features of UNICOS

The 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.

1.2.1 High-performance I/O — Model V Based Systems

The UNICOS operating system can perform asynchronous I/O operations, used in multitasking applications, which allow an I/O request to proceed while the main processing continues to execute. *List I/O* permits a linked list of I/O requests by using either synchronous or asynchronous control. Another type, known as *raw I/O*, moves data directly into a user's process space, bypassing kernel system buffers.

1.2.2 Cray Scalable I/O (SIO) — GigaRing Based Systems

The Cray Scalable I/O (SIO) architecture consists of an array of I/O nodes (IONs) connected by a 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 Cray J90se, and Cray SV1 series GigaRing based systems.

For more details about the GigaRing, see Section 1.5.1, page 9.

1.2.3 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.

1.2.4 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.

1.2.5 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.

1.2.6 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 `/etc/passwd` file is an ASCII file, which is the UNIX equivalent of this data file (contains one entry for each user on the system), and is automatically maintained for compatibility with older system versions.

1.2.7 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.

1.2.8 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.

1.2.9 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.

1.2.10 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 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.

1.2.11 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 systems to become a peer node of any established TCP/IP network and permits other users and networks to access the UNICOS environment.

1.2.12 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 *UNICOS System Configuration Using ICMS*.

1.2.13 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.

1.2.14 System Activity Monitor (SAM)

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

For more information see *UNICOS Resource Administration*.

1.2.15 Network Queuing Environment (NQE)

The optional Cray 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*.

1.3 UNICOS Online Glossary

The `define`(1) command allows quick, online retrieval of 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.

1.4 Scalar Cache

Scalar cache is supported and enabled by default. This feature was disabled in the initial software release for the Cray J90 system due to possible data corruption, miscalculations, and/or ORE aborts. The following information describes the fix to the problem and identifies other areas that are affected by enabling scalar cache.

The fix to the scalar cache problem is in the following areas:

- `scanit`(1) command

  When this command is run against an executable, it looks for code sequences that create a problem when run with scalar cache enabled. The `scanit` command changes the code so that the sequence of instructions do not create a problem. The `scanit` command may be used on other vendor-supplied programs if, and only if, agreements with those vendors permit the programs to be modified by the customer.

  This code also sets a bit in the executable's header that lets the UNICOS kernel know that the code has been filtered by `scanit`.

- `segldr`(1) command

  The `segldr` command has the same module integrated into it as is used by the `scanit`(1) command. This module is run when the user's code is linked and the target platform is a Cray J90se or Cray SV1 series GigaRing based system or a Cray J90, Cray J90se, or Cray SV1 series Model V based system

- Kernel portion

  The kernel portion of the fix will query the bit in the executable's header to see whether scalar cache can be utilized for the program. If this bit is set, then the kernel code will attempt to use scalar cache, with some exceptions.

  The kernel utilizes a bit in the exchange package to turn scalar cache on or off in the hardware. When the kernel sets this bit, scalar cache is enabled for the process. When this bit is clear, scalar cache is disabled.

  The kernel will turn off scalar cache for processes that are part of a multitasking group. When the kernel sees that the program is multitasked, scalar cache will be disabled from that point on to the end of program execution. Scalar cache cannot be automatically reenabled when a program reaches the end of a multitasking region unless the associated tasks explicitly exit.

  To ensure that your program is executed entirely with scalar cache, set the environment variable `NPCUS=1` before calling the program to execute. For example:

  ```
  # NCPUS=1; export NCPUS
  # ./a.out
  ```

  Certain programs executed with cache enabled may not see a benefit in terms of decreased run time. This may be due to the piece of the program executing a multitasked library routine that turns off scalar cache until completion of the program. To verify that a program will benefit from scalar cache, perform the following steps:

  1. Run the program without recompiling on a system running UNICOS 8.0.4B or later. This ensures that code will not have the scalar cache bit set in the exchange package and will not enable scalar cache for the process. Use the `timex`(1) command to measure the real, `usr`, and system times by executing:

     ```
     # timex ./a.out
     ```

2. Recompile and relink the program on a Cray J90 as the `TARGET`. If the program cannot be recompiled, run the `scanit`(1) command against the program executable. This will ensure that the code is enabled for scalar cache execution.

3. Run the program again using the `timex` command to measure the real, `usr`, and system times by executing the following:

   ```
   # timex ./a.out
   ```

4. Compare the times received from step 1 against those from step 3. If the times from step 3 did not decrease, the program may have executed a multitasked operation. When a multitasked operation is encountered, the UNICOS kernel automatically disables scalar cache for the duration of the process. To verify that the program has performed a multitasked operation, execute the following:

   ```
   # NCPUS=1; export NCPUS
   # timex ./a.out.
   ```

5. Compare the times received from the `timex` command in step 4 against the times in step 3.

   If the times in step 4 decreased, the program has a multitasked operation in it but would benefit more with scalar cache. In this case, restrict the program to be executed in scalar mode by setting `NCPUS=1` before executing the program.

   If the times received in step 4 are greater than those in step 3, the program has enough multitasked operations so that it benefits more from a multitasked environment. Continue to run code as normal.

1.5 Cray Scalable I/O Overview — GigaRing Based Systems

The GigaRing scalable input/output (I/O) and networking channel supports the I/O peripherals for the Cray T90, Cray T3E, Cray J90se, and Cray SV1 series systems.

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

- Performance
- Reliability
- Maintainability

- Availability
- Scalability

1.5.1 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 *GigaRing* high-speed system 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 mainframes) and IONs that support I/O peripherals.

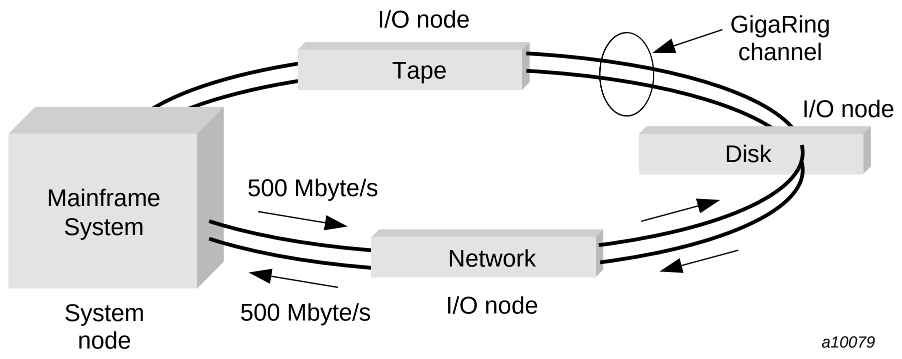


Figure 1. GigaRing Channel Topology

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.

1.5.1.1 I/O Nodes

Cray 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 1.5.1.1.1, 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 1.5.1.1.2, page 13).

Note: The MPN-1 contains a 90–MHz hyperSPARC microprocessor.

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 involved in the hot swap.

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 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 HIPPI 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.
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 space for 10 drives supported by two redundant power supplies and alternate path access.
DSF-2	Fibre disk subrack	Contains space for 10 3.5 inch disk drives supported by two redundant power supplies and alternate path access.
DSS-1	SCSI disk subrack	Contains n+1 power and space for up to 8 SCSI-2 interface disk drives.
GigaRing channel support equipment		
FOX-1	Fibre Optic GigaRing channel extender	Optical extension to the standard GigaRing channel.

1.5.1.1.1 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, Cray T3E, Cray J90se, and Cray SV1 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 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.

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).

For more information about SPNs, see *Cray Scalable I/O Functional Overview*.

1.5.1.1.2 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 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.

For more information about multipurpose nodes (MPNs), see *Cray Scalable I/O Functional Overview*.

1.5.1.1.3 Solid-state Storage Device

The next generation of Cray 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.

1.5.1.2 System Workstation for GigaRing Environments

The system workstation (SWS) serves in these capacities:

- As the operator workstation for Cray T90, Cray T3E, Cray J90se, and Cray SV1 series of systems.

- As a maintenance platform, providing support for diagnostics, GigaRing channel 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.*

Basic System Security [2]

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.

2.1 Related Basic System Security Documentation

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

- *UNICOS Administrator Commands Reference Manual*: `diskusg`(8) man page
- *General UNICOS System Administration*
- *UNICOS User Commands Reference Manual*: `chown`(1), `du`(1), `find`(1), `login`(1) `su`(1), and `umask`(1) man pages

2.2 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 four areas: system security (ensuring that the super-user privileges are safe), user security, partition security, and tape device access.

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.

2.2.1 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.

2.2.1.1 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 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).

2.2.1.2 Physical Security — GigaRing Based Systems

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 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.

2.2.1.3 Physical Security — Model V Based Systems

A person with access to the console 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 recommends that the console 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.

2.2.1.4 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 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.

2.2.1.5 `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.

2.2.2 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.

2.2.2.1 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.

2.2.2.2 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 2.2.2.4, page 20. The current directory should thus be the last entry in the default `PATH`, after the normal system directories.

2.2.2.3 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.

2.2.2.4 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.

2.2.2.5 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)).

2.2.3 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`.

2.3 Tape Device Access

You can access tape devices by using a kernel device driver, the tape daemon, and the character-special tape interface to the UNICOS tape subsystem.

The tape daemon assists the device driver by performing additional functions such as tape resource management, device management, volume mounts and dismounts through operator communication or autoloader requests, label processing, volume switching, and error recovery. The character-special tape interface does not support these capabilities. The tape daemon is recommended for general use.

The character-special tape interface provides unstructured access to the tape hardware similar to the traditional UNIX method of accessing tape devices. It is not recommended for general use, but it is useful in performing specific tasks.

- System administrators use the interface for routine tape manipulations such as copying, particularly for porting tapes from one UNIX system to another. To manage your tapes, you 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*, and the *Tape Subsystem User's Guide*.

Startup and Shutdown [3]

This chapter includes procedures to do the following:

- Start your Cray J90se or Cray SV1 series GigaRing based system mainframe.
- Start your Cray J90, Cray J90se, or Cray SV1 series Model V based system mainframe.
- Bring up UNICOS to a multiuser run state mode (startup; also called *booting*) on Model V based systems and on GigaRing based systems.
- Bring UNICOS back to single-user mode (shutdown).

This chapter also briefly describes several start-up scripts, configuration scripts and files, the aspects of the start-up process that can be customized for your site, and run-level configuration information.

For information about power up and power down, refer to Appendix G, page 317.

3.1 Startup and Shutdown for GigaRing Based Systems

This section covers only GigaRing based systems. See Section 3.2, page 33 for Model V based systems.

3.1.1 Related Startup and Shutdown Documentation for a GigaRing Based System

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

- *UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems*
- *UNICOS Administrator Commands Reference Manual*: `bcheckrc`(8), `brc`(8), `dmdstop`(8), `fuser`(8), `init`(8), `msgdstop`(8), `rc`(8), `sdaemon`(8), and `shutdown`(8) man pages
- *SWS-ION Administration and Operations Guide*
- *SWS-ION Reference Manual*
- *SWS-ION Release Overview*
- *SWS Solaris Operating System and Devices Installation Guide*

3.1.2 How to Boot the Default Configuration for a GigaRing Based System

Note: For a comprehensive description of startup scripts, see *General UNICOS System Administration*.

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 mainframe, SWS, and MPN must be correctly cabled in order to boot the Cray J90se or Cray SV1 series system. In addition, the SWS and MPN must be booted and running. For more information, see the *SWS-ION Administration and Operations Guide.*

To boot your Cray J90se or Cray SV1 series 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 `bootj90se` or `bootsv1` 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.*

Note: By default, your Cray J90se or Cray SV1 series 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: How to Boot Your GigaRing Based 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:
   ```

3. Log in as user `root` and use the password `initial0`.

   Caution: Change the `root` password by using the `/bin/passwd` command. To guard against intentional or inadvertent damage caused by unauthorized use of super-user privileges, you should change the password now.

4. Finish setting up the basic system environment for your site, such as user accounts, file systems, networking, and so on.

3.1.2.1 Multiuser Mode on a GigaRing Based System

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`
  - `tpdaemon` (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.

3.1.2.2 Typical Tasks You Can Perform While in Multiuser Mode on a GigaRing Based System

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 193. 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`.

3.1.2.3 Dedicated System on a GigaRing Based 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*.

3.1.2.4 Run-level Configuration on a GigaRing Based System

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.

3.1.2.4.1 Changing Run Level on a GigaRing Based System

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.

3.1.2.4.2 Strategies for Using Run Levels on a GigaRing Based System

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.

3.1.2.4.3 Single-user Mode on a GigaRing Based System

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`*xxxx*#) 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.

3.1.3 How to Shut down the UNICOS GigaRing Based System

The `/etc/shutdown` script terminates all user processes and system daemons, releases all logical device cache, and unmounts all UNICOS file systems (except for `root`). Unlike the `/etc/rc` start-up script, the operation of the `/etc/shutdown` script is not altered by any UNICOS control files. You do not have to modify the `/etc/shutdown` script directly.

Procedure 2: How to Shut Down UNICOS on a GigaRing Based System

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 Chapter 8, page 187). 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
# /etc/ldsync
```

   (if a logical cache device (ldcache) is configured on your system)

```
# 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.

3.1.4 Startup, Shutdown, and Configuration Files and Scripts for the UNICOS GigaRing Based System

Note: For a comprehensive description of startup scripts, see *General UNICOS System Administration*.

This section lists the files and scripts that are used for starting up, shutting down, and configuring your system.

The following are UNICOS-resident configuration files and start-up scripts:

File	Description
`/etc/config/daemons`	File listing and daemons to be started during multiuser startup; used by `/etc/sdaemon`. See Chapter 4, page 49.
`/etc/config/rcoptions`	Sets environment variables that control `/etc/rc`.
`/etc/inittab`	Read by `/etc/init` at system boot.
`/usr/src/uts/cf.sn`*xxxx*`/config.h`	Parameter file that defines the UNICOS kernel. You should not change these parameters manually.
`/usr/src/uts/cf.sn`*xxxx*`/sn.h`	Parameter file that defines machine-specific characteristics of your mainframe.

The following are UNICOS shell scripts:

Script	Description
`/etc/bcheckrc`	Checks the system date and time, and verifies the integrity of the UNICOS file systems before being mounted.
`/etc/brc`	Detects presence of a UNICOS system dump.
`/etc/rc`	UNICOS multiuser start-up script.

`/etc/shutdown`	UNICOS shutdown script.

At boot time, the following files are created in the UNICOS `root` (`/`) file system (the root file system is chosen by the `ROOTDEV` line in the IOS `/opt/CYRIos/sn`*xxxx*`/param` file):

File	Description
`/CONFIGURATION`	Contains processed CSL definitions. This file matches the `/opt/CYRIos/sn`*xxxx*`/param` file that was used to boot the system.
`/unicos`	A copy of the running UNICOS kernel. This file is **not** an exact copy of the bootable image that resides on the in `/opt/CYRIos/sn`*xxxx*`/param`.

3.2 Startup and Shutdown for Model V Based Systems

This section covers only Model V based systems. See Section 3.1, page 23 for GigaRing based systems.

3.2.1 Related Startup and Shutdown Documentation for Model V Based Systems

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

- *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems*
- *UNICOS Administrator Commands Reference Manual*: `bcheckrc`(8), `brc`(8), `dmdstop`(8), `fuser`(8), `init`(8), `msgdstop`(8), `rc`(8), `sdaemon`(8), and `shutdown`(8) man pages
- *CRAY IOS-V Commands Reference Manual*
- *CRAY IOS-V Messages*

3.2.2 How to Boot the Default Configuration on a Model V Based System

Note: For a comprehensive description of startup scripts, see *General UNICOS System Administration*.

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.

Procedure 3: How to Boot Your System on a Model V Based System

Note: The IOS-V is case sensitive; enter all lowercase characters on the system console.

To boot the IOS and UNICOS software, enter the following commands at the system console:

1. To invoke the system console, press the right mouse button in the OpenWindows root window and select the `J90 Console`, `J90se Console`, or `SV1 Console` menu item. This invokes the `jcon` command, which will log on remotely to `sn`*xxxx*`-ios0` (`sn`*xxxx* is the mainframe serial number, and `ios0` is used for the initial boot and load of the IOS and the UNICOS software to the system).

2. Start the IOS by loading the appropriate device strategies and drivers and by loading and executing the IOS kernel. To do this, enter the `load` command at the IOS boot prompt, which is `BOOT[sn`*xxxx*`-ios0]>` on your system.

   ```
   BOOT[snxxxx-ios0]> load
   ```

   The IOS load command produces output on your terminal and returns the IOS prompt when complete:

   ```
   snxxxx-ios0>
   ```

   Note: When using the system console, press `CONTROL-a` to toggle from the UNICOS prompt to the IOS prompt. To toggle from the IOS prompt to the UNICOS prompt, press `CONTROL-a`.

3. Start the UNICOS system by entering the `/bin/boot` command at the IOS prompt:

   ```
   snxxxx-iosx> /bin/boot
   ```

   The `/bin/boot` script contains IOS commands that clear the mainframe memory, load the UNICOS kernel and the IOS configuration parameter file, initiate communication between the IOS and the UNICOS system, and begin executing the UNICOS system. The prompt on your system console terminal will be the `root` user prompt (#). It may be preceded by `sn` and your system's serial number, as follows:

```
snxxxx#
```

After executing the initial `/bin/boot` command, the UNICOS system is in single-user mode.

After booting the UNICOS system to single-user mode, you should run the `mfsck`(1) command to check the file systems for inconsistencies as follows:

```
snxxxx-iosx> CONTROL-a (+Press RETURN)

(toggles to the UNICOS console)

# /etc/mfsck
```

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 `fsck`(8).

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]
```

If you are using a WYSE terminal, type the following command lines to solve this problem. To backspace, use the `DELETE` key.

```
:wq
# export TERM=vt100
# resize
# echo $TERM
```

If the console does not respond , it may help to power cycle the WYSE terminal by turning the power off, and then on again.

