UNICOS® Basic Administration Guide for CRAY J90™ Model V based Systems

SG–2416 10.0

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

Record of Revision

Version	*Description*
8.0	March 1994 Original Printing.
9.0	March 1996 Reprint of this guide provided to support the CRAY J90 UNICOS 9.0 release.
10.0	October 1997 Reprint of this guide to support the CRAY J90 UNICOS 10.0 release.

Contents

Page

Page

Page

Page

Page

Page

Page

Page

Page

Figures

Page

Tables

Preface

This guide is written for system administrators of CRAY J90 systems with Model-V IOS running UNICOS 10.0. It includes information required for basic system setup and administration.

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

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

UNICOS system administration publications

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

- *General UNICOS System Administration*, Cray Research publication SG–2301, contains information on performing basic administration tasks as well as information about system and security administration using the UNICOS multilevel (MLS) feature. This publication contains chapters documenting file system planning, UNICOS startup and shutdown procedures, file system maintenance, basic administration tools, crash and dump analysis, the UNICOS multilevel security (MLS) feature, and administration of online features.
- *UNICOS Resource Administration*, Cray Research publication SG–2302, contains information on the administration of various UNICOS features available to all UNICOS systems. This publication contains chapters documenting accounting, automatic incident reporting (AIR), the fair-share scheduler, file system quotas, file system monitoring, system activity and performance monitoring, and the Unified Resource Manager (URM).

- *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303, provides information about the UNICOS kernel configuration files and the run-time configuration files and scripts.
- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, contains information on administration of networking facilities supported by the UNICOS operating system. This publication contains chapters documenting TCP/IP for the UNICOS operating system, the UNICOS network file system (NFS) feature, and the network information system (NIS) feature.
- *NQE Administration*, Cray Research publication SG–2150, describes how to configure, monitor, and control the Cray Network Queuing Environment (NQE) running on a UNIX system.
- *Kerberos Administrator's Guide*, Cray Research publication SG–2306, contains information on administration of the Kerberos feature, a set of programs and libraries that provide distributed authentication over an open network. This publication contains chapters documenting Kerberos implementation, configuration, and troubleshooting.
- *Tape Subsystem Administration*, Cray Research publication SG–2307, contains information on administration of UNICOS and UNICOS/mk tape subsystems. This publication contains chapters documenting tape subsystem administration commands, tape configuration, administration issues, and tape troubleshooting.

Related publications

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

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

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

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

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

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

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

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

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

Ordering Cray Research publications

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

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

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

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

Conventions

The following conventions are used throughout this document:

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

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

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

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

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

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

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

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

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

Reader comments

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

- Send us electronic mail at the following address:

  `publications@cray.com`

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

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

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

  +1–612–683–5600

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

We value your comments and will respond to them promptly.

Introduction [1]

This chapter discusses the system administrator's role, the log books you need to administer the system smoothly, and the characteristics of the UNICOS operating system. It provides a brief overview of the tasks explained in this manual and directs you to more information and documentation.

1.1 The role of a system administrator

A UNICOS system administrator provides, maintains, and ensures efficient and effective access to the Cray Research UNICOS computing environment. Users typically expect system administrators to have a broad base of skills and insight into many components of the UNICOS operating system. A system administrator of a Cray Research supercomputer running the UNICOS operating system may be responsible for some or all of the following tasks:

- Getting the system up and running and available for job submissions.
- Making the appropriate site-specific configuration changes.
- Resolving hardware and software problems.
- Taking care of the day-to-day administrative duties necessary to maintain a system and its users.

Daily administrative duties may consist of the following functions:

- Configuring and maintaining system accounting
- Backing up and restoring file systems (dumps and restores)
- Adding and deleting users
- Maintaining file systems and structures
- Tracking, analyzing, and resolving problems
- Configuring and administering the network
- Tuning the system and monitoring performance
- Upgrading and modifying the system
- Maintaining system security

1.2 Create and maintain a log book

To help you and your staff administer your system, it is essential that you create and maintain a log book, which should contain the following kinds of information:

- An incident report log, noting any problems that occurred and how the problem was resolved.
- Backup logs, including any scripts used to perform backups, the location of backup tapes, and any other pertinent details that relate to backups.
- System crash log and crash recovery procedures.
- Local documentation, detailing site-specific procedures, such as operator procedures, backup procedures, and so on.
- Listings and full path names for any essential scripts or files (especially the current configuration and parameter files).
- Emergency phone numbers, the names of any contact people, and any other emergency procedures that are relevant for the site.

Always keep the log book as current as possible; when you are trying to troubleshoot system problems, an up-to-date log book can be invaluable.

1.3 Major characteristics of the UNICOS operating system

Based on the UNIX System V operating system with Berkeley extensions, the UNICOS operating system is both an interactive and batch operating system that offers many advantages in performance, functionality, application portability, and connectivity.

The UNICOS operating system combines all of the inherent strengths of UNIX, such as its familiar user interface, with production-oriented features, including high-performance I/O, multiprocessing support, ANSI/IBM tape support, resource allocation and control, enhanced process scheduling, and an advanced batch processing subsystem called the Network Queuing System (NQS).

The following sections describe the major characteristics of the UNICOS operating system.

1.3.1 High-performance I/O

The UNICOS operating system can perform asynchronous I/O operations, used in multitasking applications, allowing 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.

1.3.2 File systems

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

1.3.3 Disk devices

The UNICOS operating system permits the use of disk striping and banding techniques for improving file system performance and reliability. A unique language, called the *configuration specification language* (CSL), is used to define the physical and logical characteristics of your UNICOS disk devices.

1.3.4 File system quotas

File system quotas have been implemented under the UNICOS operating system to control the amount of file system space consumed. You may set quotas for three different ID classes (user, group, and account IDs). Two different types of quotas are supported (file and inode).

1.3.5 User database (UDB)

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

1.3.6 Resource control

Resource control was added to the UNICOS operating system to permit a system administrator to set limits on CPU, memory, tapes, and file allocation. User limits are applied to processes or jobs, and they establish the maximum

amount of a resource that can be consumed. You can specify limits for interactive and batch workloads, as well as for per process and per job. This lets a system provide restricted resources for interactive use, without limiting a user's batch resources to the same degree.

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

1.3.7 Unified Resource Manager (URM)

The Unified Resource Manager (URM) is a job scheduler that balances the demands of both batch and interactive sessions. URM provides a high-level method of controlling the allocation of system resources to run jobs that originated either in batch mode or in an interactive session.

For more information on URM, see *UNICOS Source Manager (USM) User's Guide*, Cray Research publication SG–2097.

1.3.8 Fair-share scheduler

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

1.3.9 System accounting

The UNICOS operating system supports two kinds of system accounting; the standard System V version and Cray Research system accounting (CSA). CSA is designed to meet the unique accounting requirements of Cray Research customers. Like the standard System V accounting package, CSA provides a method to collect per-process resource usage data, to record connect sessions, to monitor disk usage, and to charge fees to users. CSA also permits sites to perform per-job and device accounting, along with daemon accounting. Individual sites can select which accounting system they want to use simply by starting the appropriate shell scripts and programs.

1.3.10 TCP/IP

The Transmission Control Protocol/Internet Protocol (TCP/IP) suite provides network communications that use the TCP/IP family of protocols and

applications. It allows Cray Research systems to become a peer node of any established TCP/IP network and permits other users and networks to access the UNICOS environment.

1.3.11 Network Queuing Environment (NQE)

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 known as a batch complex. Batch requests are submitted from NQE clients and executed at NQE master and execution servers.

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

1.3.12 Network Queuing System (NQS)

The Cray Research NQS product lets users submit, terminate, monitor, and control jobs submitted to either the local system or another appropriately configured computer system within your network.

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

1.3.13 Menu system

The UNICOS operating system contains a set of shell scripts, parameter files, and a user interface written in menu specification language (MSL). You may use the menu system to perform configuration changes after you have installed the UNICOS operating system.

For more information on the menu system, see *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412.

1.3.14 Data migration

The optional UNICOS Data Migration Facility (DMF) tries to ensure the availability of file system space by moving selected files from online disks to an offline storage device. The files remain cataloged in their original directories and behave in most ways as though they were still disk resident. Online disk can be considered a cached copy of a larger virtual disk space. The UNICOS DMF is not included as part of the standard UNICOS operating system software package; it is available as an optional software package.

For more information on DMF, see the*Cray Data Migration Facility (DMF) Administrator's Guide*, Cray Research publication SG–2135.

1.3.15 System activity monitor (SAM)

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

For more information on the system activity monitor, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

1.4 How this guide will help you

After you boot your IOS and UNICOS operating system software and bring it to multiuser mode by following your UNICOS operating system installation guide (*UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271) or your *Open First* documentation, this guide will enable you to perform each of the following tasks:

- Establish and maintain basic system security; see Chapter 2, page 11.
- Start up and shut down the IOS and UNICOS operating system; see Chapter 3, page 19.
- Verify and change date and time of both the IOS and UNICOS operating system ; see Chapter 3, page 19.
- Start and stop UNICOS system daemons; see Chapter 4, page 45.
- Determine existing file systems; see Chapter 5, page 51.
- Plan and configure file systems; see Chapter 5, page 51.
- Create, label, mount, and check the integrity of a file system; see Chapter 5, page 51.
- Monitor disk usage; see Chapter 5, page 51.
- Back up and restore a file system; see Chapter 6, page 133.
- Create and maintain user accounts; see Chapter 7, page 171.
- Communicate with your system users; see Chapter 8, page 217.

- Interpret system logs and determine when to "clean up" logs; see Chapter 9, page 225.
- Set up Cray system accounting (CSA) and monitor accounting functions; see Chapter 10, page 241.
- Add your CRAY J90 system to an existing network; see Chapter 11, page 275.
- Configure NIS; see Chapter 12, page 289.
- Configure NFS; see Chapter 13, page 299.

This guide contains several appendixes that may be of interest to you.

This guide also refers you to other publications for additional information you may need to perform more advanced system administration tasks.

Although each topic described in this guide includes a list of documentation you can read to get a greater understanding of the topic, the following list identifies some additional topics not covered in this guide that you may want to learn about to determine whether you should use the functions to administer your CRAY J90 system.

For information about	Read
File system space monitoring	SG-2302; `df`(1) and `du`(1) man pages
File system quotas	SG-2302
System activity monitoring	SG-2302; `sag`(1), `sar`(8), `sdc`(8), `tsar`(8), and `timex`(1) man pages
Automated incident reporting (AIR)	SG-2302; `aird`(8), `airdet`(8), `airprconf`(8), `airsum`(8), and `airtsum`(8) man pages
Job and process recovery	SG-2301; `chkpnt`(1), `chkpnt`(2), and `crash`(8) man pages
Reinstalling your system software	SG-5271
Updating your system software	SG-5271
Using the `cron`(8) and `at`(8) utilities	SG-2301; `at`(1) and `cron`(8) man pages

Configuring network interfaces	SG-2304
Monitoring networks	SG-2304
Unified Resource Manager (URM) centralizes resource allocation with a formal method of communication	SG-2302
Fair-share scheduler	SG-2302; `shradmin`(8) and `shrdist`(8) man pages
Memory scheduling	SG-2302
Multilevel security (MLS)	SG-2301
UNICOS message system	`explain`(1) man page
Data migration facility (DMF)	`dmmode`(2), `dmofrq`(2), `dm`(4) and `dmf_offline`(3C) man pages
Tape subsystem	SG-2307

1.5 UNICOS online glossary

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

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

striped group

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

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

Basic System Security [2]

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. The UNICOS operating system includes programs that provide the necessary tools for the creation of a set of procedures that lets you automate much of the daily work of monitoring system security. This chapter discusses security issues in four 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*, Cray Research publication SR–2022: `diskusg`(8) man page
- *General UNICOS System Administration*, Cray Research publication SG–2301
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `chown`(1), `du`(1), `find`(1), `login`(1) `su`(1), and `umask`(1) man pages

2.2 Super-user privileges

As in standard UNIX systems, 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 that govern 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 system 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 unauthorized users 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 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` command to acquire super-user privileges uses this password.

To maintain secure access to the root account, you should use the following steps:

- Ensure that the `root` password is not obvious and is 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.
- Change the root password frequently, at least once a month.
- Do not write down the `root` password.
- Ensure that the `root` password is known to as few people as possible; generally, these should be the system administrator and the administrator's staff.

You can monitor the use of the `root` password, and catch potential security breaches, by checking the `/usr/adm/sulog` file. (For information about system logs, see Chapter 9, page 225.) You can compare the log entries 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).

2.2.2 Physical security

A person who has access to the system workstation (SWS) 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, your SWS and your system itself should be physically accessible only to those persons who genuinely need that access, and your SWS should not be left unattended while you are logged into the system. If this is not possible, your SWS should at least be monitored to prevent unauthorized persons from trying to enter commands on the system console.

Store all removable media in a secure location. Always store backup tapes and cartridges and other media in a location different than your system. You also should make certain that any external media is in a physically secure location.

2.2.3 setuid programs

An executable UNICOS program may have the `setuid` (set user ID) 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 `root` (user ID 0) owns and has the `setuid` bit on can 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, you should make regular checks of all disk partitions on the system for programs that have a `setuid` or a `setgid` (which is the set group ID) of 0.

The `find`(1) command can generate a list of all `setuid` or `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 | xargs sum
```

Compare this list against a list of known `setuid` or `setgid` 0 programs. Any new `setuid` or `setgid` 0 programs that are not on the known list and whose creation you cannot account for may indicate a security breach.

As administrator, you should check the list of known `setuid` or `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 (that is, were made by you or a member of your staff). Unknown modification of a `setuid` or `setgid` 0 program may indicate a security breach. To generate a checksum and block count list of a file, you can use the `sum`(1) command; to do this, you can pipe the preceding `find` command line through the `sum` command as follows and then compare any changes in sizes:

`| sum >` *filename*

To ensure that write permission on each file is properly restricted, you also should check the list of known `setuid` or `setgid` 0 programs.

Because checking the entire system for `setuid` or `setgid` 0 programs uses a lot of I/O and CPU time, you should perform this check during off-peak hours.

To make the task less obtrusive, use the `cron`(1) or `at`(1) command to perform the check automatically.

2.2.4 `root` PATH

The `PATH` environment variable consists of a list of the directories that the shell searches 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) command changes the `PATH` after a user has entered the `root` password to successfully assume super-user privileges.

To make sure that the path has not been changed in either place since the last approved change, you should monitor both places occasionally.

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`. In addition to performing the expected system function (listing the files in the current directory), the new `ls` utility makes a copy of a program such as `sh` and turns on the `setuid` bit on the copy. An unsuspecting super user who has 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.

2.3 User security

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

2.3.1 `umask` command

The system default umask value is usually set in `/etc/profile` by using the `umask`(1) command. It lets you choose the permissions that typically will 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.

Generally, only the owner of the file should have write permission, which makes a default umask value of `022` appropriate. If members of a given user group should not be able to read the files of other user groups, you should use a umask value of `026` to remove other read permission.

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` command themselves, do not make the default umask value too stringent, because 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 value to open their files. As an alternative, they may want to use the groups mechanism; see Section 2.3.3, page 16.

2.3.2 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 preceding the program name with `./`, which explicitly indicates 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. The current directory should thus be the last entry in the default `PATH`, after the normal system directories.

2.3.3 User groups

You can enhance user security by the careful placement of users into groups. Generally, when deciding on the placement of users into groups, you should use factors external to the system. 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 who have access to each other's files. Members of most logical groups (for example, members of a software development project) want to share files with one another, and the default umask should permit this.

To prevent inappropriate sharing of data, you should create a group that has only one user in it, rather than create a default "other" or "miscellaneous" group for users who do not fit elsewhere. Because users may belong to more than one group, and groups are active simultaneously, you also may 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.

2.3.4 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 that involves several other users. Files that reside in one user's home directory tree may also be owned by another user.

Users may become confused by this situation and then use the `chown`(1) command 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). To get a general idea of the users who trade ownership of files, you can use the `diskusg`(8) and `du`(1) commands together.

2.3.5 Login attempts

Unauthorized users might try to gain access to the system by making repeated attempts to log in. 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)).

2.4 Partition security

When administered properly, the UNICOS file system should provide adequate protection for user and system files. To enhance system security, however, mount file systems 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 their files during nondedicated time or by not mounting the file systems that contain other users' accounts during dedicated time. (For more information about file systems, see Chapter 5, page 51.)

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`.

Example:

```
brw-------   1 root     root       0,245 Nov 28 20:24 bk_udb
brw-------   1 root     root       0,247 Nov 28 20:24 bkroot
brw-------   1 root     root       0,246 Nov 28 20:24 bkusr
```

2.5 Tape device access

For CRAY J90 systems, you should be using the tape daemon character-special tape interface. The character-special tape interface provides unstructured access to the tape hardware similar to the traditional UNIX method of accessing tape devices. It is useful in performing specific tasks, such as the following:

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

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

Startup and Shutdown [3]

This chapter includes procedures to do the following:

- Start the CRAY IOS-V and CRAY J90 mainframe and bring up the UNICOS system to a multiuser run state mode (startup; also called *booting*).
- Bring the UNICOS system 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.

If you have access to a windowing environment, the UNICOS operating system provides a point-and-click, X Window System based interface to the UNICOS Installation / Configuration Menu System. For more information, see the *UNICOS System Configuration Using ICMS*, Cray Research publication SG–2412.

To start and stop UNICOS system daemons, see Chapter 4, page 45.

3.1 Related startup and shutdown documentation

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

- *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `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 Research publication SR–2170
- *CRAY IOS-V Messages*, Cray Research publication SQ–2172

Procedure 1: Starting up the system

Note: The CRAY J90 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 CRAY J90 console, press the right mouse button in the OpenWindows root window and select the `J90 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 CRAY J90 systems.

```
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 RETURN`.

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:

```
sn1234#
```

   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-aRETURN
(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); for the procedure, see Procedure 7, step 3, page 125.

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 CRAY J90 IOS-V system console (CRAY J90 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 (for more information, see Section 3.4.4, page 34).

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. For additional information about run-level configuration, see Section 3.4.1, page 30, and Section 3.5, page 36.

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.

Procedure 2: Shutting down the UNICOS system and the IOS

To shut down the UNICOS system and the IOS, follow these 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 217). Before executing `/etc/shutdown`, you can use the `ps -eaf` command to see processes that are running, and the `who -u` command to see whether people are actively using the system. The `shutdown`(8) command uses the following format:

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

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

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

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

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

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

(if you are using ldcache)

```
# df
```

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

```
# /bin/sync
```

At this point, you are in single-user mode but the UNICOS system 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 F, page 335, or see your hardware installation manual).

3.2 Shutdown information

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. For CRAY J90 series systems, you do not have to modify the `/etc/shutdown` script directly.

3.2.1 User exits

The `/etc/shutdown` script provides three user exits (`shutdown.pre`, `shutdown.mid`, and `shutdown.pst`) that allow you to modify the shutdown process.

3.2.1.1 `shutdown.pre`

The `shutdown.pre` script is the first user exit of the `shutdown` script. If an executable named `/etc/shutdown.pre` exists, it will be executed during shutdown. At this point, nothing has been done toward shutting down the system. All daemons are still running, all file systems are mounted, and all users are still active and unaware that this script is running.

A possible use of this exit would be to verify the user's permission to run the `shutdown` script or to run some system cleanup routines. The `shutdown` script will check the return status from the `shutdown.pre` program. If the return status is nonzero, the user will be queried as to whether or not to continue the shutdown processing. At this point, the shutdown can be stopped without any effect on the system.

3.2.1.2 `shutdown.mid`

The `shutdown.mid` script is the second user exit of the `shutdown` script. If an executable named `/etc/shutdown.mid` exists, it will be executed during shutdown.

At this time, all processes (users and daemons) have been terminated, all the disk cache (ldcache or pcache) has been released, but the network interfaces are still configured, and all of the file systems are still mounted.

A possible use of this exit would be to allow NFS file systems to be unmounted before the networks are stopped.

The `shutdown` script will check the return status from the `shutdown.mid` program. If the return status is nonzero, the user will be queried whether to continue the shutdown processing or not. This exit is given to address any possible problem that may exist with the file systems still mounted and the networks that are still running.

3.2.1.3 `shutdown.pst`

The `shutdown.pst` user exit is the third (and last) user exit of the `shutdown` script. If an executable named `/etc/shutdown.pst` exists, it will be executed during shutdown. At this point, all processes (users and daemons) have been terminated, but the file systems are still mounted. This is virtually single-user mode, except for the file systems.

After this point, the file systems are unmounted and `/etc/init` is invoked to go to single-user mode. The `/etc/init s` command will kill all remaining processes (including the process running the `shutdown` script), so there is no place to put a user exit beyond this point.

Because the system is virtually shut down by this point, there is no reason to halt the script if the user exit return status is not zero. The status returned from `/etc/shutdown.pst` is checked, but `shutdown` will only issue a warning message and then go to single-user mode.

> **Note:** Be careful in what you allow `shutdown.pst` to execute. Because the various logging daemons (such as `syslogd` are not available to free up space, `shutdown.pst` could potentially fill up the file system(s) that contain the log files.

3.2.2 Shutdown process

If you install and start any local processes or daemons during the execution of the `/etc/rc` script, to stop them within the `/etc/shutdown.sh` script, you can add control information into the `/etc/config/daemons` file and insert `/etc/sdaemon` commands into the `/etc/shutdown.sh` script. You should make any changes either in `/etc/config/rcoptions` or in one of the user exits; **do not** modify `/etc/rc`. For information on starting and stopping UNICOS system daemons, see Chapter 4, page 45.

The process of a UNICOS system shutdown is as follows:

1. Executes the user exit `/etc/shutdown.pre`, if it exists. If a nonzero return status is returned from the user exit, `shutdown` will prompt the user for confirmation before continuing.

2. Sends a message, using `wall`(8), warning the users who are currently logged in to the system that the system is being shut down.
3. Shuts down the NQE subsystem to allow batch jobs to be checkpointed before they are terminated.
4. Sends a `SIGSHUTDN` signal to all currently running processes.
5. Stops the DM daemon, Tape daemon, and error logging.
6. Shuts down daemons in the `SYS1` and `SYS2` groups (defined in the `/etc/config/daemons` file), using the `sdaemon`(8) command.
7. Sends a `SIGHUP` signal to all currently running processes.
8. Sends a `SIGKILL` signal to all currently running processes.
9. Shuts down system accounting, using the following command (see `acctsh`(8)):

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

   `shutacct`(8) records the action of shutting down system accounting in the `/etc/wtmp` file.
10. Releases partition cache, using the `ldcache`(8) command. This ensures that all partition cache buffers are flushed.
11. Executes the user exit `/etc/shutdown.mid`, if it exists. If a nonzero return status is returned from the user exit, `shutdown` will prompt the user for confirmation before continuing.
12. Shuts down all configured network interfaces (defined in the `/etc/config/interfaces` file), using the `ifconfig`(8) command.
13. Executes the user exit `/etc/shutdown.pst`, if it exists. If a nonzero return status is returned from the user exit, a warning message will be printed (the shutdown cannot be stopped at this point, because all daemons and processes have been terminated).
14. Unmounts all file systems. If any local file systems cannot be unmounted, the `shutdown` script will issue a warning message.
15. Brings the system to single-user mode, using the `init`(8) command with an `s` argument, as follows:

    ```
    /etc/init s
    ```

3.3 Startup, shutdown, and configuration files and scripts for IOS and the UNICOS system

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

Note: The CRAY J90 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 CRAY J90 systems. 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 CRAY J90 systems. 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 45.

`/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 IOS `/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 IOS disk in `/sys/unicos.ymp`.

3.4 Start-up scripts

This section describes the `/etc/init` command and start-up scripts.

3.4.1 The /etc/init command

The /etc/init command is the process control initialization command and is invoked as the last step in the UNICOS system boot procedure. init is the process from which all other processes are spawned, either directly or indirectly. The process ID (PID) of init is always 1.