If you are using the IOS-V system console, type the following command lines:

```
:wq
# TERM=xterm
```

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

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

Note: Before going to multiuser mode, or if you intend to work in single-user mode, you should run `fsck` on the `root` (`/`) file system.

4. 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:
```

5. Log in as user `root` and use the password `initial0`.

Caution: Change the `root` password by using the `/bin/passwd` command. To guard against intentional or inadvertent damage caused by unauthorized use of super-user privileges, you should change the password now.

6. Finish setting up the basic system environment for your site, such as user accounts, file systems, networking, and so on.

3.2.2.1 Multiuser Mode on a Model V Based System

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`
  - `tpdaemon` (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.

3.2.2.2 Typical Tasks You Can Perform While in Multiuser Mode on a Model V Based System

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 193. 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`.

3.2.2.3 Dedicated System on a Model V Based 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*.

3.2.2.4 Run-level Configuration on a Model V Based System

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.

3.2.2.4.1 Changing Run Level on a Model V Based System

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.

3.2.2.4.2 Strategies for Using Run Levels on a Model V Based System

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.

3.2.2.4.3 Single-user Mode on a Model V Based System

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`*xxxx#*) 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.

3.2.3 How to Shut Down the UNICOS System and the IOS on a Model V Based System

The `/etc/shutdown` script terminates all user processes and system daemons, releases all logical device cache, and unmounts all UNICOS file systems (except for `root`). Unlike the `/etc/rc` start-up script, the operation of the `/etc/shutdown` script is not altered by any UNICOS control files. You do not have to modify the `/etc/shutdown` script directly.

Procedure 4: How to Shut Down UNICOS and the IOS on a Model V Based System

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 Chapter 8, page 187). 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
# /etc/ldsync
```

(if a logical cache device (ldcache) is configured on your system)

```
# 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.

3. Optional step. If you want to stop the UNICOS system from running, toggle to the IOS and enter the `mc`(8) (master clear) command, as follows:

```
# CONTROL-a
snxxxx-ios0> mc
```

Note: The Cray J90 IOS-V is case sensitive; enter all lowercase characters on the system console.

Note: You should not reboot the UNICOS system without reloading the IOS.

4. Optional step. At this point, you can stop the IOS software by entering the `reset`(8) command, which returns the IOS boot prompt and puts the system as close as possible to the state it was in after being powered up.

```
snxxxx-ios0> reset
BOOT[snxxxx-ios0]>
```

5. Optional step. Power off your system if you choose to do so (for procedures to power off your system, see Appendix G, page 317, or see your hardware installation manual).

3.2.4 Startup, Shutdown, and Configuration Files and Scripts for IOS and the UNICOS Model V Based System

Note: For a comprehensive description of startup scripts, see *General UNICOS System Administration*.

This section lists the files and scripts that are used for starting up, shutting down, and configuring your system.

Note: The IOS-V is case sensitive; therefore, you must enter the following file names in all lowercase characters on the system console.

The following are IOS-resident configuration and start-up files:

File	Description
`/autoboot`	If the file exists, the IOS automatically tries to load itself and to boot the UNICOS system after any reset or power cycle on your system. This file may contain the absolute path to an alternative IOS kernel to be loaded; otherwise, `/ios/ios` will be loaded, followed by `/bin/boot`.
`/bin/boot`	UNICOS boot script.
`/config`	IOS-V configuration file.
`/sys/*.cfg`	ASIC configuration files created by the `jconfig`(8) command for your system. For details on `.cfg` files, see the `jconfig`(8) man page.
`/sys/param`	Default IOS parameter file that contains configuration specification language (CSL) statements defining physical, striped, and logical disk devices, system disk devices, and kernel parameters.
`/sys/unicos.ymp`	Default UNICOS kernel.

The following are UNICOS-resident configuration files and start-up scripts:

File	Description
`/etc/config/daemons`	File listing and daemons to be started during multiuser startup; used by

	`/etc/sdaemon`. See Chapter 4, page 49.
`/etc/config/rcoptions`	Sets environment variables that control `/etc/rc`.
`/etc/inittab`	Read by `/etc/init` at system boot.
`/usr/src/uts/cf.sn`*xxxx*`/config.h`	Parameter file that defines the UNICOS kernel. You should not change these parameters manually.
`/usr/src/uts/cf.sn`*xxxx*`/sn.h`	Parameter file that defines machine-specific characteristics of your mainframe.

The following are UNICOS shell scripts:

Script	Description
`/etc/bcheckrc`	Checks the system date and time, and verifies the integrity of the UNICOS file systems before being mounted.
`/etc/brc`	Detects presence of a UNICOS system dump.
`/etc/rc`	UNICOS multiuser start-up script.
`/etc/shutdown`	UNICOS shutdown script.

At boot time, the following files are created in the UNICOS `root` (`/`) file system (the root file system is chosen by the `ROOTDEV` line in the IOS `/sys/param` file):

File	Description
`/CONFIGURATION`	Contains processed CSL definitions. This file matches the `/sys/param` file that was used to boot the system.

`/unicos`	A copy of the running UNICOS kernel. This file is **not** an exact copy of the bootable image that resides on the in `/sys/unicos.ymp`.

3.2.5 IOS Prompts, and Permissible Actions on a Model V Based System

You can toggle the system's console screen and keyboard between an interface to the software operating on the IOS and the UNICOS software operating on the mainframe. To toggle between the IOS and the UNICOS console interfaces, use the `CONTROL-a` two-key sequence. You may toggle between the two consoles at any time. If you toggle from one to the other, and get no response, the system to which you toggled may no longer be responding to the console interface. This could happen if that system (either the IOS or UNICOS system) has hung or panicked. In this case, you should be able to toggle back to the original console. This section describes when you will see specific IOS prompts, what the condition(s) of the system may be at that time, and the actions that you can take.

> **Note:** When using the IOS master console, `CONTROL-a` toggles between the IOS and UNICOS prompts.
>
> When going from the UNICOS prompt, after you press `CONTROL-a`, the prompt changes to `sn`*xxxx*`-ios0>`.
>
> When going from the IOS prompt, the UNICOS prompt is not displayed until you press `RETURN`.

3.2.5.1 IOS Boot Prompt on a Model V Based System

The IOS boot prompt is as follows:

`BOOT[sn`*xxxx*`-ios`*x*`]>`

When you see this prompt, the following are possible system conditions:

- The IOS is down; it is running in PROM; no strategies or drivers are loaded.
- The mainframe is down; the UNICOS system is not running.

When the power is turned on and after typing `reset`, you will always see the IOS boot prompt.

From this state, you can perform only the following actions:

- Take an IOS dump (not a UNICOS dump) by typing `iosdump`.

- By using the `tar` command, transfer files between the system console disk and the IOS DAT (rpd03) tape drive.
- Load the IOS kernel, strategies, and drivers into memory by typing `load`. This command also starts the execution of all IOSs defined in the `/config` file.
  - The IOS kernel resides on the system console disk and has the path name `/ios/ios`.
  - The IOS strategies and drivers reside on the system console disk in the `/dev` directory.
  - The IOS `load` command uses the IOS configuration file `/config` to determine which strategies and drivers to load into IOS memory.

To start the IOS by loading the appropriate device strategies and drivers and loading and executing the IOS kernel, enter the `load` command at the IOS boot prompt:

```
BOOT[snxxxx-ios0]> load
```

After loading is complete, the prompt changes to the IOS prompt, which signifies that the IOS software loaded in the IOS memory is now executing instead of the PROM code.

3.2.5.2 IOS Prompt on a Model V Based System

The IOS prompt is as follows:

`sn`*xxxx*`-ios`*x*`>`

When you see this prompt, the following are possible system conditions:

- The IOS is up; it is running the IOS kernel, strategies, and drivers. Any slave IOP in the IOS may or may not be running. Check the `/adm/syslog` file on the system console disk for messages indicating that a slave IOS has panicked if this is suspected.
- The IOS and mainframe are both up; `CONTROL-a` was pressed to change from the mainframe prompt to the IOS prompt.
- The mainframe is down; the UNICOS system is not running. A mainframe system panic has occurred. The IOS is still running, however.

If a mainframe system panic occurs, the IOS may still be running, but it will be in an undefined state. Taking an IOS dump at this point may be helpful; use the `iosdump`(8) command. See the *CRAY IOS-V Messages*.

From this state, you can perform the following actions:

- Run diagnostics.

  Note: Diagnostics should complete successfully and cause no load problems. However, if you have run diagnostics and a failure was detected or the diagnostic did not exit cleanly (for example, if you entered a `CONTROL-c` to exit a diagnostic), the system may have been left in an undefined state. This could cause the system to hang during the boot process. If you experience this problem, enter the `reload`(8) command after the IOS prompt to set the system to a known state, and then start the UNICOS system by entering the `/bin/boot` command after the IOS prompt.

- Take a UNICOS dump by entering the IOS `mfdump`(8) command.
- Flush buffers to disk, reset the VME bus, and return the IOS to PROM (the IOS boot prompt) by entering the IOS `reset`(8) command.
- Master clear the mainframe CPUs, which stops all CPU activity, by entering the `mc` command.
- Clear central memory, as well as load and start the UNICOS system, by entering `boot`.
- Initiate a reboot of the IOS from PROM, and reload the IOS by entering `reload`.

UNICOS System Daemons [4]

This chapter describes how to start and stop UNICOS system daemons; it also includes a sample `/etc/config/daemons` file. A *daemon* is a process that executes in the background; a daemon (the process) is always available.

If you have access to a window environment, UNICOS provides a point-and-click, X Window System based interface to the UNICOS Installation / Configuration Menu System. For more information, see *UNICOS System Configuration Using ICMS*.

4.1 Related UNICOS System Daemons Documentation

The following documentation contains additional information about UNICOS system daemons: *UNICOS Administrator Commands Reference Manual*, `bcheckrc`(8), `brc`(8), `dmdstop`(8), `fuser`(8), `init`(8), `msgdstop`(8), `rc`(8), `sdaemon`(8), and `shutdown`(8) man pages.

Procedure 5: Starting and stopping UNICOS system daemons

You can use the menu system to start and stop UNICOS daemons or you can start and stop daemons manually.

If you are using the menu system, select the following:

```
Configure System
    ->System Daemons Configuration
        ->System Daemons Table
```

Then, select the submenu of the daemon you want to start or stop, and change the `Start up at boot time?` field. When you exit out of the submenu, the `StartOpts` field of the `System Daemons Table` menu will reflect the change you made. As you exit the `System Daemons Table` menu, update the form file, then activate your changes through the `System Daemons Configuration` menu.

Note: All daemons that have YES in the `Start up at boot time?` field will be started automatically in subsequent system startups. If you have changed a daemon setting to be YES in the `Start up at boot time?` field and want to start it before the next system startup, see **To Start One Daemon** in this procedure.

A sample `System Daemons Table` submenu screen and `exportfs` NFS daemon submenu screen follow:

```
Configure System
    ->System Daemons Configuration
        ->System Daemons Table
```

```
                            System Daemons Table

      Group  Name      StartOpts    Kill            Program
      -----  ----      ---------    ----            -------
      SYS1   errdemon  YES          /etc/errstop    /etc/errdemon
      SYS1   cnfsd     NO           *               /etc/shrdaemon
      .
      .
      .
      NFS    cnfsd     YES          *                /etc/cnfsd
 E->  NFS    -         YES          -                /etc/exportfs
      NFS    mountd    YES          *                /etc/mountd
      .
      .
      .

   Keys:   ^? Commands   H Help   Q Quit   V ViewDoc   W WhereAmI
```

```
                            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, edit the `/etc/config/daemons` configuration file to set which daemons to start or stop. You can modify this file by using your preferred UNICOS editor (for a sample `/etc/config/daemons` file, see Section 4.1, page 49).

Note: All daemons that have `YES` in the start field of the `/etc/config/daemons` configuration file will be started automatically when you do subsequent system startups. If you have changed a daemon setting to be `YES` in the start field of the `/etc/config/daemons` configuration file and want to start it before the next system startup, see **To Start One Daemon** in this procedure.

To Start One Daemon

To start or stop a daemon or group of daemons with the arguments that are included in the `/etc/config/daemons` file, use the `sdaemon` (`/etc/sdaemon`)command at any time.

To start one daemon, use the `sdaemon -s`command, as follows:

`/etc/sdaemon -s` *daemon*

To start a group of daemons, use the `sdaemon -s -g` command, as follows:

`/etc/sdaemon -s -g` *daemongroup*

`SYS1` is a group of daemons defined in the daemon configuration file that contains all daemons (such as, the message daemon) that must be started **before** network startup.

`TCP` and `NFS` are the network daemon groups.

`SYS2` is a group of daemons defined in the daemon configuration file that contains all daemons (such as, the NQS daemon) that must be started **after** network startup.

During the shutdown process, daemons are stopped automatically. If you want to stop specific daemons or group(s) of daemons without shutting down your system, you can use the `sdaemon -k` command, as follows:

To stop one daemon, use the `sdaemon -k` command, as follows:

`/etc/sdaemon -k` *daemon*

To stop a group of daemons, use the `sdaemon -k -g` command, as follows:

`/etc/sdaemon -k -g` *daemongroup*

To verify whether a given daemon process was created or killed successfully, use the `ps -e` command.

Note: To identify whether a daemon is running, use the `ps -ale | grep` *daemon_name* command. The maximum length of *daemon_name* is 8 characters; if you use more than 8 characters, no information will be returned to your screen.

For additional information, see the `sdaemon`(8) man page.

A sample `/etc/config/daemons` file follows:

```
# Configuration file for daemons (and other commands) started by /etc/rc
# and other startup scripts (through /etc/sdaemon).
# File format is:
# group tag           start kill                pathname                      arguments
#
SYS1    errdemon      YES   /etc/errstop        /etc/errdemon
SYS1    share         NO    *                   /etc/shrdaemon
SYS1    share         NO    *                   /etc/shradmin            -t100 -F06 -K60s -R4
SYS1    cron          YES   *                   /etc/cron
SYS1    msgdaemon     YES   /etc/msgdstop       /usr/lib/msg/msgdaemon
SYS1    fsdaemon      NO    *                   /etc/fsdaemon
SYS1    fsdaemon      NO    *                   /etc/fsmon               -a all
TCP     myroutes      NO    -                   /etc/myroutes
TCP     gated         NO    /etc/gated.pid      /etc/gated               /usr/spool/gated.log
TCP     named         NO    *                   /etc/named               /etc/named.boot
TCP     inetd         YES   *                   /etc/inetd               /etc/inetd.conf
TCP     talkd         NO    *                   /etc/talkd
TCP     sendmail      YES   *                   /usr/lib/sendmail        -bd -q30m
TCP     printer       YES   -                   /bin/rm                  -f /dev/printer
                                                                         /usr/spool/lpd.lock
TCP     printer       YES   /usr/spool/lpd.lock /usr/lib/lpd             -l
TCP     snmpd         NO    snmpd               /etc/snmpd
TCP     yp_domainname NO    /usr/bin/domainname ""
TCP     portmap       YES   *                   /etc/portmap             -i
TCP     keyserv       NO    *                   /etc/keyserv
TCP     ntpd          YES   *                   /etc/ntpd                -r4
NFS     nfsd          YES   *                   /etc/nfsd                4
NFS     cnfsd         YES   *                   /etc/cnfsd               4
NFS     -             YES   -                   /etc/exportfs            -av
NFS     mountd        YES   *                   /etc/mountd
NFS     automount     YES   *                   /etc/automount           -m -f
                                                                         /etc/auto.master
NFS     biod          YES   *                   /etc/biod                4
NFS     pcnfsd        NO    *                   /etc/pcnfsd
SYS2    scp           NO    /usr/lib/uscpterm   /usr/lib/uscpd
SYS2    syslogd       YES   *                   /etc/newsys              -s
SYS2    tpdaemon      YES   /etc/tpdstop        /usr/lib/tp/tpdaemon     -cr
SYS2    dmdaemon      NO    /usr/lib/dm/dmdstop /usr/lib/dm/dmdaemon
SYS2    NQS           YES   /usr/bin/qstop      /usr/bin/qstart          -i
                                                /etc/config/nqs_config   -c
                                                /usr/tmp/nqs.log
SYS2    samdaemon     YES   *                   /usr/lib/sam/samdaemon
SYS2    air           YES   -                   /usr/air/bin/start_air
```

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 60). 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 systems. If you have Cray systems other than Cray J90se or Cray SV1 series GigaRing based systems or Cray J90, Cray J90se, or Cray SV1 series Model V based systems, see *General UNICOS System Administration*, and *UNICOS Configuration Administrator's Guide*, to determine differences in file systems and how to configure them.

5.2 Related File Systems Documentation

The following publications contain information related to this section:

- *General UNICOS System Administration*
- *UNICOS Configuration Administrator's Guide*
- *UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems*

- *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems*
- *UNICOS Resource Administration*
- *UNICOS User Commands Reference Manual*: `df`(1), `du`(1), `mkdir`(1), and `rm`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*: `dir`(5), `dsk`(4), `fs`(5), `fstab`(5), `inode`(5), `ldd`(4), `mnttab`(5), and `pdd`(4) man pages
- *UNICOS Administrator Commands Reference Manual*: `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 and Cray SV1 series 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 and Cray SV1 series GigaRing based systems, minor number 253 is used for the `ce` partition. For additional information, see *General UNICOS System Administration*.