At any moment, init considers the system to be in one of eight different run levels: through 6, or S (s) (a run level of S or s refers to single-user mode). When you specify S, init operates in single-user mode with the additional result that /dev/syscon is linked to the user's terminal line, thus making it the virtual system console). For more information about run-level configuration, see Section 3.5, page 36.

By default, init considers the system to be in run level S at the end of the normal system boot procedure.

For further details about /etc/init, see the init(8) man page.

3.4.2 The /etc/inittab file

The /etc/inittab file contains directions for actions when changing run levels. Each entry within the /etc/inittab file contains four fields, separated by colons.

These fields identify and provide the "when," "how," and "what" to the /etc/init process, which starts all processes as specified in the /etc/inittab file.

Field	Description
ID	A label that uniquely identifies the entry. The label can consist of a maximum of 4 characters.
run state	Run level in which an entry should be processed. A null entry (two colons) indicates that the entry must be executed when changing to any numbered (0 through 6) run state. Numbered run states signify varying levels of UNICOS system functionality and rely on the /etc/init process starting or stopping system processes as required. See the init(8) man page. The run state field is the "when" portion for the entry.

`action`	The action field specifies "how" to start the command or program specified in the process field for this entry.
`process`	The command or the name of the program to execute. This action field is the "what" portion of the entry. To insert comments into this field, prefix a line with a # symbol.

The following values are action field values for `/etc/inittab`:

Value	Description
`boot`	Starts process at multiuser boot time if the specified run level matches the `init` run level at boot time. `init` does not wait for the process to terminate. When the process dies, `init` does not restart it.
`bootwait`	Starts process at multiuser boot time if the specified run level matches the `init` run level at boot time. `init` waits for the process to terminate. When the process dies, `init` does not restart it.
`generic`	Instructs `init` to accept login requests from privileged daemons through the `/etc/initreq` FIFO special file (named pipe).
`initdefault`	Specifies run level to enter when `init` is initially invoked. If no `initdefault` action field entry exists in `inittab`, `init` requests an initial run level from the user at boot time.
`ldsynctm`	Sets the ldsync rate (the frequency with which the `init` daemon causes the data in ldcache to be flushed to disk). The default is 120 seconds.
`off`	If process is running, sends it a `SIGTERM` signal, waits 20 seconds, and then sends it a `SIGKILL` signal if it is still running. If process is not running, it does not restart it.
`once`	Starts process, does not wait and does not restart it.
`respawn`	Starts process and restarts it when it dies.

`sysinit`	Starts process at system boot time, before accessing system console, and waits for its termination before proceeding.
`timezone`	Establishes the systemwide value of the time zone; this value is exported to all processes spawned by `init`.
`wait`	Starts process and waits for its termination before proceeding.

The `/etc/inittab` file should have the following attributes:

- The initial run level (specified by an entry with the action field `initdefault`) should be single-user mode (specified by the letter `s` in the run state field).
- Following the `initdefault` entry, an entry with the action field `timezone` should exist to set the `TZ` environment variable to the appropriate value for the time zone in which the system is located.
- Following the `timezone` entry, calls to shell scripts should actually initialize the system's state for the run level being entered. By convention, the `bcheckrc` (see `brc`(8)) program is called by an entry with the action field `bootwait` to perform boot-time-only actions, and the `rc` (see `brc`(8)) program is called by an entry with the action field `wait` to perform actions for switching from one run level to another (including switching from the initial single-user mode to multiuser mode).
- An entry with the action field `wait` should link the special file `/dev/systty` to `/dev/syscon`.
- An entry must exist for all run levels, with an action field of `respawn`, which executes the following command (see `consoled`(8)) to allow logins on the system console:

  ```
  /etc/consoled
  ```

- Any run levels that accept logins from users on front-end systems need an entry with an action of `generic`. This entry instructs `init`(8) to accept login requests from daemons through the `/etc/initreq` FIFO special file (named pipe). This is true even when the run level is intended for use by just one dedicated user; you should restrict access to the system by using the `rc`(8) script (see Section 3.4.6, page 35), rather than limiting logins to specific devices, as is often done on traditional UNIX systems.

A sample `/etc/inittab` file follows:

```
#  more /etc/inittab
is:S:initdefault:
tz::timezone:TZ=CST6CDT
sd::sysinit:/etc/setdate 1>/dev/console 2>&1 #setdate from iop
bl::bootwait:/etc/bcheckrc </dev/console >/dev/console 2>&1 #bootlog
bc::bootwait:/etc/brc </dev/console >/dev/console 2>&1 #bootrun command
rc:2:wait:/etc/rc </dev/console 1>/dev/console 2>&1 #run com norm not just 2
pf::powerfail:/etc/powerfail 1>/dev/console 2>&1 #powerfail routines
fe:2:generic:#no command to execute
co::respawn:/etc/getty console console
lt::ldsynctm:300
```

3.4.3 Interaction between `/etc/init` and `/etc/inittab`

When you boot the UNICOS system, or signal `/etc/init` to change to another run level, `/etc/init` reads the `/etc/inittab` file for directions. Command lines whose run state fields match the desired new run level are executed sequentially.

> **Note:** The `/etc/init` command reads and processes entries in your `/etc/inittab` file sequentially. The order of the entries is important and determines the sequence followed when booting your UNICOS system. Except for the `timezone` entry, you should not have to modify your `/etc/inittab` file. For details on the structure of the `/etc/inittab` file, see the `inittab`(5) man page.

If you enter a digit from 0 to 6, `/etc/init` enters that multiuser run level. If you have signaled `init` to change from a single-user run level (S or s) to a multiuser run level (0 to 6), `init` scans the `/etc/inittab` file for any entry that has a `bootwait` or `boot` action field type. Any `bootwait` or `boot` type entries are executed before any normal processing of the `inittab` file occurs. This step ensures that any system initialization happens before anyone (including the system administrator) gains access to the system.

The following single-character arguments are used to signal the actions of `init`:

- 0 through 6 places the UNICOS system in one of the multiuser run levels.
- S or s places the UNICOS system in single-user mode.

While the system is running, a system administrator can use the `init q` command to force `init` to reread `inittab`. In this way, `init` can be made aware of changes to `inittab` without changing run states.

By default, the UNICOS system has a standard `/etc/inittab` file that is designed to activate the `/etc/rc` script when you execute an `/etc/init 2` command.

3.4.4 `/etc/bcheckrc` script

The `/etc/bcheckrc` script is one of the start-up scripts that `/etc/init` invokes when it reads through the `/etc/inittab` file. The `/etc/bcheckrc` script performs two major activities. It resets the system date, if necessary, and checks all file systems that will be mounted during the start-up process.

This script is invoked only the first time you change the system from single-user to multiuser mode after a reboot.

> **Note: Do not** change or reset the system date and time when the system is running in multiuser mode. For more details, see the `date`(1) man page. You should set or change the date and time only when starting multiuser mode.

After checking and, if needed, setting the system date, the `/etc/bcheckrc` script invokes the `/etc/mfsck` utility. The `/etc/mfsck` command runs several copies of `/etc/fsck`(8) in parallel, which can speed up system startup. Usually, the `/etc/mfsck` command runs several passes, checking all file systems; only the `root` (`/`) file system is checked during the first pass. The `/etc/fstab` file (see `fstab`(5)) determines when other file systems are checked.

The only UNICOS start-up scripts you should modify are the `/etc/rc.pre`, `/etc/rc.mid`, and `/etc/rc.post` scripts. To change the behavior and actions of other UNICOS start-up scripts easily, modify the `/etc/config/rcoptions` file and the `/etc/config/daemons` configuration file. For additional information, see Section 3.4.6, page 35, Section 3.4.7, page 35, and Section 3.4.8, page 36.

3.4.5 `/etc/brc` script

The `/etc/brc` script is invoked by `/etc/init` through `/etc/inittab` and is intended for use in initializing hardware devices. It also copies system dumps into a separate file system by executing the `/etc/coredd`(8) script. The `/etc/brc` script should not be used to start processes because those processes will be killed and not restarted during the subsequent system shutdown or startup.

Like the `/etc/bcheckrc` script, the `/etc/brc` script is invoked only the first time you change from single-user to any multiuser (numeric) run level.

3.4.6 The multiuser start-up script /etc/rc

The /etc/rc script is invoked by /etc/init when the UNICOS system goes from a single-user to multiuser run level.

Note: Do not modify the /etc/rc start-up script; to alter the behavior of your script, make any needed changes to the /etc/config/rcoptions file.

The /etc/rc.log file collects messages generated during execution of /etc/rc. The /etc/rc.log file is cleared during execution of /etc/rc and messages written to the /etc/rc.log file during an earlier execution of /etc/rc are lost. Not all start-up output is written into the /etc/rc.log file.

When finished, /etc/rc returns control to /etc/init, which then continues reading and processing subsequent lines from /etc/inittab.

For more information about the /etc/rc script, see *General UNICOS System Administration*, Cray Research publication SG–2301.

3.4.7 Using rcoptions to modify the actions of /etc/bcheckrc, /etc/brc, and /etc/rc

You should not modify the start-up scripts manually to alter their behavior. Instead, you must manually edit the control file that the UNICOS system uses for configuration and installation.

The control file used to alter the actions of the various start-up scripts is /etc/config/rcoptions.

The RC_LOG parameter changes the name of the log file (/etc/rc.log) used to capture output messages generated during execution of the /etc/rc script.

You may use rcoptions to do the following:

- Set the device name for the /usr, /usr/tmp, and /tmp file systems.
- Set the path name of the /etc/rc log file.
- Specify whether to run mkfs on /tmp or /usr/tmp at boot time.
- Mount the usr file system.
- Determine whether to activate ldcache.
- Determine whether to start accounting, sadc, or network.

For additional information, see *General UNICOS System Administration*, Cray Research publication SG–2301.

3.4.8 To add site-specific code to the start-up process

To add your site-specific code to the start-up process, create the following executable files; each file will be executed at a specific point during the start-up process:

File	Description
`/etc/rc.pre`	If you must do some local work before the file systems are mounted, create this executable file. The `/etc/rc` script executes the `/etc/rc.pre` file after some initial preparatory work (checking whether or not security is configured, initializing start-up logging, and so on) but before any file systems are mounted.
`/etc/rc.mid`	If you must do some local work after the user file systems are mounted, but before any daemons (except the security log daemon, `/etc/slogdemon`) are started, create this executable file. It will be executed after mounting (and optionally caching) the user file systems, starting up `/etc/slogdemon`, and preserving interrupted `vi` or `ex` sessions.
`/etc/rc.pst`	If you must do some local work after everything has been started, create this executable file. It will be executed before the `/etc/rc` script exits and returns a `Console Login:` prompt to the system console. Given that networks will already have been configured and networking daemons will have been started, `/etc/rc.pst` is not necessarily executed before any users have logged in to the system from the network. To start local daemons, configure them into the `/etc/config/daemons` file and call `/etc/sdaemons` by using the daemon name.

3.5 Run-level configuration

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

convenient method of tailoring the system's resources to accommodate users' needs.

Two main modes of operation exist for the UNICOS system: 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.

3.5.1 Changing run level

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

`/etc/init` *level*

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

3.5.2 Strategies for using run levels

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

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

3.5.2.1 Single-user mode

Many system maintenance, modification, testing, configuration, and repair procedures are performed while the system is in single-user mode to protect

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

When the UNICOS system is in single-user mode, all network connections and hard-wired terminals are disabled, and only the console terminal can interact with the system. This mode of operation lets you make necessary changes to the system without doing any other processing. When the UNICOS system is in single-user mode, the # symbol (or `sn`*xxx*#) is the system prompt.

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

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

3.5.2.2 Multiuser mode

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

- Recording system start-up time in `/etc/wtmp`.
- Mounting all file systems required for normal system operation. This includes the regular system file systems (`/usr` and `/tmp`), the file system or systems that contain the home directories' `/tmp` file system of the system's users, and other file systems that contain files to which the users must have access.
- Removing any lock files that may interfere with normal system operation (for example, a lock file for a system daemon).

- Running daemons that provide various system services. The list may include, but is not restricted to, the following:
  - `errdemon`
  - `slogdemon` (for the UNICOS multilevel security (MLS) feature)
  - `cron`
  - `tapestart` (for online tapes)
  - `syslogd`
  - `nqsdaemon` (for NQS)
- Running the `netstart` script to initialize the system's TCP/IP network connections.
- Starting system accounting.
- Moving or truncating log files (for example, `/usr/lib/cron/log` or `/usr/spool/nqs/log`) to prevent them from growing without limits.
- Allowing users to log in.

3.5.2.3 Typical tasks you can perform while in multiuser mode

The following are some typical system administration tasks that you can perform while the UNICOS system 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 (see Chapter 5, page 51).
- 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` (see Chapter 8, page 217).

- 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 225. 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`.

3.5.2.4 Dedicated system

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

3.6 IOS prompts, and permissible actions

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 CRAY J90 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`.

3.6.1 IOS boot prompt

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 CRAY J90 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 CRAY J90 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 CRAY J90 system console disk and has the path name `/ios/ios`.
  - The IOS strategies and drivers reside on the CRAY J90 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.

3.6.2 IOS prompt

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 CRAY J90 system console disk for messages indicating that a slave IOS has panicked if this is suspected.
- The IOS and CRAY J90 mainframe are both up; `CONTROL-a` was pressed to change from the mainframe prompt to the IOS prompt.
- The CRAY J90 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*, Cray Research publication SQ–2172.

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 VMEbus, 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.

4.1 Related UNICOS system daemons documentation

The following documentation contains additional information about UNICOS system daemons: *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022, `bcheckrc`(8), `brc`(8), `dmdstop`(8), `fuser`(8), `init`(8), `msgdstop`(8), `rc`(8), `sdaemon`(8), and `shutdown`(8) man pages.

Procedure 3: 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 page Section 4.1, page 45).

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 -scommand, 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 chapter 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 your users have sufficient free space on their file systems to accomplish their work

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 56). 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 chapter will prove useful when you decide on a file system plan for your system.

Note: Although all UNICOS file systems have some common aspects, file system creation and organization varies on Cray Research systems. If you have Cray Research systems that are not CRAY J90 systems, see *General UNICOS System Administration*, Cray Research publication SG–2301, to determine differences in file systems and how to configure them.

5.2 Related file systems documentation

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

- *General UNICOS System Administration*, Cray Research publication SG–2301
- *UNICOS Resource Administration*, Cray Research publication SG–2302

- *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `df`(1), `du`(1), `mkdir`(1), and `rm`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `dir`(5), `dsk`(4), `fs`(5), `fstab`(5), `inode`(5), `ldd`(4), `mnttab`(5), and `pdd`(4) man pages
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `ddstat`(8), `diskusg`(8), `dmap`(8), `econfig`(8), `fsck`(8), `fsmap`(8), `fuser`(8), `labelit`(8), `mkfs`(8), `mknod`(8), `mount`(8), `stor`(8), and `umount`(8) man pages

5.3 An overview of file systems

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

File systems have 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 the UNICOS system. 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 the UNICOS system 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, 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 section provides terminology associated with file systems. Everything is viewed by the UNICOS system as a file, whether it is an ASCII file of user data or a physical disk device. The UNICOS system 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* hold user data of various formats. *Directory 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.

A *physical device* 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.

A *partition* is one slice on one physical device.

One or more slices create 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.

An *inode* contains information such as permissions and file size for all five types of files.

Regular files are composed of readable characters; these can include data, text, or program files that can be executed.

The following *special 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 J90 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.

All special files have a major and a minor device number associated with it. A *major device 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. A *minor device 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 J90 systems, minor number 253 is used for the `ce` partition. For additional information, see *General UNICOS System Administration*, Cray Research publication SG–2301.

All UNICOS special files are located in the `/dev` directory or one of its subdirectories. Your CRAY J90 system is initially supplied with sufficient UNICOS special files for most basic device configurations. You should create additional (block) special files to match your unique file system layout. All special files are created using the `mknod` command.

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

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

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

5.3.2 UNICOS file system structure

UNICOS file systems are often stored on several different physical devices. When you configure a file system, you first specify the physical locations that compose the file system. This information is stored in the `/sys/param` file,

and it is written using the menu system. You can store file systems on disk or in random-access memory (RAM), or a combination of both.