All UNICOS special files are located in the `/dev` directory or one of its subdirectories. Your 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. For **GigaRing** based systems, this information is stored in the `/opt/CYRIos/sn`*xxxx*`/param` file. For **Model V** based systems, this information is stored in the `/sys/param` file. In both cases the file is written by using the menu system. You can store file systems on disk or in random-access memory (RAM), or a combination of both.

For **GigaRing** based systems the definition of your system's logical and physical disk devices is defined in the `/opt/CYRIos/sn`*xxxx*`/param` file. For **Model V** based systems, the definition of your system's logical and physical disk devices is defined in the `/sys/param` file on the IOS. In both cases 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:

<u>Command</u>	<u>Description</u>
`/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/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 5.7, page 66, and for information about disk banding, see Section 5.8, page 66.) 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 service 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 software products, see the *UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems* **or** *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems.*

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.

On **GigaRing** based systems, for details on peripherals supported by single-purpose nodes (SPNs) and multi-purpose nodes (MPNs), see Section 1.5.1.1, 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 or Cray SV1 series system is 220,000, 440,000, or 660,000 blocks, depending on the type of UNICOS release loaded. This size is sufficient to hold all of the files necessary to relink the UNICOS kernel, if the UNICOS executable package has been loaded. You also must allow enough space in your default value to handle additional 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 UNICOS system is 300,000 blocks. You also must allow enough space in your default value to handle additional 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 5.7, page 66, and *General UNICOS System Administration*.

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 100,000 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` (`/`) Back-up `/usr`, and Back-up `/usr/src` File Systems

Size recommendations: The back-up `root` (`/`) (called `rootb`), back-up `/usr` (called `usrb`), and back-up `/usr/src` file (called `srcb`) systems are equal in size to the original `roota` (`/`), `/usra`, and `srca` 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 Device Characteristics

You may define and mount one or more file systems on disk devices. For more information on disk device characteristics, see Appendix E, page 305.

5.7 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.

For **Model V** based systems, slice members of a stripe group must be previously defined in the physical device statement of the configuration specification language (CSL). In addition to physical CSL definition statements in the IOS `/sys/param` file, a special stripe device definition statement also is required to configure a stripe group. For information about using CSL, see Section 5.9, page 66.

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.

5.8 Disk Banding

The term *disk banding* refers to 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.

5.9 Configuring Your Devices and Their File System Allocation

The system configuration file that configures disks on **GigaRing** based systems is the `/opt/CYRIos/sn`*xxxx*`/param` file on the SWS. The system configuration file that configures disks on **Model V** based systems is the `/sys/param` file on the IOS disk drive.

The Configuration Specification Language (CSL) is used to define the configuration and parameter settings that are used at boot time.

Note: For additional CSL information, see *UNICOS Configuration Administrator's Guide.*

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/sn`*xxxx*`/param` file that describe your system configuration for **GigaRing** based systems or `/sys/param` file that describe your system configuration for **Model V** based systems.

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

- Verify your disk configuration file (Procedure 6, page 68)
- Identify devices defined on your system and how they are allocated to file systems (Procedure 7, page 72)
- Modify your system configuration manually (for a GigaRing based system, see Procedure 8, page 76; for a Model V based system, see Procedure 9, page 78)
- Set up a quota control file (Procedure 10, page 83)
- Create file systems (Procedure 13, page 93)

5.10 Network Disk Array Configuration

For information on configuring network disk arrays (HIPPI disks), see *Network Disk Array (HIPPI Disk) Configuration Options and Performance.*

5.11 CSL Parameter File, Syntax and Usage

For CSL information, such as placement of CSL statements, see the *UNICOS Configuration Administrator's Guide* chapter, "*CSL Parameter 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 6: Verifying your disk configuration file manually

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 `/opt/CYRIos/sn`*xxxx*`/param` configuration file from a **GigaRing** based system follows.

```
/*
 *
 * Configuration parameter file (GigaRing based system)
 *
 */

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 7: 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 sn*xxxx* # 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.

   ```
   snxxxx# cd /etc/install
   snxxxx# ./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.

   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 **GigaRing** based system. The fields are defined in the diagram that follows:

```
$ ddstat /dev/dsk/tmp

/dev/dsk/tmp b 34/74 0 0 /dev/ldd/tmp
           /dev/xdd/tmp_1 c 33/5 01046 1102000 898000 0 0 0 0
           /dev/xdd/tmp_2 c 33/8 01026 1102000 898000 0 0 0 0
```

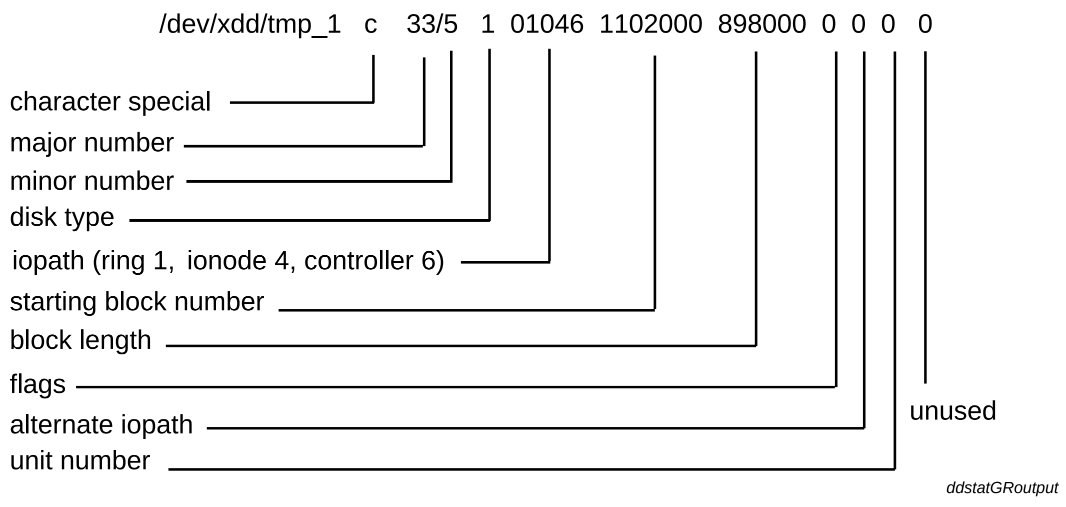


Figure 2. ddstat Output Field Definitions (GigaRing based system)

The following is an example of `ddstat` output from a Cray J90se **Model V** based 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/pdd/tmp_1 c 32/69 12 01036020 0 201600 0 0 0 0
       /dev/pdd/tmp_2 c 32/96 12 01036020 0 201600 0 0 1 0
       /dev/pdd/tmp_3 c 32/97 12 01036020 0 201600 0 0 2 0
```

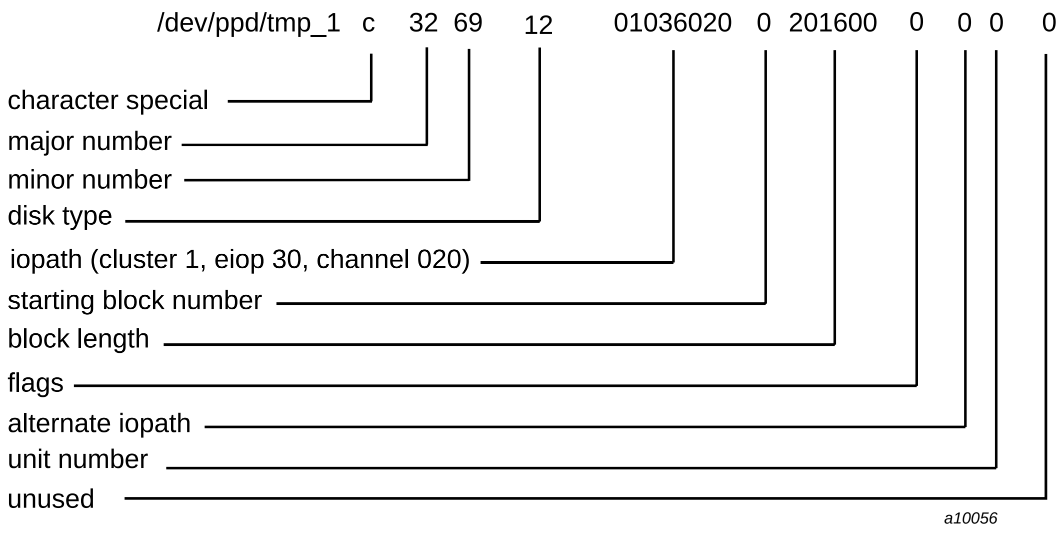


Figure 3. ddstat Output Field Definitions (Model V based system)

Procedure 8: Modifying your configuration file on a GigaRing based system

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

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

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/sn`*xxxx*`/param`) by using the `dump` command. See Section 6.3.1, page 106.

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

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

3. Make sure that you are in `/etc/config` on UNICOS. Copy the configuration file `/opt/CYRIos/sn`*xxxx*`/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/sn`*xxxx*`/param` file will be read from the system disk and be named `new.param`:

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

4. Edit your copy of the parameter file on UNICOS.

```
snxxxx# vi new.param
```

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

```
snxxxx# /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 command. The following command specifies that the `new.param` file will be written to the SWS system disk and be named `/opt/CYRIos/sn`*xxxx*`/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.

```
snxxxx# rcp new.param SWS:/opt/CYRIos/snxxxx/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/sn`*xxxx*`/param` file to the UNICOS `/CONFIGURATION` file.

7. 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:

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

8. 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/sn`*xxxx*`/param` file you changed. (Section 5.15, page 92 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.

Remove the existing devices:

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

Generate the new device definitions:

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

Procedure 9: Modifying your configuration file on a Model V based system

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

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

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 (`/sys/param`) by using the `dump` command. See Section 6.3.1, page 106.

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

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

3. Make sure that you are in `/etc/config` on UNICOS. Copy the configuration file `/sys/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 `/sys/param` file will be read from the system disk and be named `new.param`:

```
snxxxx# exdf -i /sys/param > new.param
```

4. Edit your copy of the parameter file on UNICOS.

```
snxxxx# vi new.param
```

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

```
snxxxx# /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 command. The following command specifies that the `new.param` file will be written to the IOS and named `/sys/param`.

```
snxxxx# exdf -ro /sys/param < new.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 IOS `/sys/param` file to the UNICOS `/CONFIGURATION` file.

7. 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:

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

8. 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 IOS `/sys/param` file you changed. (Section 5.15, page 92 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.

Remove the existing devices:

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

Generate the new device definitions:

```
snxxxx# cd /dev
snxxxx# chmod 755 mkdev
snxxxx# ./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 header.
	3	infinite	Infinite quota (no quota evaluation 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 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*.

Procedure 10: 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/mfsck 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 mfsck. 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 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 Applying Changes

Procedure 11: Applying changes to the new disk layout `param` file

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 Proceeding with Changes

Procedure 12: Proceeding with your file system change plan

Perform the following steps as you proceed:

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 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*, and *UNICOS Resource Administration*, include information on other aspects of file system maintenance. For example, *UNICOS Resource Administration* 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* 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 13: 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. 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. 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*).

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 list (immediately calculated) is not the same as the disk 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.

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 off line

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/sn`*xxxx*`/param` file of a **GigaRing** based system or `/sys/param` file of a **Model V** based system, the mount point will also 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.

5. 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 14: 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 15: 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*.

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
```

Backup and Restore [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*, section on file system planning
- *UNICOS Configuration Administrator's Guide*
- *UNICOS File Formats and Special Files Reference Manual*: `dump`(5) and `fstab`(5) man pages
- *UNICOS Administrator Commands Reference Manual*: `dump`(8), `rdump`(8), `restore`(8), and `rrestore`(8) man pages
- *Tape Subsystem Administration*: tape-related information

Special books for **GigaRing** based systems:

- *UNICOS Installation Guide for Cray J90se and Cray SV1 GigaRing based Systems*

Special books for **Model V** based systems:

- *CRAY IOS-V Messages*
- *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems*

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*.

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

At initial installation, two sets of production disk partitions are created with the following names:

/dev/dsk/**roota** (/)
/dev/dsk/**usra**
/dev/dsk/usr/**srca**

and

/dev/dsk/**rootb** (/)
/dev/dsk/**usrb**
/dev/dsk/usr/**srcb**

This lets you install into another set of partitions in multiuser mode without disturbing the running system. The 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, on a Cray J90se or Cray SV1 series GigaRing based system or Cray J90, Cray J90se or Cray SV1 series Model V based 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 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 16: Creating `rootb` and `usrb` file systems

You can create a bootable copy of your production `root` and `usr` file systems into file systems called `rootb` and `usrb`. 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 `rootb` and `usrb`.

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 `rootb` and `usrb` 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/rootb
# mkfs  -q -ADEVTYPE SECURITY_OPTIONS /dev/dsk/usrb
```

2. Label the file systems, as follows:

```
# labelit  /dev/dsk/rootb rootb cray
# labelit  /dev/dsk/usrb  usrb cray
```

3. Check file system consistency, as follows:

```
# fsck -u /dev/dsk/rootb
# fsck -u /dev/dsk/usrb
```

4. Mount the file systems, as follows:

```
# mount /dev/dsk/rootb /mnt
# mount /dev/dsk/usrb  /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 `rootb` file system, as follows:

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

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

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

Procedure 17: Booting `rootb` and `usrb` into production

The following are procedures to boot the `rootb` and `usrb` 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 `rootb` and `usrb` 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 `rootb` creation procedure in reverse order to create a new production `root` and/or `usr` file system. Boot to multiuser mode on `rootb` 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
# df
     # /etc/fuser -ku /dev/dsk/filesystem
     # /etc/umount /dev/dsk/filesystem
# sync
```

(wait for system to shut down to single user mode)

```
# ldsync
```

(if a logical cache device (ldcache) is configured on your system)

2. Edit the parameter file.

   For **GigaRing** based systems, edit the `/opt/CYRIos/sn`*xxxx*`/param` file and boot the SIO by executing the following commands (this may be done on the system console, in which case the prompt will be `sn`*xxxx*>):

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

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

For **Model V** based systems, edit the `/sys/param` file and boot the IOS by executing the following commands (this may be done on the system console, in which case the prompt will be `sn`*xxxx*`-ios0>`):

```
IOS> cd /sys
IOS> cp param param.prd
IOS> cp param param.bkr
IOS> ed param.bkr
     1
     /rootdev/
     s/ldd root /ldd rootb/p
        rootdev is ldd rootb;
     w q
IOS> cp param.bkr param
IOS> cd /
IOS> boot -c
```

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

```
# fsck -u /dev/dsk/usrb
# mount /dev/dsk/usrb /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, rootb and usrb lines from:


/dev/dsk/root     /          NC1FS  CRI_RC=NO,rw  1  1
/dev/dsk/rootb   /mnt       NC1FS  CRI_RC=NO,rw  1  4
/dev/dsk/usr      /usr       NC1FS  CRI_RC=NO,rw  1  2
/dev/dsk/usrb    /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/rootb   /          NC1FS  CRI_RC=NO,rw  1  1
/dev/dsk/usr      /mnt/usr   NC1FS  CRI_RC=NO,rw  1  2
/dev/dsk/usrb    /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='rootb'
   PIPEDEV='rootb'
   USRDEV='usrb'
```

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 18: Backing up the SWS — for GigaRing based systems

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*.

Procedure 19: Backing up the IOS — for Model V based systems

The following is the procedure for creating a backup copy of the IOS. To back up the Solaris files related to the installation process and the IOS system software, you should refer to the “Backup Console Environment chapter of the *UNICOS Installation Guide for Cray J90, Cray J90se, and Cray SV1 Model V based Systems*.

> **Note:** This procedure applies to DAT devices. It can be run from the IOS console and can be performed regardless of whether the UNICOS system is running. The prompt will be `sn`*xxxx* `-IOS0`.

1. If the tape is not already in a physically writable condition, physically alter the tape so that you can write to it.

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 separate) 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. To see currently supported tape devices, refer to the Cray Product Support Grid in CRInform at:

`http://crinform.cray.com`

For details on how to access this information, see your Cray service representative.

6.9 Procedures for Backup and Restore

This subsection includes the following procedures:

- Backing up (dumping) a file system without `tpdaemon`
- Restoring a full or partial 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 20: Backing up (dumping) a file system without `tpdaemon`

Warning: Cray does not recommend using the non-`tpdaemon` procedures for routine use because of the possibility of overwriting the data on the tape. Instead, Cray recommends that you use the `tpdaemon` procedures whenever possible or be in single user mode when not using `tpdaemon` procedures.

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. ether the tape is in a physically writable condition, load the device and enter the following command:

```
snxxxx# 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 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 a tape, type the following command line:

```
snxxxx# 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:

```
snxxxx# touch /etc/dumpdates
```

The following example shows a full file system dump (`-t 0`) to a square tape (`-f /dev/tape/t120`):

```
snxxxx# /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 21: Restoring a full or partial file system without `tpdaemon`

Warning: Cray does not recommend using the non-`tpdaemon` procedures for routine use because of the possibility of overwriting the data on the tape. Instead, Cray recommends that you use the `tpdaemon` procedures whenever possible or be in single user mode when not using `tpdaemon` procedures.

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:

```
snxxxx# /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:

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

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

```
snxxxx# /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 55, and the `mkfs`(8) man page.