The definition of your system's logical and physical disk devices is defined in the `/sys/param` file on the IOS. 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 most important responsibilities of the system administrator is to monitor system disk usage and to ensure that the system's users have sufficient free space on their file systems to accomplish their work.

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

Command	Description
`/usr/lib/acct/diskusg`	Summarizes the disk usage on the file system you specify by file ownership and identifies users who are using most of the space on a file system. The `/usr/lib/acct/diskusg -h` command is the preferred command for summarizing disk use; the `-h` option provides headings.
`/etc/econfig`	The `-d` option prints out `mknod` commands to generate file systems. You may want to do this when you first get your system in case you have to manually recreate these nodes.
`/etc/dmap`	Displays information about the configuration of a disk subsystem.
`/etc/bmap`	Displays which file is using the block on a given file system.
`/etc/fsmap`	Displays file system free block layout.
`/bin/df`	Displays the number and percentage of free blocks available for mounted file systems; the `-p` option is particularly useful.

`/bin/du`	Summarizes the disk usage on a file system, by directory structure. The `-s` option provides the total number of disk blocks used under each directory (or file) specified.
`/etc/errpt`	Processes errors report generated by `errdaemon`. This UNICOS command is for disk hardware errors; `errpt -a` produces a detailed list of errors; `errpt -d` *device-type* produces list of errors for the specified device type.
`/etc/mount`	Displays the list of all currently mounted disk files and their mount points when issued without arguments.
`/etc/stor`	Sorts special files by physical device numbers and writes to standard output information about starting and ending disk addresses and sizes.
`/ce/bin/olhpa`	Displays hardware errors by reading the `/usr/adm/errfile` file. The `-d` option lets you view disk errors.
`/bin/fck`	Displays information concerning the names files that are gathered from reading the inode and the address blocks from the block special device in the `/dev/dsk` directory.
`/etc/ddstat`	Displays configuration information about disk type character and block special devices.
`/etc/pddconf`	Controls the state of a disk drive.
`/etc/pddstat`	Displays information from the disk table, which controls disk I/O.

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 file system minimum size requirements and 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 61, and for information about disk banding,

see Section 5.8, page 61.) You must consider disk capacity and transfer rate, as well as file system size and usage, along with the number of users and types of applications your Cray Research representative installed on your system.

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

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

For up-to-date information regarding minimum file system and amount of free blocks needed to install other Cray Research software products, see the *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271.

5.5.1 The `root (/)` file system

Size recommendations: You should define a **minimum** region of 110,000 blocks to hold your `root` file system.

If you have them, DD-4, DD-5I, DD-5S, or DD-6S disk drives are the preferred type of drive on which to configure the `root` (`/`) file system, with enhanced serial driver interface (ESDI) drives a second choice.

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.

5.5.2 The `/usr` file system

Size recommendations: You should define a **minimum** region of 190,000 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.

If you have them, DD-4, DD-5I, DD-5S, or DD-6S disk drives are the preferred type of drive on which to configure these two file systems, with ESDI drives a second choice.

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 CRAY J90 system is 120,000 blocks. This size is sufficient to hold all of the files necessary to relink the UNICOS kernel. You also must allow enough space in your default value to handle additional Cray Research asynchronous products you will load and use (for this information, see your UNICOS installation guide and related errata).

5.5.4 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*.

5.5.5 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. If possible, stripe the `swap` device across two DD-4, DD-5I, DD-5S, or DD-6S disks or across disks attached to separate ESDI controllers. For more information about striping, see Section 5.7, page 61, and *General UNICOS System Administration*, Cray Research publication SG–2301.

5.5.6 The dump device

Size recommendations: The **minimum** size of your `dump` device should be a little larger than the amount of memory you actually want to examine to allow an additional 1200 blocks for a `dump` header. You should start with a minimum of 50,000 blocks for the dump size. A dump entry must be in the logical device portion of the file system section of the `/sys/param` file. Take this into account when the range of memory you select to dump by using the `mfdump` command or the area you desired to dump will be truncated to preserve the dump header when the dump is written. The UNICOS system does not dump onto the swap device.

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

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

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

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

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

5.5.8 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 a table of disk device characteristics, see Appendix D, page 323.

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. 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 61.

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

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

5.9 Configuring your devices and their file system allocation

The system configuration file that configures disks 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. 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 `/sys/param` file that describe your system configuration.

The remainder of this section provides information and procedures to help you do the following:

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

5.9.1 Network disk array configuration

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

5.9.2 CSL syntax

Three classes of tokens make up the CSL: identifiers, constants, and operators/separators. White space (horizontal tabs, newlines, carriage returns, and spaces) separate individual tokens.

- An identifier is a sequence of digits and letters that specify either special keywords (such as `CONFIG`) or specific objects (such as a physical device). You can enclose the digits and characters in double quotation marks. The underscore (_) and dash (-) are interpreted as letters. Identifiers consist of letters, numbers, and the - and _ symbols. CSL identifiers are case sensitive. The first character may be any of the valid identifier characters.
  - Reserved disk type identifiers in CSL (descriptive comments are within brackets). `DD3`, `DDESDI`, `RD1`, and `DD4` are not supported disk type identifiers on CRAY J90 systems:

`DD3` (New ESDI)	`DD4` (IPI-2 Sabre-7)
`DDESDI` (Old ESDI)	`RD1` (Removable DD-3 ESDI)
`DDRAM` (RAM device)	`DD5I` (Buffered IPI)
`DDSTR` (Striped device)	`DD5S` (3-Gbyte SCSI drive)

`HD16` (HIPPI disk, 16-Kbyte sector)

`HD32` (HIPPI disk, 32-Kbyte sector)

`HD64` (HIPPI disk, 64-Kbyte sector)

`DD6S` (9-Gbyte SCSI drive)

`DD314` (4-Gbyte drive)

`DD318` (8-Gbyte SCSI drive)

- For a list of additional keyword identifiers that have special meaning, see the *UNICOS Configuration Administrator's Guide*, Cray Research publication SG–2303.

- Constants : All constants are positive integers. If a constant begins with a 0, octal format is assumed; otherwise, decimal format applies. The use of digits 8 or 9 in an octal constant causes an error.
- Operators/separators: { } ; ,
- Comments : Words within the paired /* */ symbols are comments.

5.9.3 Placement of CSL statements

All CSL statements that define the size, location, and other attributes of your UNICOS file systems are found in the `/sys/param` file. The placement and order of your configuration and CSL statements are important.

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

CSL statements must conform to the following requirements:

- All CSL statements must be terminated by a semicolon, and all section definitions must be placed within one pair of braces { } or ("curly brackets").
- The UNICOS system processes CSL statements in order of appearance in the IOS parameter file.
- Your IOS parameter file must begin with the keywords `revision sn` *xxxx*; *xxxx* is your machine's serial number.
- The first section of CSL statements defines IOS information.
- The second section of CSL statements defines the mainframe information.
- The third section of CSL statements defines the UNICOS configuration information.

- The fourth section of CSL statements defines the file systems information, which includes configuration statements for the physical devices, logical devices, and special system devices.
- The fifth section of CSL statements defines the network configuration for your system.
- The file must end with the closing brace.

The following sections describe the parameter file sections; a sample file is included.

5.9.3.1 Revision section

The revision section marks the configuration file with a site-defined name for identification purposes, particularly for programs and other Cray Research products. The revision section is specified in the parameter file by the following statement and is designated as `revision sn` *xxxx*; *xxxx* is your machine's serial number.

5.9.3.2 ios_v section

This statement section sets the number of IOSs configured. Using the UNICOS Installation / Configuration Menu System (ICMS), you specify the characteristics on the

```
Configure System
    ->IOS Configuration
```

submenu.

This section should include the following:

Statement	Description
`cluster` *n*	The *n* argument is the IOS; the first entry should be for cluster 0.
`miop`	Required keyword for the cluster.
`eiop 0`	Required keywords for the cluster.

`eiop` *xx*	Designates the controller numbers for the disks associated with the specific IOS.

5.9.3.3 Mainframe section

The mainframe section defines the number of CPUs, size of the mainframe memory, and channel information. Using the ICMS, you specify the characteristics on the

```
Configure System
    ->Mainframe Hardware Characteristics
```

submenu.