```
snxxxx# /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:

   ```
   snxxxx# /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:

```
snxxxx# /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:

```
snxxxx# cd /
snxxxx# /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:

```
snxxxx# 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/t120`. 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:

```
snxxxx# 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:

```
snxxxx# /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.

```
snxxxx#  /etc/restore -r -f /dev/t120
```

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`.

```
snxxxx# /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:

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

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

```
snxxxx# /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:

```
snxxxx# /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:

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

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

Procedure 22: Backing up (dumping) a file system by using `tpdaemon`

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, a new tape is used, and it will be specified as unlabeled. The volume name used is arbitrary.

Note: Generally, when backing up (dumping) a file system, the system should be in single-user mode and the file system to be backed up (dumped) should be unmounted. An alternate choice is to back up the file system while in multiuser mode with the file system being dumped in the unmounted state.

Caution: If you use `/etc/udbrestict -r -m R` (a system-wide command) to restrict system access, the `/etc/udbrestrict` utility also disables the Network Queuing System (NQS), `cron`, batch, and interactive jobs. When `/etc/udbrestict -r` is started, all checkpointed and all queued NQS jobs of all restricted users will be deleted. So, every process belonging to any restricted account will fail.

If you are in single-user mode (no file systems mounted other than `root` and no daemons started) and you want to back up the `/dev/dsk/usr` file system, you cannot invoke an operator window (needed to answer `tpdaemon` tape-related questions during the backup process), because that command is in `/usr (/usr/lib/msg/oper)`. If you mount `/dev/dsk/usr` so that you can use the commands and daemons that reside in the `/dev/dsk/usr` file system to do the backup and restore, you can perform the backup successfully. You also must start any daemons you need (such as `tpdaemon` and `msgdaemon`) if you are in single-user mode.

While in single-user mode and working from a nonwindow environment, you should run the `dump` command with an & symbol appended so that the command runs as a background process. Running in the background enables you to keep your window free to invoke the operator message window (`/usr/lib/msg/oper`) to answer the operator messages your backup procedure will generate.

1. Determine which tape device group you plan to use for your backup. Device groups are defined in your `/etc/config/tapeconfig` file. Typical device groups are `CART`, `TAPE`, and `DAT`. The `/etc/tpgstat` command displays the user reservation status for all users (to use this command, you must be `root` or a member of group `operator`). The `/bin/tprst` command displays the number and type of devices reserved by the current user, with no restrictions on the use of the command.

A `tpgstat` display shows all possible available tape types for each user, how many of each type that user has reserved, and for how long. In the following sample `tpgstat` display, the output shows that no user has any tapes reserved. For all tape types (device groups) that are currently configured up (through the `tpconfig` command), it will appear as if the `tpdaemon` has reserved that tape type:

```
sn5111# tpgstat
    user   job id      dgn w rsvd used mins NQSid
tpdaemon       22 TAPE          0    0  139
               22 CART          1    0  139
```

If user `jones` has used the `rsv` command to reserve a square (`CART`) tape drive, the `tpgstat` command shows each of the possible tape types that `jones` can reserve and that this user has reserved one `CART` tape drive:

```
sn5111#  tpgstat
    user   job id      dgn w rsvd used mins NQSid
tpdaemon       22 TAPE          0    0  142
               22 CART          1    0  142
jones          14 TAPE          0    0    1
               14 CART          1    0    1
```

The `tprst` command displays the status of the tapes reserved for just the current job ID.

2. If the tape is not already in a physically writable condition, 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 digital audio tapes (type `DAT`), 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.

3. Perform a full file system backup by using the `/etc/dump` command.

This step shows the actual command to perform a file system backup by using the `/etc/dump` command. In this example, a full (`-t 0`) backup of the file system `/dev/dsk/src` is performed to `CART` tape (`-g`). The specified volume serial number (`-v`) and label type (`-l`) are used, and the date and time of the beginning of the backup and the file system dumped are written to a log file called `/etc/dumpdates` by specifying the `-u` option.

The `dump` command and its output follow:

```
sn5111# /etc/dump -t 0 -u -g CART -v JON1 -l nl /dev/dsk/src
dump (/src to tape): Date of this level 0 dump: Mon Oct 11 20:23:39 1993
dump (/src to tape): Dumping /src
dump (/src to tape): to tape
dump (/src to tape): mapping (Pass I) [regular files]
dump (/src to tape): mapping (Pass II) [directories]
dump (/src to tape): estimated 23618 sectors on 0.00 tape(s).
dump (/src to tape): dumping (Pass III) [directories]
dump (/src to tape): dumping (Pass IV) [regular files]
dump (/src to tape): dump has completed, 23618 blocks
```

For non-MLS systems, if this is the first time that the `dump` command has been used on your system with the `-u` option, an `/etc/dumpdates` file may not exist. This causes the following error message at the end of the `dump` command screen output:

```
dump (/src to tape): dump has completed, 23618 blocks
dump (/src to tape): cannot open an existing /etc/dumpdates
file
dump (/src to tape): 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 is shown as follows:

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

Note: Because an interrupt will cause an abort, 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.

4. If no `-g` (tape type group name) option is supplied on the `/etc/dump` command to specify a particular kind of tape device to reserve, `dump` 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/dump` command also defaults to `DEF_DGN` tapes, you can specify other tape types by using the `-g` option.

5. The `dump` initiated tape mount request 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.

6. Respond to the next operator message by specifying the volume serial (VSN) number 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 rss000:: display truncated:
> /usr/lib/msg/rep 2 JON1
```

All messages related to your tape job have now disappeared.

7. 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#
```

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

9. Write down (preferably on the actual paper label on the tape itself) the contents of the tape, the VSN it was assigned by using the `-v` option of the `dump` command (in this case `JON1`), and the date the tape was created. It may be useful to add the `dump` command that was used to write the tape.

Your file system backup (dump) using `tpdaemon` is now complete.

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

```
sn5111# rsv CART
sn5111# tpmnt -l nl -r in -n -v JON1 -g CART -p /tmp/dumpfile
sn5111# /etc/dump -t 0 -u -f /tmp/dumpfile /dev/dsk/src
sn5111# rls -a
```

Procedure 23: Restoring a full 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 full file system restore is being performed. This means that the entire contents of the specified file system are read in from tape and that they overwrite the current disk version of that file system.

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.

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

```
snxxxx# /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.

Steps

The following are the steps for performing a **full file system restore**. This means that the entire contents of the specified file system are read in from tape and that they overwrite the current disk version of that file system:

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. 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 the rest of this example.

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

```
snxxxx# /etc/mount
/ on /dev/dsk/root read/write on Mon Oct 11 10:38:15 1993
/tmp on /dev/dsk/tmp read/write on Mon Oct 11 10:40:56 1993
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Mon Oct 11 10:40:59 1993
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Mon Oct 11 10:41:02 1993
/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; a file system should be idle before you unmount it. 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:

```
snxxxx# /etc/dump /dev/dsk/src
```

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

```
snxxxx# /etc/mount
/ on /dev/dsk/root read/write on Mon Oct 11 10:38:15 1993
/tmp on /dev/dsk/tmp read/write on Mon Oct 11 10:40:56 1993
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Mon Oct 11 10:40:59 1993
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Mon Oct 11 10:41:02 1993
```

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

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 that follows is optional syntax. For more about the `mkfs` command, see the `mkfs` man page.

   ```
   snxxxx# mkfs -q /dev/dsk/src
   ```

5. Check the file system by using the `/etc/fsck` command. You must perform this step before mounting the file system:

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

6. 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 in which you can perform the remaining

administrative tasks without users being affected or interfering. Traditionally, the mount point administrators use for such tasks is `/mnt`, because `/mnt` is not a directory users probably will 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:

```
snxxxx# /etc/mount
/ on /dev/dsk/root read/write on Tue Oct 12 19:09:25 1993
/tmp on /dev/dsk/tmp read/write on Tue Oct 12 19:10:01 1993
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Tue Oct 12 19:10:04 1993
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Tue Oct 12 19:10:07 1993
/usr/src on /dev/dsk/src read/write,rw,CRI_RC="YES" on Tue Oct 12 19:10:09 1993
```

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

7. 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 forcibly from a file system, 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:

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

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

```
snxxxx# cd /mnt
```

9. Restore the file system (in this case `/dev/dsk/src`).

There are two methods of doing file system restores. This step 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.

For example, you can invoke the interactive method at this point, as follows:

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

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

The `-r` option invokes a full file system restore. The `-g` option specifies that square (`CART`) tapes are being used in this restore example:

```
snxxxx# /etc/restore -r -V JON1 -l nl -g CART
```

To run the `restore` command as a background process, append an & symbol to the end of the command line.

> **Note:** If the `-r` option fails, the `-x` option of the `restore` command also is used for a full file system restore.

10. 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.

11. 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:

```
snxxxx# /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.

12. 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 rss000
:
: display truncated
:
> /usr/lib/msg/rep 2 JON1
```

    All messages related to your tape job have now disappeared.

13. 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
snxxxx#
```

14. Perform any applicable incremental restores, as necessary. In this particular example, this is not applicable.

    If your site regularly performs both full and incremental backups and you are performing a restore following a disk failure, this would be applicable. Incremental backups contain only those files that have changed since the last lower-level (more complete) backup. Thus, if your site had a full (level 0) backup and a level 9 incremental backup for the file system, you would first restore the full backup, followed by the incremental backup. The command line would look something like the following:

```
snxxxx# /etc/restore -r -V INC1 -l nl -g CART
```

15. Remove the restore symbol table from the file system following the last restore performed. This table is used to pass information between incremental restore passes, and it can grow to be of significant size.

    Example:

    ```
    snxxxx# rm /mnt/restoresymtabl
    ```

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

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

17. 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:

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

    The full 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:

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

Procedure 24: 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:

```
snxxxx# /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.

Steps

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:

```
snxxxx# 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:

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

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

Two examples follow. **Example 1** restores the `/src/uts/Nmakefile` file to the `/src` file system. **Example 2** 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.

Example 1 restores the `Nmakefile` file. 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`.

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

Example 2 restores the `/src/uts/c1/sys` directory, including all of its files and subdirectories and their contents:

```
snxxxx# /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:

```
snxxxx# /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
snxxxx#
```

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:

```
snxxxx# rsv CART
snxxxx # tpmnt -l nl -r in -n -v JON1 -g CART -p /tmp/dumpfile
snxxxx# /etc/restore -r -f /tmp/dumpfile
snxxxx# 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 187.

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*: `chgrp`(1), `chown`(1), `passwd`(1), `su`(1), and `udbsee`(1) man pages
- *UNICOS Administrator Commands Reference Manual*: `nu`(8), `udbgen`(8), and `udbpl`(8) man pages
- *UNICOS File Formats and Special Files Reference Manual*: `acid`(5), `cshrc`(5), `group`(5), `passwd`(5), `profile`(5), `shells`(5), and `udb`(5) man pages
- *General UNICOS System Administration*

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 window 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 window environment, you can use the `/etc/nu` utility (see Section 7.5, page 154).

  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 169). 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 145, 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 26, page 152).
- 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 27, page 153).

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

- Follow the procedures in Section 7.5, page 154, to make the new entry in `/etc/udb` by using the `/etc/nu -a` command. (Section 7.5, page 154, 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 169, to make the desired entry in `/etc/udb` by using the `/etc/udbgen` command. (Section 7.6, page 169, 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 143.

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 25: 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, page 154).

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, page 169).

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).
`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 26: 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.

For information on how to use `/etc/nu`, see Section 7.5, page 154. For information about how to use `/etc/udbgen`, see Section 7.6, page 169. To determine which utility will work best for you in a given situation, you might want to read both sections.

Procedure 27: 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.

For information on how to use `/etc/nu`, and Section 7.5, page 154. For information about how to use `/etc/udbgen`, see Section 7.6, page 169. To determine which utility will work best for you in a given situation, you might 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 28: 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 29: 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 28, page 154.)

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 30: 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 145.

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 access 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 (n) for Cray J90 Model V 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 31: 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 32: 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 33: 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 30, page 158. 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/udbgenudbgen_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 34, page 177).

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 34: 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 181.

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 35: 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 36: 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 37: 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 38: 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 39: 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*: `mail`(1), `news`(1), `su`(1), `wall`(1), and `write`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*: `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`(8) 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`(1) 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`(1) 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 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*.

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 209.

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*: `brc`(8), `cron`(8), `newsys`(8), `reduce`(8), `sar`(8), and `syslogd`(8) man pages
- *UNICOS User Commands Reference Manual*: `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 143).

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*.

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 Cray 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)

For more information on DMF, see the *Cray Data Migration Facility (DMF) Administrator's Guide.*

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`

Cray System 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 10.3, page 211, which describes features unique to CSA.

For information on using standard UNIX System V accounting, as well as CSA, see *UNICOS Resource Administration*.

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.

Your accounting configuration file is located in `/etc/config/acct_config`.

10.1 Related Accounting Documentation

The following publications contain more detailed information about the material covered in this section:

- *UNICOS Resource Administration*, "Accounting" chapter
- *UNICOS Administrator Commands Reference Manual*, including the following man pages: `acctdusg`(8), `chargefee`(8), `csaboots`(8), `csabuild`(8), `csacon`(8), `csacrep`(8), `csadrep`(8), `csaedit`(8), `csajrep`(8), `csaline`(8), `csanqs`(8), `csapacct`(8), `csaperiod`(8), `csarecy`(8), `csarun`(8), `csaverify`(8), `devacct`(8), `diskusg`(8), `dodisk`(8), `nulladm`(8), `runacct`(8), and `setacid`(8).
- *UNICOS User Commands Reference Manual*: `acctcom`(1), `ja`(1), `last`(1b), and `who`(1) man pages

10.2 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 will 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 data for only terminated sessions, unless you invoke the `csarun` command 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. For information about recycled data, see *UNICOS Resource Administration*.
Session	CSA organizes accounting data by sessions and boot times, and then it 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 one 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/boot period	A period delineated by the system boot times found in `/etc/csainfo`. The `csaboots` command writes to this file during system boot.

10.3 Unique Features of CSA

Like the UNIX System V accounting package, CSA provides methods to collect per-process data, record connect sessions, monitor disk usage, and charge fees. However, CSA also provides other facilities not available from the standard accounting package, including the following:

- Per-job accounting.
- Device accounting; categories include logical, block, and character special devices. Disk usage information is not available on a job basis; however, to bill disk usage on a user or account ID basis, you can use output from the `dodisk` command.

  Note: 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 is kept, the more file system space 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.

- Daemon accounting information (for NQS and the tape subsystem) is available from the NQS and online tape daemons. Data is written to the `nqacct` and `tpacct` files, respectively, in the `/usr/adm/acct/day` directory. The NQS and online tape daemons also must enable accounting.
- Device usage by account ID.
- Arbitrary accounting periods; for example, you can set your accounting period to be from 4 A.M. to 4 P.M. rather than using the default period.
- Flexible system billing unit (SBU) scheme.
- One file that contains all data from the accounting period.

- Archiving of accounting data, so you can move the data to a front-end system and merge it with your other accounting information.
- Capability to perform additional site-specific processing during daily accounting.
- Error recovery and automatic repairing of file systems.

For detailed information on these facilities, see *UNICOS Resource Administration*.

10.4 Accounting Directories and Files

This section provides a brief overview of the CSA file and directory structure. The following directories apply to both UNIX System V and CSA and are the main accounting directories. For a more complete description of the files and directories, see *UNICOS Resource Administration*.

> **Note:** Consider configuring the `/usr/adm` directory as another file system so that if `/usr/adm` fills up, other directories (such as `/usr/mail`) are still usable.
>
> The `/tmp` directory also is used while the `csarun` script is running. (For information about the `/tmp` directory, see Chapter 5, page 55.)

Directory	Description
`/usr/lib/acct`	Contains all of the programs and scripts used to run either CSA or UNIX System V accounting. (For a complete list of programs and scripts, see *UNICOS Resource Administration*.) The only CSA program not located here is `/etc/csaboots` (see the `csaboots`(8) man page), which records boot times at system startup. Programs used only by CSA begin with the characters `csa`. This directory also may contain the `csa.archive1`, `csa.archive2`, `csa.fef`, and `csa.user` user-exit scripts if you enable them. (To determine whether your UNICOS release level allows these scripts, see *UNICOS Resource Administration*.)

`/usr/adm/acct/day`

Contains the following current and recycled accounting files for per-process, disk, and daemon accounting:

- `dtmp` (disk accounting data)
- `nqacct*` (NQS daemon accounting data)
- `pacct*` (per-process accounting data)
- `tpacct*` (tape daemon accounting data)

Note: Accounting files in `/usr/adm/acct/day` that include the suffix 0 in their file names, 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 offline. For a description of how to do this, see the "Accounting" chapter in *UNICOS Resource Administration*. By default, all raw data files are deleted after use and are not archived.

`/usr/adm/acct/fiscal`

Contains periodic files created by `csaperiod` (CSA) or `monacct` (System V). Within this directory, the `rpt/MMDD/hhmm/rprt` file contains a variety of CSA periodic reports.

`/usr/adm/acct/nite`

Contains files that are reused daily by `csarun` (CSA) or the `runacct` (System V) procedure. Contains processing messages and errors (files that have names beginning with `E` and ending with the date and time).

`/usr/adm/acct/sum`

Contains cumulative summary files updated by `csarun` (CSA) or `runacct` (System V). Within this directory, the `rpt/MMDD/hhmm/rprt` file contains a variety of CSA daily reports.

`/usr/adm/acct/work`

Contains temporary files used by daily accounting procedures.

The following are the main basic accounting files:

File	Description
`/etc/config/acct_config`	Accounting configuration file contains the configurable parameters used by the accounting commands.
`/etc/csainfo`	Contains system boot time, written to this file by `/etc/csaboots`.
`/etc/wtmp`	Contains log-in records and log-out records for users. Records `tty`, process ID, type of process, and connect-time accounting data.
`/usr/adm/acct/day/pacct`	Contains data file written by the UNICOS kernel. Source of all process accounting for both CSA and System V accounting.
`/usr/adm/acct/nite/statefile`	Contains the name of the next reentrant state so that the `csarun` accounting script can be restarted at a specified point. For descriptions of CSA states (files that have names beginning with `E` and ending with the date and time contain errors), see Section 10.7, page 220.

CSA outputs the following six data files:

File	Description
`/tmp/AC.`*MMDD*/*hhmm*`/Super-record`	Session record file; this file usually is deleted after CSA has used it.
`/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; if you specify the `-r` option, `csaperiod` deletes this file.
`/usr/adm/acct/sum/data/`*MMDD*/*hhmm*`/cms`	Daily command usage data; if you specify the `-r` option, `csaperiod` deletes this file.
`/usr/adm/acct/sum/data/`*MMDD*/*hhmm*`/dacct`	Daily disk usage data; if you specify the `-r` option, `csaperiod` deletes this file.

Note: 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 to mount this file system along with each different system. You also must copy two other files, `/etc/csainfo` and `/etc/wtmp`, from the previously booted `/` file system. You must copy these files to the new `/` file system before it is brought up. If you do not copy `/etc/csainfo` file to the new `/` file system correctly, `csarun` may abort abnormally. When `/etc/wtmp` is not copied, incorrect connect-time data is reported.

10.5 Daily Operation Overview of CSA

When the UNICOS system is run in multiuser mode, accounting behaves as described in the following steps and shown in Figure 4, page 218. However, if you modify CSA to meet your own requirements, the following steps may not reflect the process at your site:

1. `/etc/csaboots`

   Writes boot record to `/etc/csainfo`, which contains a record of every system boot; executed by `/etc/rc` on entering multiuser state.

2. `/usr/lib/acct/startup`

   a. Executed by `/etc/rc` on entering multiuser state (see `acctsh`(8) for additional information).

   b. `acctwtmp` adds a boot record to `/etc/wtmp`.

   c. `turnacct` starts per-process accounting.

   d. `turnacct` enables daemon accounting if it is enabled in the `acct_config` file. By default, `/usr/lib/acct/startup` enables daemon accounting.

   e. `remove` cleans up previous day's files.

3. When they are started by `/etc/rc`, the NQS and tape daemons enable daemon accounting.

4. `/usr/lib/acct/ckpacct`

   a. Executed by `cron` every hour to check size of `/usr/adm/acct/day/pacct`. If `pacct` gets too large, a new file is started. The new file(s) will have a `.x` suffix; `.x` is .1, .2, .3, and so on.

   b. Verifies that at least 500 free data blocks exist in the file system that contains the `/usr/adm/acct` directory; if the file system is full, `ckpacct` will turn off accounting.

5. `/usr/lib/acct/ckdacct`

   a. Executed by `cron` every hour to check size of daemon accounting files. If an accounting file gets too large, a new file is started.

   b. Verifies that at least 500 free data blocks exist in the file system that contains the `/usr/adm/acct` directory. `ckdacct` turns off daemon accounting if the file system is full.

6. The `cron` utility runs the `dodisk` script periodically to generate a snapshot of the amount of disk space being used by each user.

7. `/usr/lib/acct/csarun` (also see Section 10.6, page 219)

   a. Executed by `cron` at specified times.

   b. Processes active accounting files, combining data from `pacct`, `/etc/wtmp`, `nqacct`, and `tpacct`.

   c. Produces accounting reports and a consolidated data file.

8. `/usr/lib/acct/shutacct`

   a. Executed by `/etc/shutdown`.

   b. `acctwtmp` writes shutdown reason in `/etc/wtmp`.

   c. `turnacct` stops per-process accounting.

   d. `turndacct` stops daemon accounting.

9. (Optional) `/usr/lib/acct/chargefee`

   a. Creates a fee file; a site must invoke this (see the `chargefee`(8) man page).

   b. (Optional) `/usr/lib/acct/csaperiod`

      i. Runs periodic accounting and is executed by `cron` to process consolidated accounting data from previous accounting periods.

      ii. Produces a consolidated periodic accounting file and an ASCII report.

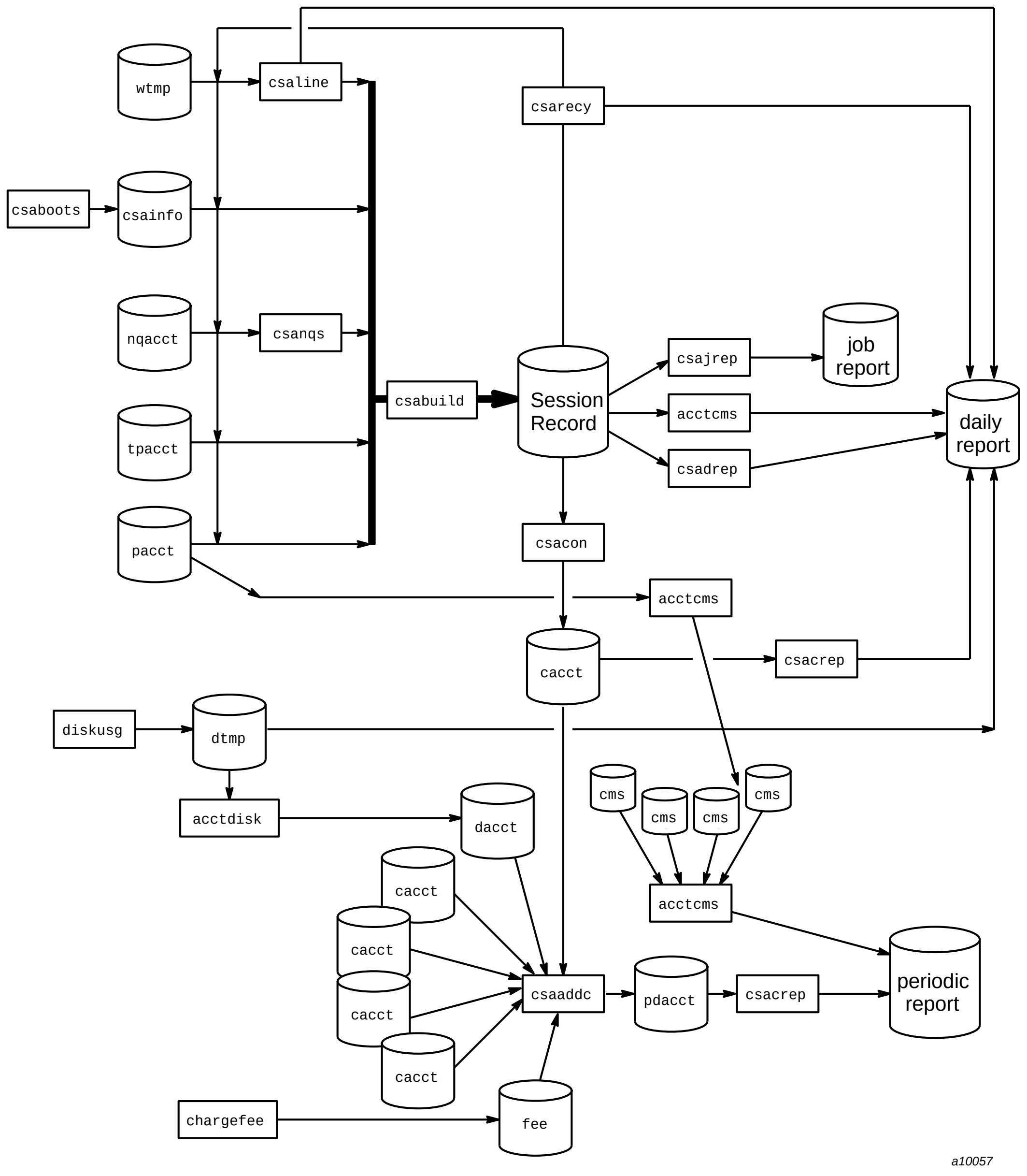


Figure 4. Daily Operation Overview of CSA

10.6 The csarun Command

The /usr/lib/acct/csarun command (see the csarun(8) man page) is the primary daily accounting shell script. It processes connect, per-process, and daemon accounting files and usually is initiated by cron during nonprime hours. csarun also contains four user-exit points, allowing you to tailor the daily run of accounting to your specific needs (for information on setting up user exits callable from csarun and callable from runacct, see *UNICOS Resource Administration*).

If errors occur, csarun does not damage files. It contains a series of protection mechanisms that try to recognize an error, provide intelligent diagnostics, and terminate processing in such a way that csarun can be restarted with minimal intervention.

You also can interactively process data for a new accounting period by executing the following command. Before executing csarun in this manner, ensure that the previous invocation completed successfully by looking at the active and statefile files in /usr/adm/acct/nite . Both files should specify that the last invocation completed successfully.

```
nohup csarun 2> /usr/adm/acct/nite/fd2log &
```

Note: All command lines you enter that include I/O redirection, such as 2>, require that you use either the ksh or sh shell; the csh shell will not accept the same I/O redirection command syntax as the ksh and sh shells.

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 active file. Diagnostic output during the execution of csarun is written to fd2log. The lock and lock1 files prevent concurrent invocations of csarun; csarun aborts 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 that begin with E and end with the current date and time. For example, Ebld.11121400 is an error file from csabuild for a csarun invocation on November 12, at 14:00.

If csarun detects an error, it sends an informational message to the operator by using 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 `/etc/config/acct_config` configuration file.

10.7 CSA Accounting States

During daily execution, `csarun` writes its starting time into `nite/clastdate`. The main `csarun` processing is divided into several separate, restartable states (see the following list). At the conclusion of each state, `csarun` writes the name of the next state into `nite/statefile`. The `csarun` procedure also writes descriptive messages into `nite/active` and any diagnostic messages into `nite/fd2log`.

If daily accounting does not complete successfully, check the `active`, `fd2log`, and `statefile` files. You may then restart `csarun` from the current state, or you may specify the state at which to restart.

Example:

`csarun 0415`	Restarts accounting for April 15, using the time and state specified in `clastdate` and `statefile`.
`csarun 0415 0400 CMS`	Restarts at the specified time and at the CMS state.

If `csarun` is run without arguments, the previous invocation must have terminated normally. If not, `csarun` will abort.

The following is a list of CSA accounting states in the order in which they occur:

State	Description
`COMPLETE`	Ensures that accounting successfully completed the last time it was run.
`SETUP`	The current accounting files are switched by using the `turnacct` and `turndacct` files. These files are then moved to the `/usr/adm/acct/work/`*MMDD*/*hhmm* directory. File names are prefaced with `W`. The `/etc/wtmp` and `/etc/csainfo` files also are moved to this directory.
`WTMPFIX`	The `wtmpfix`(8) command checks the `wtmp` file in the work directory for accuracy. Some date changes cause `csaline`(8) to fail; therefore, `wtmpfix` tries to adjust the time stamps in the `wtmp` file if a data change record appears.

	If `wtmpfix` cannot fix the `wtmp` file, you must repair the `wtmp` file manually, as described in the section “Verifying and Correcting Data Files” of the *UNICOS Resource Administration*.
`VERIFY`	By default, per-process, NQS, and tape accounting files are checked for valid data. Records with data that is not valid are removed. Names of bad data files are prefixed with `BAD.` in the `/usr/adm/acct/work/*` directory.
`PREPROC`	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. You can use this script to archive the raw and preprocessed accounting files. The shell `.` command executes the `/usr/lib/acct/csa.archive1` script, if it exists. By default, this feature is disabled.
`BUILD`	The per-process, NQS, tape, and connect accounting data is organized into a session record file.
`ARCHIVE2`	Second user exit of the `csarun` script. You can use this script to archive the session record file. The shell `.` command executes the `/usr/lib/acct/csa.archive2` script, if it exists. By default, this feature is disabled.
`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`.
`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`, also is produced from the session record file. Accounting data for unfinished sessions also 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. You can use this script to execute a front-end formatter. The shell `.` command executes the script `/usr/lib/acct/csa.fef`, if it exists.

`USEREXIT`	Fourth user exit of the `csarun` script. You can use this script to execute local accounting programs. The shell `.` command executes the `/usr/lib/acct/csa.user` script, if it exists.
`CLEANUP`	Cleans up temporary files, removes the locks, and then exits.
`COMPLETE`	Ensures that accounting successfully completed.

10.8 Data Recycling

A system administrator must correctly maintain recycled data to ensure accurate accounting reports. Data recycling allows CSA to bill sessions properly that are active during multiple accounting periods. By default, the `csarun` script 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` flags each session as being either active or terminated. `csarecy` reads the `Super-record` file and recycles data for the active sessions. `csacon` consolidates the data for the terminated sessions, which `csaperiod` uses later. `csarun` invokes `csabuild`, `csarecy`, and `csacon`.

The `csarun` command 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`.

Usually, an administrator should not have to purge the recycled accounting data manually. This purge should be necessary only if accounting data is missing. Missing data can cause sessions to recycle forever and consume valuable CPU cycles and disk space.

Recycling unnecessary data can consume a lot of disk space and CPU time. The session file and recycled data can occupy a vast amount of disk space on the file systems that contain `/tmp` and `/usr/adm/acct/day`. Sites that archive data also require additional offline media. `csarun` uses wasted CPU cycles to reexamine and recycle the data. Therefore, to conserve disk space and CPU cycles, you should purge unnecessary recycled data from the accounting system.

For detailed information about data recycling, see *UNICOS Resource Administration*.

Procedure 40: Setting up CSA

This procedure shows you how to ensure that accounting is started and terminated properly at system boot and shutdown time. The procedure also shows you how to configure accounting parameters, gather disk usage information, and schedule daily and periodic accounting runs automatically.

Note: Before you begin this procedure, determine your company's system billing units (SBUs). For detailed information about making site-specific modifications, see the "Tailoring CSA" section of the "Accounting" chapter in *UNICOS Resource Administration*.

1. Modify configurable accounting parameters manually or by using the menu system. Following are some of the types of changes you might want to make:

   - Setting up SBUs for per-process accounting (`pacct`) data; by default, no SBUs are calculated. You can set several variables (for example, weighting factors are set through over a dozen variables, including `P_BASIC`, `P_TIME`, and `P_STIME`). (For information, see *UNICOS Resource Administration*.)
   - Charging for NQS jobs (`NQS_TERM_`*xxx* variables); also see step 4.
   - Modifying the file system on which `/usr/adm/acct` resides; the default is `/usr` (`ACCT_FS` variable).
   - Working with an alternative accounting configuration file (the `ACCTCONFIG` variable).
   - Compiling a list of users to whom mail is sent when a warning or error is detected. The default is `root` (`/`) and `adm` (`WMAIL_LIST` and `MAIL_LIST` variables).
   - Enabling NQS and tape daemon accounting at system startup (`NQS_START` and `TAPE_START` variables).
   - Changing the minimum number of free blocks (the `MIN_BLKS` variable) needed in `ACCT_FS` to enable accounting or to run `csarun` or `csaperiod`. The default is 500 free blocks.

   If you are using the menu system, select the `Configure System->Accounting Configuration` menu. Enter your changes, and then activate the new configuration. A sample menu screen follows:

```
Configure System
    ->Accounting Configuration
```

```
                         Accounting Configuration



S-> Edit accounting configuration ...
    Import accounting configuration ...
    Activate accounting configuration ...



    Keys:   ^? Commands   H Help   Q Quit   V ViewDoc   W WhereAmI
```

If you are not using the menu system, edit the `/etc/config/acct_config` file.

2. Accounting is started by default each time `/etc/rc` is invoked. Make sure that `csaboots` is invoked from `rc`, and not from `/etc/inittab` or `/etc/brc`.

3. (Optional) Enable daemon accounting at system start-up time, as follows:

   a. Ensure that the variables for the subsystems for which you want to enable daemon accounting are set to `on` in `/etc/config/acct_config` by editing the `/etc/config/acct_config` file or by using the `NQS Configuration` menu. Set the `NQS_START` and `TAPE_START` parameters to `on` to enable NQS and online tapes, respectively.

   b. If necessary, enable accounting from the daemon's side (required for NQS and tape).

      To turn on NQS accounting, do **one** of the following actions:

      • Insert the line `set accounting on` in the `/etc/config/nqs_config` and `/etc/config/NQS.startup` file (recommended).

        or

      • Turn on NQS accounting by using the `qgmr set accounting on` command.

To turn on tape accounting, execute the `/usr/lib/tp/tpdaemon` by using the `-c` command-line option from `/etc/config/daemons` or from the `System Daemons Table` menu.

4. (Optional) If you plan to gather disk usage statistics, create or modify the `/etc/checklist` file. This file contains a list of file systems (full path names) for which `dodisk` will collect information. One special file name is listed on each line. A sample `/etc/checklist` file follows:

```
# more /etc/checklist
       /dev/dsk/home
       /dev/dsk/tmp
       /dev/dsk/filesystemA
       /dev/dsk/filesystemB
```

   Generally, `root` executes `dodisk` through `cron` (see the next step). `csarun` incorporates the disk usage data into the daily accounting report.

5. Ensure that entries similar to the following are included in `/usr/spool/cron/crontabs/root` so that `cron` automatically runs daily accounting. The `ckdacct` and `ckpacct` scripts check and limit daemon and standard accounting files sizes.

   Caution: The `dodisk` script **must** run at least 1 hour before the `csarun` script, so that the `dodisk` script has time to complete before `csarun` tries to access that data. You also **must** invoke `dodisk` with either the `-a` or `-A` option; if you do not, `csaperiod` aborts when it tries to merge the disk usage information with other accounting data.

```
0 23 * * 0-6 /usr/lib/acct/dodisk -a -v 2> /usr/adm/acct/nite/dk2log
0 0 * * 0-6 /usr/lib/acct/csarun 2> /usr/adm/acct/nite/fd2log
0 * * * * /usr/lib/acct/ckdacct nqs tape
0 * * * * /usr/lib/acct/ckpacct
```

   If you want to run periodic accounting, ensure that an entry similar to the following is included 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
```

6. Ensure that the following lines are included in `/usr/lib/acct/csarun` if you want the following information:

a. To get trace information, ensure that the following line is the first line after the first set of comment lines in the file:

```
set-xS
```

b. To get the SBU report , add the `b` option to the `csacrep` line, as follows:

```
csacrep -hucwb < ${CDATA}/cacct > ${CRPT}/conrpt 2> ${NITE}/Ecrpt.${DTIME}
```

7. Update the `/usr/lib/acct/holidays` file, which should reflect your site's prime/nonprime time and holiday schedules. The year field must contain either the current year or the wildcard character symbol, `*`, which specifies that the current year should be used. Following is a sample `holidays` file:

```
# USMID @(#)acct/src/acct/holidays
#
#       (c) Copyright Cray Inc.
#       Unpublished Proprietary Information.
#       All Rights Reserved.
#
# Prime/Nonprime Table for UNICOS Accounting System
#
# Curr  Prime   Non-Prime
# Year  Start   Start
#
  199x   0730    1730
```

(Accounting software references this line to confirm current year)

```
#
# Day of        Calendar        Company
# Year          Date            Holiday
#
    1           Jan 1           New Year's Day
  146           May 25          Memorial Day
  184           Jul 2           Independence Day Thursday
  185           Jul 3           Independence Day Friday
  186           Jul 4           Independence Day
  251           Sep 7           Labor Day
  331           Nov 26          Thanksgiving Day
  332           Nov 27          Thanksgiving Friday
  359           Dec 24          Christmas Eve
  360           Dec 25          Christmas Day
  366           Dec 31          New Year's Eve day
```

8. Label file systems with accounting types while they are mounted by using the `devacct`(8) command. If a file system does not contain a device type label, device accounting ignores it.

10.9 Daily CSA Reports

The `csarun` script generates various daily reports, all of which are placed in a file named `/usr/adm/acct/sum/rpt/`*MMDD*`/`*hhmm*`/rprt` (for example, `0415/2000/rprt`). By default, the report includes statistics only for sessions that have terminated. The reports include the following:

- Interactive connect time by ttyp.

- CPU usage by user ID and account ID. For a description of the fields in this report, see the "Accounting" chapter of *UNICOS Resource Administration*.
- A listing of active interactive and batch jobs by job ID.
- Disk usage by user ID and account ID.
- Command summary data by total CPU time used. For a description of the fields in this report, see the "Accounting" chapter of *UNICOS Resource Administration*.
- Last interactive login information by date.
- Job mix (interactive versus NQS), tape, and NQS usage.

Many of the periodic reports that `csaperiod` generates are similar to the preceding reports; however, not all of the reports listed have a periodic equivalent. Periodic reports are located in `/usr/adm/acct/fiscal/rpt/`*MMDD*`/`*hhmm*`/rprt`.

Your system `config` file is `/etc/config/acct_config`.

Adding Your Cray System to Your Network [11]

This section describes how to place your Cray J90se or Cray SV1 series GigaRing based system or your Cray J90, Cray J90se, or Cray SV1 series Model V based 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 system on an existing TCP/IP network by using the information in this section, see the *UNICOS Networking Facilities Administrator's Guide*, 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*
- *UNICOS Networking Facilities Administrator's Guide*
- *UNICOS Administrator Commands Reference Manual* (man pages):

`arp`(8)	`inetd`(8)	`rexecd`(8)
`atmarp`(8)	`initif`(8)	`rlogind`(8)
`enstat`(8)	`mkbinhost`(8)	`route`(8)
`fingerd`(8)	`netstart`(8)	`rshd`(8)
`ftpd`(8)	`ntalkd`(8)	`sdaemon`(8)
`gated`(8)	`ping`(8)	`tcpstart`(8)
`ifconfig`(8)		
`traceroute`(8)		

- *UNICOS User Commands Reference Manual* (man pages):

  `ftp`(1b)

  `hostname`(1)

  `netstat`(1b)

- *UNICOS File Formats and Special Files Reference Manual* (man pages):

  `gated-config`(5)

  `hosts`(5)

  `hosts.equiv`(5)

  `inetd.conf`(5)

  `lo`(4)

  `networks`(5)

  `protocols`(5)

  `rhosts`(5)

  `services`(5)

- The following publications also are recommended, but Cray 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`.

11.2 Adding a Cray System to an Existing TCP/IP Network — GigaRing Based System

If you have **GigaRing** based Cray system to place in the TCP/IP network, see Procedure 41, page 230 below. If you have a **Model V** based system to place in the TCP/IP network, see Procedure 44, page 237.

Procedure 41: Adding a Cray J90se or Cray SV1 series GigaRing based system to an existing TCP/IP network

To place a system on an existing TCP/IP network, you should complete the following steps:

Note: To add your system to an existing TCP/IP network, you must have several configuration files. The easiest way to create these files is to configure the system to talk to another host on the network, copy the necessary files from that machine to your 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 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 GigaRing based system to your existing TCP/IP network.

1. Verify that the network section of the `/opt/CYRIos/sn`*xxxx*`/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 system, and an entry for at least one other host on the same network as the 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 GigaRing based system)


   456.789.16.125  cyclone cyclone-eth1
   (other host)
   ```

3. Compile a binary hosts file.

   Cray 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 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*. 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 system. To reach hosts that are not on the same FDDI or Ethernet network as the your 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 your 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 which 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 are also 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 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 your 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 or Cray SV1 series system should now be on the network. Check for entries for this system in the `/etc/hosts` file. You may want to also transfer an `/etc/networks` file to your system.

11.2.1 TCP/IP Path between the GigaRing Based System and SWS

A TCP/IP path between your Cray J90se or Cray SV1 series GigaRing based 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 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 Cray J90se system Ethernet interface and an SWS quad-board plug. The path may also be more complex;

this TCP/IP traffic could use an 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 mainframe while still in single-user mode, use one of the following procedures. (This will allow files to be transferred between the mainframe and the SWS.)

Procedure 42: Start TCP/IP from mainframe for GigaRing based systems

- For direct Ethernet connection between the Cray J90se or Cray SV1 series GigaRing based mainframe and the SWS:

  ```
  # /etc/ifconfig gether0 CRAY_ETHERNET_IP_addr netmask 0xffffff00
  # /etc/inetd
  ```

- When packets are routed from the 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 snxxxx
```

11.2.2 Changing the SWS Host Name and IP Address — GigaRing based systems

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 43: 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 your mainframe.

11.2.3 Changing the Cray J90se or Cray SV1 series GigaRing Based Host Name and IP Address

Section 11.2.2, page 236, described how to change the SWS host name and IP address.

Use the following procedure to change the Cray J90se or Cray SV1 series mainframe host name and IP address:

- On the UNICOS system, edit the `/etc/hosts` file.
- On the SWS, edit the `/etc/hosts` file.

***** This ends the GigaRing based system procedure. *****

11.3 Adding a Cray System to an Existing TCP/IP Network — Model V Based System

If you have **Model V** based Cray system to place in the TCP/IP network, see Procedure 44, page 237 below. If you have **GigaRing** based Cray system to place in the TCP/IP network, see Procedure 41, page 230.

Procedure 44: Adding a Cray J90, Cray J90se, or Cray SV1 series Model V based system to an existing TCP/IP network

To place a system on an existing TCP/IP network, you should complete the following steps:

Note: To add your system to an existing TCP/IP network, you must have several configuration files. The easiest way to create these files is to configure the system to talk to another host on the network, copy the necessary files from that machine to your 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 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 system to your existing TCP/IP network.

1. Verify that the network section of the `/sys/param` file on the console 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 system, and an entry for at least one other host on the same network as the system. The entry format is as follows:

   IPaddress host_name annotations

3. Compile a binary hosts file.

   Cray 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 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*. The entry format is as follows:

   interface_name family address ifconfig parameters

   Example:

```
# cat /etc/config/interfaces

...some comment lines omitted...

lo0   -    inet   localhost   -
en0   -    inet   cray.001    - netmask 0xffffff00
fddi0 -    inet   cray.fddi   - netmask 0xffffff00
atm0  -    inet   cray.atm    - netmask 0xffffff00
```

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 en0 fddi0 atm0
```

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 system. To reach hosts that are not on the same FDDI or Ethernet network as the your 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 your 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 which 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 are also 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
en0*     1496  cray-net         cray           0         0       2        0
fddi0    4352  crau-fddi-net    cray-fddi      249466    0       57636    0
fddi1*   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 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 your 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 J90, Cray J90se, or Cray SV1 series Model V based system should now be on the network. Check for entries for this system in the `/etc/hosts` file. You may want to also transfer an `/etc/networks` file to your system.

***** This ends the Model V based system procedure. *****

11.4 Backing up All Changes

Remember, back up any changes you have made to files on the SWS.

11.5 Verifying the UNICOS Configuration File

The UNICOS configuration file controls kernel configuration parameters. Because your 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 for **GigaRing** based systems is in the`/opt/CYRIos/sn`*xxxx*`/param` file.

The default configuration file for **Model V** based system is in the `/sys/param` file.

> **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*.

11.6 Rebooting in Multiuser Mode — GigaRing Based Systems

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 45: Rebooting in multiuser mode

1. Ensure that the your 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 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.

11.7 Domain Name Service (DNS)

If you want to use domain name service (DNS) to perform host name lookup, you should configure your Cray J90se or Cray SV1 series 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 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 46: Configuring a caching-only server by using the menu system

Note: If you have not completed Procedure 41, page 230, "Adding a Cray J90se or Cray SV1 series GigaRing based system to an existing TCP/IP network" or Procedure 44, page 237, “Adding a Cray J90, Cray J90se, or Cray SV1 series Model V based system to an existing TCP/IP network”, you should complete steps 3, 4, and 5 of the appropriate procedure before configuring a caching-only server.

For example, to configure your Cray J90se GigaRing based system or Cray J90se Model V based 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 47: Configuring a caching-only server without using the menu system

Note: If you have not completed Procedure 41, page 230, "Adding a Cray J90se or Cray SV1 series GigaRing based system to an existing TCP/IP network" or Procedure 44, page 237, “Adding a Cray J90, Cray J90se, or Cray SV1 series Model V based system to an existing TCP/IP network”, you should complete steps 3, 4, and 5 of the appropriate procedure before configuring a caching-only server.

To configure your GigaRing based system or Model V based 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
```

11.8 Common TCP/IP Configuration Files

After you have the Cray 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 3 describes some of the most common TCP/IP configuration files. The *UNICOS Networking Facilities Administrator's Guide,* describes all of these files.

Table 3. 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

File (relative to `/etc`)	Description	Change
`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

Network Information Service (NIS) Configuration [12]

12.1 Related NIS Documentation

The following documentation contains information covered in this section:

- *UNICOS Networking Facilities Administrator's Guide*
- *UNICOS Administrator Commands Reference Manual*: `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*: `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 also supports 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.*

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 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 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*.

An important distinction to make is that a Cray system running UNICOS can **serve** other NIS maps for its domain; however, the Cray 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*.

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.

Caution: *Clients* do not contain their own copies of the NIS maps. Instead, they request information from the servers in their domain.

You should configure your system as an NIS slave server.

This section contains procedures for the following:

- Using the menu system to configure your system as an NIS slave server
- Configuring your 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 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*, to familiarize yourself with UNICOS NIS before you configure NIS on your system.

Procedure 48: Using the menu system to configure your Cray system as an NIS slave server

This procedure pertains to configuring Cray J90se or Cray SV1 series GigaRing based system and Cray J90, Cray J90se, or Cray SV1 series Model V based 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 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*, and the `rpcbind`(8) and `rpcbstart`(8) man pages.

Before you configure NIS on your system, read the cautionary note and other important information in Section 12.2, page 251.

The following steps use the menu system to configure your Cray 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 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 your 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 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 yoursystemhostname
```

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 system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

5. Verify that the 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
yoursystem
```

Procedure 49: Configuring your Cray system as an NIS slave server without using the menu system

This procedure pertains to configuring Cray J90se or Cray SV1 series GigaRing based system and Cray J90, Cray J90se, or Cray SV1 series Model V based 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 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 system, read the cautionary note and other important information in Section 12.2, page 251.

The following steps explain how to configure your 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 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 yoursystemhhostname
```

   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 your 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
yourCRAYsystem
```

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 your system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

Procedure 50: Configuring user accounts to use NIS

Note: This procedure assumes that your 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 143). The `yppasswd` command works only if you have started the `yppasswdd` daemon on the NIS master server machine.

Network File System (NFS) Configuration [13]

13.1 Related NFS Documentation

The following documentation contains information covered in this section:

- *UNICOS Networking Facilities Administrator's Guide*
- *UNICOS Administrator Commands Reference Manual*: `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*: `exports`(5) and `fstab`(5) man pages

13.2 What Is NFS?

The network file system (NFS) is a 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 262). 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 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*:

- 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 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 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 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 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.*

Procedure 51: Configuring your Cray system as an NFS client

This procedure pertains to configuring Cray J90se or Cray SV1 series GigaRing based system and Cray J90, Cray J90se, or Cray SV1 series Model V based 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 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*:

```
# 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*:

```
# /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 52: Configuring your Cray system as an NFS server

The following steps explain how to configure a Cray J90se or Cray SV1 series GigaRing based system or a Cray J90, Cray J90se, or Cray SV1 series Model V based 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 system; it is intended for use only between Cray 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
```

Multistreaming Processors (MSPs) — GigaRing Based Systems [14]

A Cray SV1 series GigaRing based system node can be configured with a mix of CPUs and multistreaming processors (MSPs). An MSP is made up of four CPUs. When configured, an MSP can simultaneously process array data divided among the four CPUs.

There are two ways to use MSPs for applications:

- **Multistreaming:** a multistreaming application must be created using compiler options and directives (`-O streams` compiler command line option).
- **Autotasking:** an autotasking job is run on one or more MSPs through use of the `/etc/cpu` command with the `-a` option.

14.1 Configuring MSPs

Zero through six MSPs are dynamically created and dissolved by the UNICOS kernel when required by MSP jobs. The maximum number of MSPs is limited by the number of available CPUs on the node:

- A node must always have a minimum of four CPUs available for non-MSP processing.
- Each MSP requires four CPUs.

Thus, a node with eight CPUs can have a maximum of only one configured MSP: four CPUs available for single-streaming processing and four CPUs coupled together to form an MSP. The following table illustrates the physical limitations for MSP configuration based on the number of CPUs in a node:

Number of CPUs	Maximum Number of MSPs
4	0
8	1

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

Because an MSP simultaneously uses all four of its component CPUs and can execute only MSP processes, there must always be enough dedicated CPUs remaining in the node to perform single-streaming processes. In general, balance the maximum allowed number of MSPs so that there are enough MSP resources to accommodate MSP processes while reserving enough CPUs for non-MSP processes.

14.1.1 Configuring MSPs at System Startup

To specify the maximum number of MSPs at system startup, include the `msps` statement in the mainframe section of the `/etc/config/param` file. The statement defines the maximum number of MSPs allowed on the system node:

value `msps;`

The *value* specifies the maximum number of MSPs allowed. For detailed information on the mainframe section of the CSL parameter file. See the *UNICOS Configuration Administrator's Guide.*

14.1.2 Configuring MSPs using the /etc/cpu Command

To change the maximum number of MSPs while the system is running, use the `/etc/cpu` command with the `-M` option. For example, to allow configuring a maximum of 2 MSPs, use the following:

```
cpu -M 2
```

For detailed information on the `cpu` command, see the `cpu`(8) man page.

14.2 Using MSP Resources

Only multistreaming or autotasking applications can run on a MSP. The UNICOS operating system recognizes an executable as a multistreaming application by a flag set in the header of the `a.out` file The `/etc/cpu` command may be used to run an autotasking application on one or more MSPs by using the `-a` option. The compiler generates code to use the four CPUs that make up the MSP to improve the performance of the application.

MSPs are dynamically created on an as-needed basis limited only by the configured maximum number of MSPs. Only MSP applications compete for MSP resources. When required, the UNICOS kernel chooses which physical CPUs make up the MSP. The CPUs are selected for minimum mutual interference and maximum memory bandwidth. Once an MSP is created, the physical CPUs remain members of the MSP until the MSP is no longer needed and then dissolved.

14.3 Batch Scheduling MSP Applications

MSP and CPU applications share all node resources except the processors. An administrator must establish a policy to best schedule MSP and CPU applications, based on the following guidelines:

- Limit the number of scheduled MSP applications competing for MSP resources in proportion to the maximum number of configured MSPs.
- Initiate only enough MSP work to keep the processors busy and to fill in gaps created by I/O wait time, but not so many that the elapsed time of the applications grows significantly.

The batch service provider must know the number of MSPs available on a node and be able to schedule MSP work according to the established policy of MSP oversubscription. In addition, the batch service provider must have a way for the user to identify MSP requests when they are queued so they can be scheduled and dispatched correctly.

14.4 Viewing MSP Information

UNICOS commands and displays that identify MSPs use the following convention:

MSP*n.i*

The value of *n* specifies the MSP ordinal that uniquely identifies the MSP (a number from 0 to 5) and the value of *i* specifies the logical ID of the CPU within the MSP (a number from 0 to 3). Each MSP consists of a master CPU and three slave CPUs. The master CPU is always 0; the slaves are numbered from 1 to 3.

For example, the first MSP running in the system is numbered 0, and the master CPU of MSP 0 is numbered 0; the slave CPUs of the MSP are numbered 1 to 3:

```
MSP0.0
MSP0.1
MSP0.2
MSP0.3
```

The second MSP running in the system is numbered 1, and the master CPU of MSP 0 is numbered 0; the slave CPUs of the MSP are numbered 1 to 3:

```
MSP1.0
MSP1.1
MSP1.2
MSP1.3
```

14.4.1 Viewing MSP Configuration Information

The following UNICOS commands provide MSP configuration information:

Command	Description
`cpu`(8)	The `-i` option of the `cpu`(8) command displays information for each configured MSP.
`sysconf`(1)	The `NMSP` field of the `sysconf`(1) hardware display shows the maximum number of configurable MSPs The `NSSP` field of the `sysconf`(1) hardware display shows the number of CPUs that would remain if the configurable MSPs were fully in use.
`sysconf`(2)	The `_SC_CRAY_NMSP` call option of the `sysconf`(2) system call returns the maximum number of configurable MSPs.

14.4.2 Viewing MSP Job Connect Information

The `ps`(1) and `jstat`(1) commands present CPU connect information. On systems configured with MSPs, the commands identify the MSP(s) on which the job is running, as well as the mapping of the individual task member threads to the CPUs within the MSP(s). See the `ps`(1) and `jstat`(1) man pages for detailed information on viewing MSP job connect information.

14.4.3 Viewing MSP Hardware Performance Information

Hardware performance information is generated for an MSP's component CPUs by the `hpm`(1) and `hpmall`(8) commands. See the `hpm`(1) and `hpmall`(8) man pages for information on CPU performance data.

14.4.4 Viewing MSP Hardware Accounting Information

The `ja`(1) command provides job and session accounting information. The information is gathered from the job accounting file written by the kernel when job accounting is enabled. (Job accounting is enabled and disabled by embedding `ja` commands in a user command stream.)

The `-c` option of the `ja`(1) command displays information relating to the number of CPUs connected to a multitasking/multstreaming process; the `-s` option of the `ja`(1) command produces a summary report that contains a multitasking / multistreaming breakdown of CPU usage and connect information. See the `ja`(1) man page for information on the job accounting displays.

Menu System Overview [A]

This appendix provides a brief description of the UNICOS Installation and Configuration Menu System (also referred to as the *menu system* or *ICMS*), 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 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 product release.

For a more details about the use of the UNICOS menu system, see *UNICOS System Configuration Using ICMS*.

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

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 and Cray SV1 series systems use a different installation menu system than other Cray systems. To use the UNICOS installation menu system's configuration window, you should first quit the 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, `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.* 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*.

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 is also 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
- 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
    ->Utilities
        ->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
```

Frequently Used Commands [B]

This appendix provides a brief description of several frequently used system administration commands, scripts, and files. There are two sections:

- IOS-V console (Model V based systems only)
- UNICOS console (GigaRing based systems and Model V based systems)

For more details about these commands, see the online man page for the command or consult one of the following man page manuals:

- *UNICOS User Commands Reference Manual*
- *UNICOS System Calls Reference Manual*
- *UNICOS File Formats and Special Files Reference Manual*
- *UNICOS Administrator Commands Reference Manual*
- *CRAY IOS-V Commands Reference Manual*

B.1 Commands Available from the IOS Console — Model V Based Systems

The following commands, scripts, and files are available from the IOS console:

Command/Script/File	When to use
`/adm/syslog`	ASCII file that contains a log of IOS-generated system status messages.
`/bin/boot`	A script that contains IOS commands to execute the UNICOS system.

`/bin/dflawr`
`/bin/dflaww`
`/bin/dformat`
`/bin/dslip`
`/bin/dsurf`
`/bin/dverify`
	Commands that aid in disk flaw handling.

`/bin/ed`	Command to use when editing files on the system console.
`/bin/enstat`	A command to give Ethernet addresses attached to that IOS.
`/bin/mfdump`	A command that dumps data from the mainframe if a system crash or hang occurs.
`/bin/mt`	A command that manipulates tape devices without using the UNICOS tape daemon (`tpdaemon`).
`/bin/reload`	A command that reboots the IOS. If the IOS is not running (that is, if the boot prompt is displayed), use the `load` command instead.
`/bin/whatmic`	A command that displays information about the microcode in use on certain IOS peripherals.
`/config`	IOS configuration file.
`df`	A command that displays the amount of free space left on the IOS SCSI disk or on the system console disk. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`iosdump`	A command that dumps data from the IOS if a system crash occurs; this command is available from the IOS software and the IOS PROM. This

	command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`load`	A PROM command that boots the IOS (available only at the boot prompt; see also `/bin/reload`). This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`rcmd`	A command that initiates execution of another specified command on a slave IOS. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`/sys/param`	Model V based system configuration file. The UNICOS system configuration file (`/ios-param` is a copy of this file). **Note:** Cray J90, Cray J90se, and Cray SV1 series IOS-V is case sensitive, so this file must be referenced in all lowercase.
`time`	A command to check or change the IOS's date and time clock. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`version`	A command to display the version number of the running system. If entered at the IOS prompt, the IOS software version is displayed; if entered at the boot prompt, the IOS PROM version is displayed. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`CONTROL-a`	Terminal key sequence used to toggle between IOS and UNICOS consoles. `CONTROL-a` toggles between the IOS and UNICOS prompts. When going from the UNICOS prompt, the prompt changes to the IOS prompt after you press `CONTROL-a`. When going from the IOS prompt,

the prompt does not change until you press `RETURN`.

B.2 Commands Available from the UNICOS Console

The following commands and scripts are available from the UNICOS console of either GigaRing based systems or Model V based systems:

Command/Script	When to use
`/bin/mkdir`	A command that creates a subdirectory.
`/bin/tpstat`	A command that displays the current status of tape devices under control of the tape daemon (`tpdaemon`).
`/etc/brc`	A command that resets the `mnttab` file so that a file system can be mounted.
`/etc/bcheckrc`	A command that checks file systems to be mounted during setup.
`/etc/chown`	A command that changes the ownership of a file.
`/etc/config/rcoptions`	A command used to alter the `/etc/rc` script.
`/etc/config/tapeconfig`	A file used to configure the UNICOS tape daemon (`tpdaemon`).
`/etc/coredd`	A command that copies raw core dump files to a regular UNICOS file in a separate file system.
`/etc/cpdmp`	A command that copies the dump from the dump

	directory to a file for further processing.
`/etc/cpu`	A command to monitor and control CPU usage.
`/etc/crash`	A command used to analyze a dump file.
`/etc/csaboots`	A command that writes the boot record to the `/etc/casinfo` file.
`/etc/df`	A command used to check the amount of disk space available ("disk free").
`/etc/dump`	A command used to perform full or incremental backups.
`/etc/errpt`	A command used to display errors reported in the system's error file.
`/etc/fsck`	A command used to verify the consistency of a file system.
`/etc/fuser`	A command to list processes using a particular file.
`/etc/init`	A command that signals the `init` process to change to a different run level.
`/etc/killall`	A command to kill all processes. This command is also called by etc/shutdown.
`/etc/nu`	An interactive command used to add users. This command prompts you for account information such as the user

	password, login ID, and so on.
`/etc/passwd`	A command used to change a password.
`/etc/rc`	A script run automatically at start-up time that resets the system to multiuser mode.
`/etc/shutdown`	A command that puts the UNICOS system into single-user mode.
`/etc/setdev`	A command run automatically at start-up time

	that removes and remakes disk special files.
`/etc/udbgen`	A command that alters the user database.
`/usr/bin/chgrp`	A command used to change the group ownership of a file.
`/usr/bin/chmod`	A command used to change permissions on a file or directory.
`/usr/bin/du`	A command used to check disk usage statistics.
`/usr/bin/kill`	A command used to terminate a process.
`/usr/bin/mkdir`	A command used to create a directory in the current directory.
`/usr/bin/ps`	A command used to check status of active processes.
`/usr/bin/rmdir`	A command used to remove a specified directory.
`/usr/bin/who`	A command used to list information about logged-on users.
`/usr/lib/acct/startup`	A command that enables you to track per-process usage.
`/usr/lib/acct/ckpacct`	A command that checks the size of the accounting data files.
`/usr/lib/acct/ckdacct`	A command that checks the size of daemon accounting files.
`/usr/lib/acct/csarun`	A command that produces data file and accounting reports.
`CONTROL-a`	Terminal key sequence used to toggle between IOS and

UNICOS consoles for **Model V** based systems only. `CONTROL-a` toggles between the IOS and UNICOS prompts. When going from the UNICOS prompt, the prompt changes to the IOS prompt after you press `CONTROL-a`. When going from the IOS prompt, the prompt does not change until you press `RETURN`.

File Version Numbers [C]

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
```

The preceding 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 live with 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 back-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
```

IOS and Mainframe Dump [D]

If your Cray J90se or Cray SV1 series GigaRing based system or Cray J90, Cray J90se, or Cray SV1 series Model V based system experiences an IOS (MPN or SPN for GigaRing based systems) assertion panic, IOS 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.

The first section of this chapter explains what diagnostic information to send and where to send it.

The second portion of this chapter explains the procedures for capturing memory dumps from the system workstation (SWS) for **GigaRing** based systems.

The third portion of this chapter explains the procedures for capturing memory dumps, both from the IOS and from the mainframe (UNICOS) for **Model V** based systems.

D.1 Send Dump Results to Cray

Place dumps on digital audio tapes (DAT) media in `tar`(1) or `cpio`(1) format and send them to Global Product Support at Cray Inc. in Mendota Heights, 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 should also be forwarded for analysis.

D.2 Generating a Dump — GigaRing Based Systems

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 or Cray SV1 series 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.*

D.3 Generating a Dump — Model V Based Systems

This section lists the following procedures for obtaining dumps needed for Model V based systems:

- IOS Dump for Model V Based Systems
- UNICOS Dump
- Tips on Configuring `mfdump`
- Verifying That You Have Captured a UNICOS Dump

D.3.1 IOS Dump for Model V Based Systems

When either an IOS assertion panic or an IOS processor fault panic occurs, the firmware on the IOS automatically captures a dump of that IOS and saves it to the system console disk in the IOS file `/adm/dumpx/D` *mmddyy.n*. IOS dumps are usually 9 Mbytes for each IOP. In cases where an apparent hang or unusual system behavior has occurred, you must manually perform an IOS dump.

The `iosdump` command saves all of the memory from the IOP and selected areas from the I/O buffer board (IOBB) memory to the system console disk.

> **Note:** You can invoke the `iosdump` command at the `BOOT>` prompt or the `sn` *xxxx* `-ios0>` prompt.

The format of the `iosdump` command is as follows:

```
iosdump [-n filename] [-s iobbsize]
```

The arguments are optional. The default area dumped during an IOS dump is not a complete dump of IOBB memory; however, it usually contains enough information for problem analysis. To obtain a complete dump, use the `-s` option and enter the memory size in Kbytes of the IOBB. For example, for an I/O subsystem configured with an IOBB-64, you would enter `-s 16384`. Dumps that are automatically initiated by the IOS will be of the default size, and you cannot control this. However, if an automatic dump is captured and the IOS stops at the `BOOT>` prompt, you can then initiate another valid dump if you must capture the entire IOBB contents. If the IOS kernel has reloaded, the IOBB and IOP contents will have been overwritten, so dumps performed after a reload are not valid.

By default, the IOS generates a file name for the contents of the dump. The following example displays the output of the `iosdump` command:

```
sn9xxx-ios0> iosdump -s 16384

Saving IOS dump to /adm/dump0/D011895.0...
Dump file size, 1MB, 2MB, 3MB, 4MB, 5MB, 6MB, 7MB, 8MB
IOS0 dump completed
sn9xxx-ios0>
```

D.3.1.1 Dumping a Slave IOS for Model V Based Systems

To capture a dump of a slave IOS, use the `jcon` system console command. Perform the `iosdump` procedures as previously described, then enter the following command (*xxxx* is the serial number of your system; *n* is the IOS number of slave IOS):

```
IOS> jcon snxxxx-iosn
```

D.3.2 UNICOS Dump for Model V Based Systems

Before you capture a UNICOS dump for the first time, make sure that you allocated enough disk space for the dump on the dump partition (sometimes called the dump device); if not enough space is allocated, the dump will be truncated. Space is allocated in the IOS `/sys/param` file. The minimum size of the dump device should be a little larger than the amount of memory you actually want to examine in order to provide an additional 512 words for the dump header plus additional space for ublocks, which vary by system. You should start with a minimum of 50,000 blocks (sectors) for the dump size. An example of a dump partition entry in the physical and logical device portion of the UNICOS file system section of the IOS `/sys/param` file is shown in Figure 5, page 301. The dump partition usually is named `/dev/dsk/dump`.

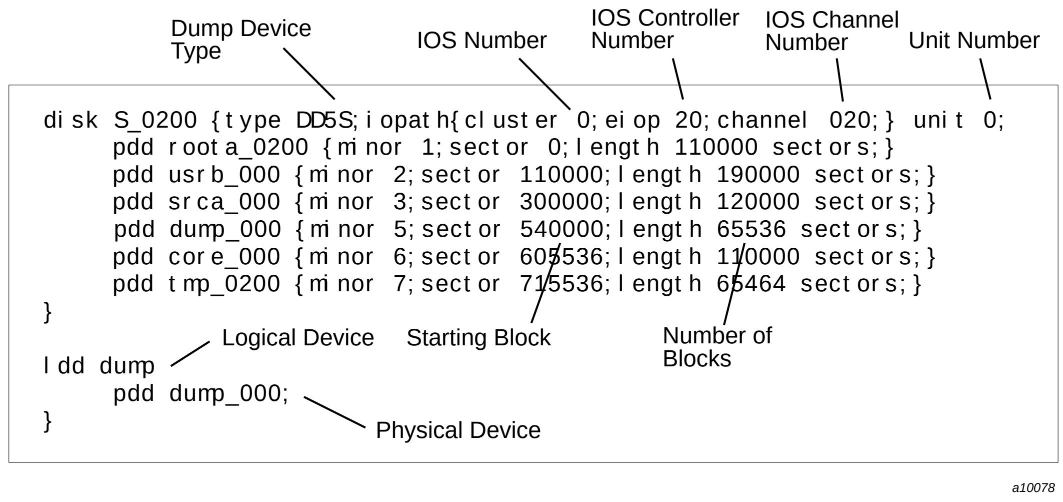
Also, before you capture a UNICOS dump for the first time, make sure that the dump partition was initialized. If not initialized, the dump will fail. Usually, the dump partition is initialized during installation. If the dump partition was not initialized when the UNICOS system was installed, use the `mkdmp`(8) command to initialize the dump partition.

To perform a UNICOS system dump, do the following:

1. If the IOS is not running, reset the VME.
2. If a dump has not already been done, perform an IOS dump using the `iosdump` command.
3. After the IOS dump has completed, reload the IOS.
4. Perform the mainframe dump using `mfdump`. (Refer to the following sections for more information about performing a dump using the `mfdump` command.)

See *CRAY IOS-V Messages*, for message explanations.

See the *CRAY IOS-V Commands Reference Manual*, or the online man pages for descriptions of IOS commands.



```
disk S_0200 {type DD5S; iopath{cluster 0; eiop 20; channel 020;} unit 0;
      pdd roota_0200 {minor 1; sector 0; length 110000 sectors;}
      pdd usrb_000 {minor 2; sector 110000; length 190000 sectors;}
      pdd srca_000 {minor 3; sector 300000; length 120000 sectors;}
      pdd dump_000 {minor 5; sector 540000; length 65536 sectors;}
      pdd core_000 {minor 6; sector 605536; length 110000 sectors;}
      pdd tmp_0200 {minor 7; sector 715536; length 65464 sectors;}
}

ldd dump
      pdd dump_000;
}
```

Figure 5. Dump Entry Example from IOS `/sys/param` File

D.3.3 Tips on Configuring `mfdump`

When you initiate multiuser mode, the `/etc/coredd` start-up script creates a subdirectory and copies into it a file containing the dump, the version of the UNICOS operating system that was running at the time of the crash, and the version of `crash`(8) to be used with the dump. The following are tips for configuring `mfdump` correctly.

- The `mfdump`(8) command is not supported from the `BOOT>` prompt. You must have the IOS loaded before running `mfdump`.
- When you run `mfdump` for the first time, check the `/sys/mfdumpa.arg` (ASCII version) argument file. Its contents are described on the `mfdump`(8) man page. An example follows. During installation, the `/sys/mfdumpa.arg` argument file is configured for the system. You can use the `mfdump -c` command line option to display the current settings for the dump configuration file.

  ```
  CPUS=4
  MEM=64
  range1=0-8000000
  range2=0-0
  range3=0-0
  range4=0-0
  regdump=yes
  ```

```
sysreg=yes
ublocks=yes
IOS#=0
channel#=16
disktype=DD5S
controller=20
unit=0
start=540000
length=65536
```

- If there is a system dump in the dump partition that has not been copied, use the `-f` command line option to force the dump. This will overwrite the previously uncopied dump.

D.3.3.1 Running `mfdump`(8)

The following example shows how to specify a reason for the dump by using the `mfdump` command with the `-r` option.

```
snxxxx-ios0> mfdump -f -v -r System panic on an ORE in user code
```

D.3.4 Verifying That You Have Captured a UNICOS Dump

After a UNICOS dump is captured using `mfdump`, it is copied off the dump device and into a file system when multiuser mode is initiated. See the example below for sample console output generated when the UNICOS system is booted in multiuser mode.

For more information about this process, see the `/etc/brc` script and refer to the UNICOS `mkdmp`(8) man page.

Example 13: Sample console output when the UNICOS system is booted in multiuser mode

```
/core: file system opened
/core: super block fname core, fpack core_000
/core: clean exit for clean file system
/etc/mount: warning <core> mounted as </mnt>
01446 blocks - NOT copied
coredd: Copying system dump into /mnt/08230838.
Sysdump copy completed
/etc/umount: /mnt unmounted successfully
```

Note: Many systems use the `/tmp` file system to store dumps instead of the `/core` file system as shown in the previous example.

Disk Capacities and Transfer Rates [E]

This appendix contains a section related to GigaRing based systems and another on IOS Model V based systems.

E.1 Disk Capacities and Transfer Rates — GigaRing Based Systems

Beginning with the UNICOS 10.0.0.8 release, for up-to-date information about disk device capacities and transfer rates for disk drives supported on Cray J90se or Cray SV1 series GigaRing based systems, see the following URL in CRINFORM:

`http://crinform.cray.com/PRIVATE/support_grid/diskspecs.htm`

This information was formerly contained in the following man pages:

`disksfcn`(7)	Physical specifications of disk devices connected to the Fibre Channel I/O Node (FCN-1).
`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).

E.2 Disk Capacities and Transfer Rates — Model V Based Systems

This section contains information about disk device capacities and transfer rates for Model V based systems.

E.2.1 DD-5I Disk Drives

The DD-5I disk drive is a high-performance, two-head, parallel enhanced IPI-2 drive.

The DC-5I disk controller is an intelligent and high-performance controller that can sustain the peak rates of four drives simultaneously to mainframe memory. You can attach up to four DD-5I drives to a DC-5I controller.

Reliability of the DD-5I disk drive is exceptionally high with a mean time between failures (MTBF) of 300,000 hours of power-on operation. No preventive maintenance is required. The following table shows DD-5I specifications.

Note: This drive is in Maintenance Mode support as of fourth quarter 2000.

Table 4. DD-5I Specifications

Unformatted capacity	3.4 Gbytes
Formatted capacity	2.96 Gbytes
Formatted capacity (in blocks)	723,000 512-word blocks (4096-byte blocks)
Number of disk platters	11
Data surfaces	20
Interface	IPI-2 (enhanced)
Transfer rate (peak)	12.4 Mbyte/s
Transfer rate (sustained)	6 Mbyte/s - 9 Mbyte/s
Average access time	11.5 ms
Maximum access time	23.5 ms
Minimum access time	1.7 ms
Average latency time	5.55 ms
Disk speed	5,400 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	2,738

E.2.2 DD-5S Disk Drives

The following table shows the specifications for DD-5S (SCSI) disk drives.

Note: This drive is in Maintenance Mode support as of fourth quarter 2000.

Table 5. DD-5S

Unformatted capacity	3.5 Gbytes
Formatted capacity	3.19 Gbytes
Formatted capacity (in blocks)	781,000 (4-Kbyte blocks)
Number of disk platters	11
Data surfaces	21
Interface	SCSI-2 Fast Wide
Transfer rate (peak)	6 Mbyte/s
Transfer rate (sustained)	3.2 Mbyte/s - 5 Mbyte/s
Average access time	11.5 ms
Maximum access time	23.5 ms
Minimum access time	1.7 ms
Average latency time	5.55 ms
Disk speed	5,400 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	2,738

E.2.3 DD-6S Disk Drives

The following table shows the specifications for DD-6S (SCSI) disk drives.

Note: This drive is in Maintenance Mode support as of fourth quarter 2000.

Table 6. DD-6S Specifications

Unformatted capacity	10.8 Gbytes
Formatted capacity	9.78 Gbytes
Formatted capacity (in blocks)	2,389,000 (4-Kbyte blocks)
Number of disk platters	14
Data surfaces	27
Interface	SCSI-2 Fast Wide
Transfer rate (peak)	7.2 Mbyte/s
Transfer rate (sustained)	4.2 Mbyte/s - 6.2 Mbyte/s
Average access time	11.5 ms
Maximum access time	24 ms
Minimum access time	1.7 ms
Average latency time	5.55 ms
Disk speed	5,400 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	4,925

E.2.4 DD-314 Disk Drives

The following table shows the specifications for DD-314 (SCSI) disk drives.

Note: This drive is in Maintenance Mode support as of fourth quarter 2000.

Table 7. DD-314 Specifications

Unformatted capacity	5.06 Gbytes
Formatted capacity	4.2 Gbytes
Formatted capacity (in blocks)	1,102,000 (4-Kbyte blocks)
Number of disk platters	10
Data surfaces	21
Interface	SCSI-2 Fast Wide
Transfer rate (peak)	7.4 Mbyte/s
Transfer rate (sustained)	7.4 Mbyte/s
Average access time	8.5 ms
Maximum access time	19 ms
Minimum access time	0.9 ms
Average latency time	4.17 ms
Disk speed	7,200 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	3,711

E.2.5 DD-318 Disk Drives

The following table shows the specifications for DD-318 (SCSI) disk drives.

Table 8. DD-318 Specifications

Unformatted capacity	11.7 Gbytes
Formatted capacity	9.1 Gbytes
Formatted capacity (in blocks)	2,221,679 (4-Kbyte blocks)
Number of disk platters	11
Data surfaces	21
Interface	SCSI-2 Fast Wide

Transfer rate (peak)	7.4 Mbyte/s
Transfer rate (sustained)	7.4 Mbyte/s
Average access time	8.5 ms
Maximum access time	19 ms
Minimum access time	0.9 ms
Average latency time	4.17 ms
Disk speed	7,200 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	5,274

Logical Device Cache Process [F]

Logical device cache is an optional feature that you may enable to reduce disk I/O wait time from a user's perspective. On Cray J90se and Cray SV1 series GigaRing based systems or Cray J90, Cray J90se, and Cray SV1 series Model V based systems, logical cache is defined in central memory (DRAM).

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 ldcache (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.

F.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. The system administrator easily configures 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 reads and writes 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.

F.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` *dev*	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
- ---- ----  ---------- --------  --------  -------  -----  -----
M  300   27    16727    30354      34865     1799     95.09  /dev/dsk/root
M  300   27    1729      4703       1399      254     84.63  /dev/dsk/home
M  300   27    6702     20794       6191      263     95.93  /dev/dsk/tmp
M  300   27      47        11         27        4     87.10  /dev/dsk/src

# ldcache -b
Cache 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
```

(This results in the output shown here.)

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
```

F.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 system 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 3, page 23.

Power Up and Power Down Procedures [G]

This appendix explains how to power up and power down a Cray J90, Cray J90se, or Cray SV1 series system mainframe cabinet. There are different procedures for GigaRing based systems and Model V based systems.

G.1 Powering Up/Down a Cray J90se or Cray SV1 series GigaRing Based System

This section contains the procedures on how to power up and power down a Cray J90se or Cray SV1 series GigaRing based system mainframe cabinet.

Procedure 53: Powering up a Cray J90se or Cray SV1 series GigaRing based 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.

   For more detailed information about booting the UNICOS GigaRing based system, see Section 3.1.2, page 24.

Procedure 54: Powering down a Cray J90se or Cray SV1 series GigaRing based system

To power down a Cray J90se or Cray SV1 series 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.

   For more detailed information about UNICOS system shutdown, see Section 3.1.3, page 30.

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.

G.2 Powering Up and Powering Down a Cray J90 Model V Based System

This section contains procedures for powering up and powering down a Cray J90 Model V based system mainframe cabinet. The procedures are similar for the Cray J90se or Cray SV1 series Model V based systems.

Procedure 55: Powering up a Cray J90 Model V based system

After the system console 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.)
4. Connect all mainframe and I/O cabinet AC plugs to main power. Do not turn on the circuit breakers.
5. Ensure that all individual components on the CCU panel are properly configured at this time.
   a. At the VME card cage, ensure that the VME and disk drives are set to Not Inhibit.
   b. At the central control unit (CCU) cabinet, verify that both `MARGIN` switches are set to `NOMINAL`, that the two voltage switches are set to `ENABLE`, that the `ALARM` switch is set to `ENABLE`, and that the `CONTROLS` switch is set to `LOCAL` (see Figure 6, page 320).
6. Click on the right mouse button at the system console. A menu that contains a `jcon` button appears.
7. Click on the `jcon` button.
8. Power on the I/O cabinet(s) first, then power on the mainframe cabinet by moving the circuit breakers (Figure 6) on the back of each cabinet to the `ON` position.
   a. Press the `CPU RESET` button on the CCU.
   b. Press the `VME RESET` button on the CCU.
9. The System Ready LED on the upper-left portion of the CCU switch panel should illuminate within about 5 seconds (see Figure 7, page 321).
10. Enter `load` at the boot prompt (>). ACT "first load" then scrolls up.
11. Close doors of all cabinets.

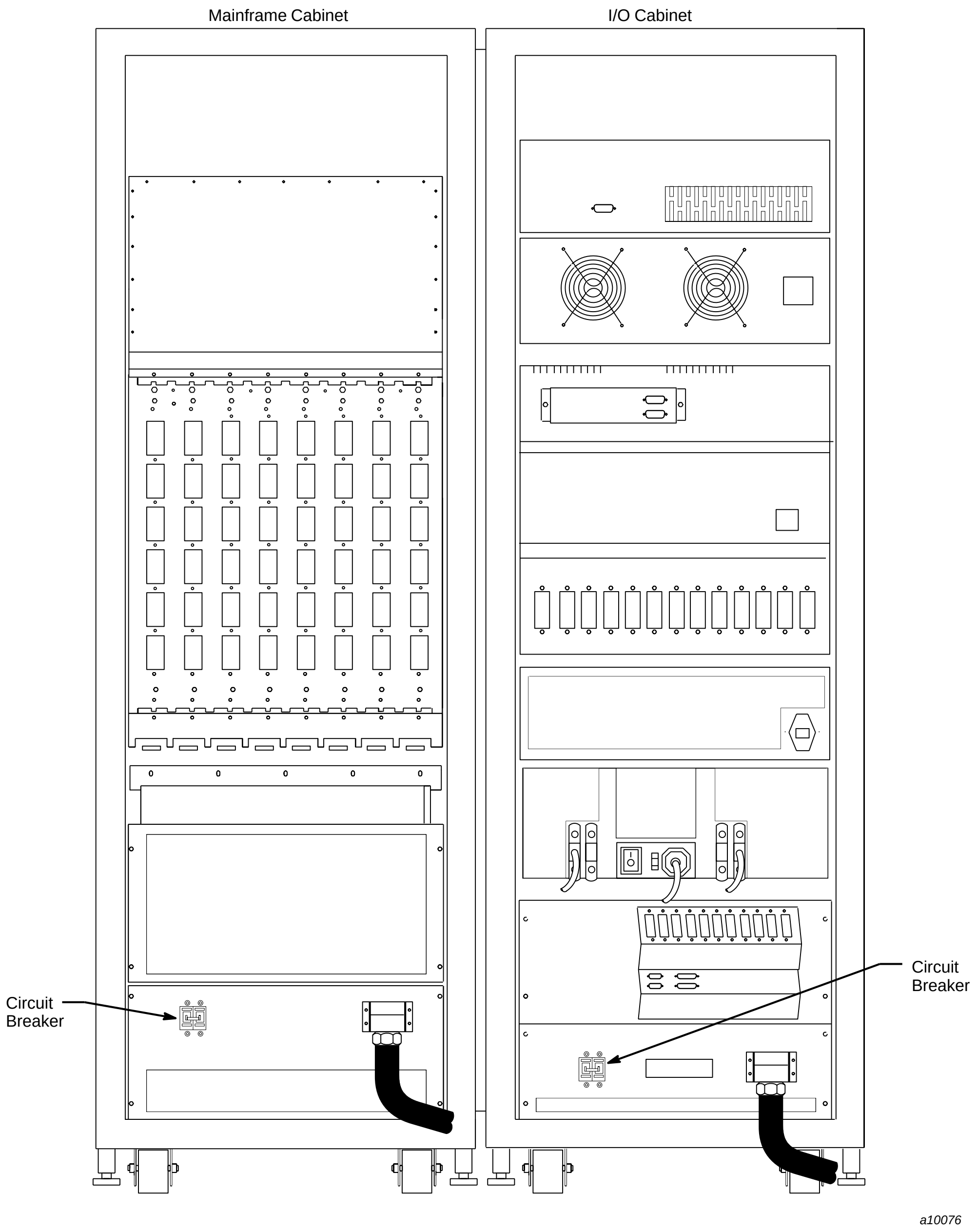


Figure 6. AC Circuit Breakers

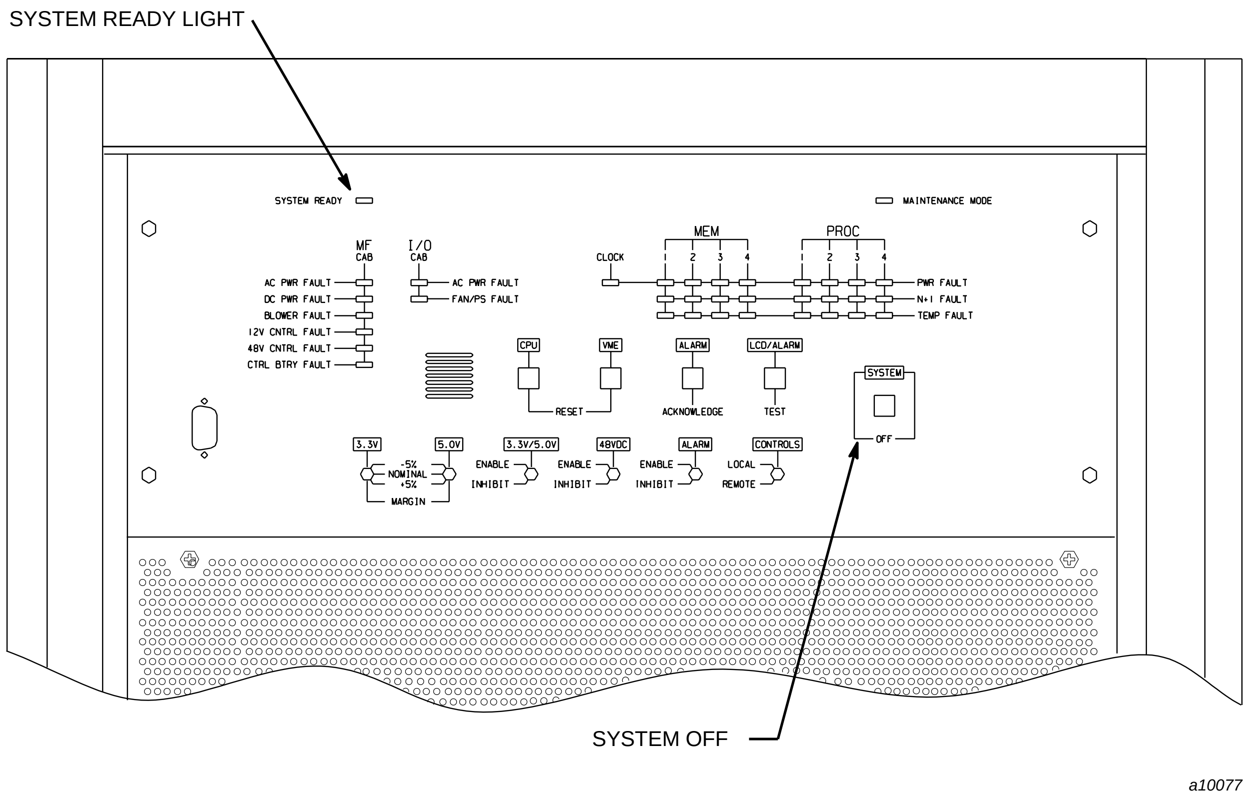


Figure 7. CCU

Procedure 56: Powering down a Cray J90 Model V based system

To power down a Cray J90 Model V based system, perform the following steps:

1. Log on to the system by entering the `jcon` command at the console.
2. 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
```

3. Stop the `jcon` connection by executing the following commands:

```
# <CONTROL-a>

(toggles to the IOS)

sn9xxx-ios0> mc
sn9xxx-ios0> reset

(takes 30 to 45 seconds)

BOOTsn9xxx-ios0> ~. <CONTROL-c>
```

4. Open the front door of the mainframe cabinet.

5. 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. Each cabinet circuit breaker trips to the 0 (`OFF`) position. All AC indicator lights on the control panel should now be off.

6. Ensure that all system lights are off.

Memory Configuration Parameters [H]

Use the following tables, based on your system's backplane configuration, to verify your system's NBANKS, CHIPSZ, and MEMORY parameters.

Table 9. NBANKS Values for Cray J916 2x2 Backplane

Memory Label	CHIPSZ	Memory Boards	NBANKS Value	MegaWords (MW)
8	M4MCH	2	128	32
V	M4MCH	2	128	32
0	M4MCH	2	256	64
P	M4MCH	2	256	64
B	M16MCH	2	128	128
Y	M16MCH	2	128	128
3	M16MCH	2	256	256
S	M16MCH	2	256	256
D	M64MCH	2	128	512
F	M64MCH	2	128	512
5	M64MCH	2	256	1024
U	M64MCH	2	256	1024

Table 10. NBANKS Values for Cray J916 4x4 Backplane

Memory Label	CHIPSZ	Memory Boards	NBANKS Value	MegaWords (MW)
8	M4MCH	4	256	64
V	M4MCH	4	256	64
0	M4MCH	4	512	128
P	M4MCH	4	512	128

Memory Label	CHIPSZ	Memory Boards	NBANKS Value	MegaWords (MW)
B	M16MCH	4	256	256
Y	M16MCH	4	256	256
3	M16MCH	4	512	512
S	M16MCH	4	512	512
D	M64MCH	4	256	1024
F	M64MCH	4	256	1024
5	M64MCH	4	512	2048
U	M64MCH	4	512	2048

Table 11. NBANKS Values for Cray J932 8x8 Backplane

Memory Label	CHIPSZ	Memory Boards	NBANKS Value	MegaWords (MW)
V	M4MCH	8	512	128
P	M4MCH	8	1024	256
Y	M16MCH	8	512	512
S	M16MCH	8	1024	1024
F	M64MCH	8	512	2048
U	M64MCH	8	1024	4096

Index