This section should include the following:

Statement	Description
value `cpus`	The number of CPUs.
value units `memory`	The *units* may be either words or Mwords; *value* is typically set to the physical amount of memory in the machine.
`channel` *value*	(See Table 1, page 65) The *value* of the master IOS is always 20 or 020 (octal). The channel numbers of the slave IOSs depend on the CPU to which they are connected. To define the channels that connect to the slave IOSs, use the `channel` keyword.

Table 1. CRAY J90 IOS Channel Values

Processor Module	IOS Channel Value
0	020 022 024 026
1	030 032 034 036
2	040 042 044 046
3	050 052 054 056
4	060 062 064 066
5	070 072 074 076

Processor Module	IOS Channel Value
6	100 102 104 106
7	110 112 114 116

5.9.3.4 UNICOS section

The UNICOS section sets certain tunable parameters in the `var` structure. You set these parameters in the

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

submenu. For more information on this topic, you should read *General UNICOS System Administration*, Cray Research publication SG–2301, to determine what parameters you might want to change.

5.9.3.5 File system section

The file system section includes the following:

- Description of physical devices in the system
- Description of logical devices (device nodes) in the system
- Identification of the root and swap devices

The following sections describe each portion of the file system section.

5.9.3.5.1 Physical device definition

The physical devices are defined in this portion of the file system section with the following syntax; lines that begin with `pdd` define the slices for the device:

```
disk name{type type; iopath { cluster number; eiop number;
   channel number;} unit number;pdd slice{minor number; sector
   measure; length number units;}}
```

You must define each field:

Field	Description
`disk` *name*	The *name* of each device is site-configurable, but it must be unique among all devices. By convention, the format is composed of the disk type, IOS number, controller number, and unit (for example, `S_0200`).
`type` *type*	Defines the disk type (see Table 2, page 67).
`cluster` *number*	Defines the IOS. Specifies the IOS number; no default.
`eiop` *number*	Defines the controller (see Table 2, page 67).
`channel` *number*	Defines the channel number.
`unit` *number*	Defines the unit attached to the controller (see Table 2, page 67).
`pdd` *slice*	Each *slice* name also is site-configurable, but it must be unique among all devices. Use a meaningful naming scheme for your file system slices. You may want to name the slices with a combination of a reference to the file system of which they are a part, and a unique number. Names of slices must be unique and consist of 8 characters or fewer.
`minor` *number*	Must be unique; you should use numbers in ascending order.
`sector` *measure*	For all disks used with CRAY J90 systems, *measure* is the starting block number of the slice.
`length` *number units*	For all disks used with CRAY J90 systems, *number units* is the number of blocks for the slice.

Table 2. Disk device types and their values

Disk type keyword	Disk device type number	`eiop` value range	Total blocks per disk unit	Maximum number of units
`DDESDI`	64	0 - 7	170,100	4 (0 - 3)
`DDLDAS`	66	8	1,269,114	-

Disk type keyword	Disk device type number	`eiop` value range	Total blocks per disk unit	Maximum number of units
`DD3`	65	0-7	334,200	4 (0 - 3)
`DDAS2`	67	8	2,502,000	-
`DD4`	68	10-17	653,000	2 (0 or 1)
`DD5S`	71	20-27	781,000	4 (0 - 3)
`DD5I`	72	30-37	723,000	4 (0 - 3)
`RD1`	69	0-7	334,200	-
`DDRAM`	104			
`DD6S`	74	20-27	2,389,000	0-3
`DD314`	75	20-27	1,102,000	0-3
`DD318`	76	20-27		0-3

Note: For HIPPI disk types `HD16`, `HD32`, and `HD64`, the capacity is not fixed according to the device type; the size depends on the specific disk device model.

5.9.3.5.2 Logical device definition

Device nodes (logical device groups) are defined using this portion of the file system section. A logical device groups one or more previously defined physical device slices. Each file system you configure has a corresponding logical device entry. Logical device entries always follow the complete set of physical device statements in the `/sys/param` file.

Each logical device is defined using the following syntax; lines that begin with `ldd` define the logical device:

`ldd` *name* `{minor` *number*`;` *device slice;* `}`

The fields are defined as follows:

Field	Description
`ldd` *name*	Consists of up to 8 characters. By convention, the name is lowercase and reflects the name of the special file that will be created automatically during UNICOS multiuser startup. Each `ldd` *name* shows up as a file name in the UNICOS

	`/dev/dsk` directory. The `ldd` *name* portion must be unique for each logical device.
`minor` *number*	Must be unique; you should use numbers in ascending order.
device slice	The previously defined physical slice name that describes this logical device.

5.9.3.5.3 Special system devices

This portion of the file system section defines the system devices. The `rootdev` and `swapdev` definitions are required. The `swapdev` is **not** a file system, it is an area of disk reserved for swapping activity.

The `rootdev` and `swapdev` definitions have the following syntax:

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

Again, each slice (`ldd` *name*) must be the name of a logical device previously defined in the `/sys/param` file.

5.9.3.5.4 Network section

The network section defines network devices and network parameters. You can configure low- and high-speed network communication devices in the

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

menu. The network section includes the following information:

- Descriptions of network parameters
- Descriptions of each specific network device that uses standard templates or customized prototypes
- Customized network device prototypes

The network section is specified in the parameter file by the following statement syntax:

```
{ network parameters network number {            iopath {
  cluster number;                        eiop number;
  channel value;              } }
```

The *network* can be `endev` for Ethernet or `fddev` for FDDI. The channel *value* for Ethernet will always begin with 020, and it also can be 021, 022, and 023, depending on the number of channels. The channel *value* for FDDI will always begin with 040, and it also can be 041, depending on the number of channels.

Each Ethernet and each FDDI connection to your system should have one network statement.

5.10 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, 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.

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

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

Remove the existing devices by using the following commands:

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

Generate the new device definitions by using the following commands:

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

A sample CRAY J90 `/sys/param` configuration file follows.

```
revision "SN9003.20";

ios_e {
/* SN9003 - param.ios_e - Edition 3 [Wed Aug 20 14:37:31 CDT 1997] */
cluster 0 {
        miop; eiop 0; eiop 20; eiop 21; eiop 24; eiop 25; eiop 30;

}
cluster 1 {
        miop; eiop 0; eiop 20; eiop 21; eiop 30;

}
cluster 2 {
        miop; eiop 0; eiop 20; eiop 21; eiop 22; eiop 23;

}
cluster 3 {
        miop; eiop 0; eiop 20; eiop 21; eiop 22; eiop 23;

}
}

mainframe {
/* SN9003 - param.mf.hardware - Edition 11 [Tue Aug 26 18:36:41 CDT 1997] */
16 cpus;
512 Mwords memory;
channel 020 is lowspeed to cluster 0;
channel 030 is lowspeed to cluster 1;
channel 050 is lowspeed to cluster 2;
channel 052 is lowspeed to cluster 3;
channel 78 is lowspeed to pseudo TCP;
}

unicos {
/* SN9003 - param.unicos - Edition 6 [Thu Aug 21 11:31:44 CDT 1997] */
 5000 NBUF;
  200 NPBUF;
98280 LDCHCORE;
```

```
 5000 NLDCH;
  256 PDDMAX;
  256 LDDMAX;
   32 HDDMAX;
  300 PDDSLMAX;
    8 MDDSLMAX;
    8 SDDSLMAX;
    8 RDDSLMAX;
    4 SSDDSLMAX;
   64 HDDSLMAX;
    0 GUESTMAX;
294912 TAPE_MAX_PER_DEV;
    8 TAPE_MAX_CONF_UP;
   16 TAPE_MAX_DEV;
}

filesystem {
/* SN9003 - param.fs.disks - Edition 20 [Tue Sep 16 11:36:54 CDT 1997] */
/*
 *      Physical device configuration
 */
disk "02020.0" {
        type DD5S;
        iopath {
            cluster 0;
            eiop 20;
            channel 020;
        }
        unit 0;
        pdd 0s00_root_b {
            minor  3;
            sector  0;
            length 240000 sectors;
        }
        pdd 0s00_root_d {
            minor  4;
            sector  240000;
            length 240000 sectors;
        }
        pdd 0s00_OPEN {
            minor  5;
            sector  480000;
            length 41000 sectors;
```

```
        }
        pdd 0s00_usr_a {
            minor  6;
            sector  521000;
            length 260000 sectors;
        }
}
disk "02020.1" {
        type DD5S;
        iopath {
            cluster 0;
            eiop 20;
            channel 020;
        }
        unit 1;
        pdd 0s01_opt {
            minor  7;
            sector  0;
            length 781000 sectors;
        }
}
disk "02120.0" {
        type DD5S;
        iopath {
            cluster 0;
            eiop 21;
            channel 020;
        }
        unit 0;
        pdd 0s10_root_f {
            minor  8;
            sector  0;
            length 240000 sectors;
        }
        pdd 0s10_usr_b {
            minor  9;
            sector  240000;
            length 260000 sectors;
        }
        pdd 0s10_usr_d {
            minor  10;
            sector  500000;
            length 260000 sectors;
```

```
        }
        pdd 0s10_dmjrnl {
            minor  11;
            sector  760000;
            length 21000 sectors;
        }
}
disk "02120.1" {
        type DD5S;
        iopath {
            cluster 0;
            eiop 21;
            channel 020;
        }
        unit 1;
        pdd 0s11_root_e {
            minor  13;
            sector  0;
            length 240000 sectors;
        }
        pdd 0s11_root_a {
            minor  17;
            sector  240000;
            length 240000 sectors;
        }
        pdd 0s11_root_g {
            minor  18;
            sector  480000;
            length 240000 sectors;
        }
        pdd 0s11_OPEN {
            minor  20;
            sector  720000;
            length 61000 sectors;
        }
}
disk "02420.0" {
        type DD6S;
        iopath {
            cluster 0;
            eiop 24;
            channel 020;
        }
```

```
        unit 0;
        pdd 0s40_usr_e {
            minor  21;
            sector  0;
            length 260000 sectors;
        }
        pdd 0s40_OPEN {
            minor  22;
            sector  260000;
            length 203698 sectors;
        }
        pdd 0s40_u01 {
            minor  85;
            sector  463698;
            length 500013 sectors;
        }
        pdd 0s40_u23 {
            minor  74;
            sector  963711;
            length 500013 sectors;
        }
        pdd 0s40_ckpt {
            minor  53;
            sector  1463724;
            length 925276 sectors;
        }
}
disk "02420.1" {
        type DD6S;
        iopath {
            cluster 0;
            eiop 24;
            channel 020;
        }
        unit 1;
        pdd 0s41_root_utna {
            minor  23;
            sector  0;
            length 240000 sectors;
        }
        pdd 0s41_usr_ISV {
            minor  24;
            sector  240000;
```

```
            length 260000 sectors;
        }
        pdd 0s41_OPEN {
            minor  26;
            sector  500000;
            length 595858 sectors;
        }
        pdd 0s41_spool {
            minor  87;
            sector  1095858;
            length 367866 sectors;
        }
        pdd 0s41_root_h {
            minor  27;
            sector  1463724;
            length 240000 sectors;
        }
        pdd 0s41_dmspool {
            minor  29;
            sector  1703724;
            length 115276 sectors;
        }
        pdd 0s41_mail {
            minor  124;
            sector  1819000;
            length 70000 sectors;
        }
        pdd 0s41_u67 {
            minor  125;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "02520.0" {
        type DD6S;
        iopath {
            cluster 0;
            eiop 25;
            channel 020;
        }
        unit 0;
        pdd 0s50_root_j {
            minor  30;
```

```
            sector  0;
            length 240000 sectors;
        }
        pdd 0s50_dmfc1 {
            minor  142;
            sector  240000;
            length 37283 sectors;
        }
        pdd 0s50_dmfc2 {
            minor  143;
            sector  277283;
            length 37283 sectors;
        }
        pdd 0s50_dmfc3 {
            minor  144;
            sector  314566;
            length 37283 sectors;
        }
        pdd 0s50_dmfc4 {
            minor  145;
            sector  351849;
            length 37283 sectors;
        }
        pdd 0s50_dmfc5 {
            minor  146;
            sector  389132;
            length 37283 sectors;
        }
        pdd 0s50_dmfc6 {
            minor  147;
            sector  426415;
            length 37283 sectors;
        }
        pdd 0s50_ptmp {
            minor  89;
            sector  463698;
            length 1925302 sectors;
        }
}
disk "02520.1" {
        type DD6S;
        iopath {
            cluster 0;
```

```
    eiop 25;
    channel 020;
}
unit 1;
pdd 0s51_src_a {
    minor  32;
    sector  0;
    length 650000 sectors;
}
pdd 0s51_OPEN {
    minor  33;
    sector  650000;
    length 146104 sectors;
}
pdd 0s51_usr_adm {
    minor  92;
    sector  796104;
    length 299754 sectors;
}
pdd 0s51_usr_i {
    minor  34;
    sector  1095858;
    length 260000 sectors;
}
pdd 0s51_OPENa {
    minor  35;
    sector  1355858;
    length 107866 sectors;
}
pdd 0s51_core {
    minor  94;
    sector  1463724;
    length 301716 sectors;
}
pdd 0s51_OPENb {
    minor  141;
    sector  1765440;
    length 123560 sectors;
}
pdd 0s51_u89 {
    minor  129;
    sector  1889000;
    length 500000 sectors;
```

```
        }
}
disk "03020.0" {
        type DD5I;
        iopath {
            cluster 0;
            eiop 30;
            channel 020;
        }
        unit 0;
        pdd 0b00_swap {
            minor  41;
            sector  0;
            length 723000 sectors;
        }
}
disk "03020.1" {
        type DD5I;
        iopath {
            cluster 0;
            eiop 30;
            channel 020;
        }
        unit 1;
        pdd 0b01_tmp {
            minor  42;
            sector  0;
            length 723000 sectors;
        }
}
disk "03020.2" {
        type DD5I;
        iopath {
            cluster 0;
            eiop 30;
            channel 020;
        }
        unit 2;
        pdd 0b02_tmp {
            minor  51;
            sector  0;
            length 723000 sectors;
        }
```

```
}
disk "03020.3" {
        type DD5I;
        iopath {
            cluster 0;
            eiop 30;
            channel 020;
        }
        unit 3;
        pdd 0b03_tmp {
            minor  52;
            sector  0;
            length 723000 sectors;
        }
}
disk "12030.0" {
        type DD6S;
        iopath {
            cluster 1;
            eiop 20;
            channel 030;
        }
        unit 0;
        pdd 1s00_swap {
            minor  82;
            sector  0;
            length 463698 sectors;
        }
        pdd 1s00_OPEN {
            minor  36;
            sector  463698;
            length 632160 sectors;
        }
        pdd 1s00_mbin {
            minor  96;
            sector  1095858;
            length 469467 sectors;
        }
        pdd 1s00_admin {
            minor  97;
            sector  1565325;
            length 116865 sectors;
        }
```

```
        pdd 1s00_OPENa {
            minor  140;
            sector  1682190;
            length 206810 sectors;
        }
        pdd 1s00_u45 {
            minor  75;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "12030.1" {
        type DD6S;
        iopath {
            cluster 1;
            eiop 20;
            channel 030;
        }
        unit 1;
        pdd 1s01_root_upt {
            minor  37;
            sector  0;
            length 240000 sectors;
        }
        pdd 1s01_OPEN {
            minor  39;
            sector  240000;
            length 223698 sectors;
        }
        pdd 1s01_u45 {
            minor  76;
            sector  463698;
            length 500013 sectors;
        }
        pdd 1s01_u67 {
            minor  77;
            sector  963711;
            length 500013 sectors;
        }
        pdd 1s01_ckpt {
            minor  54;
            sector  1463724;
            length 925276 sectors;
```

```
        }
}
disk "12130.0" {
        type DD6S;
        iopath {
            cluster 1;
            eiop 21;
            channel 030;
        }
        unit 0;
        pdd 1s10_swap {
            minor  101;
            sector  0;
            length 463698 sectors;
        }
        pdd 1s10_usr_k {
            minor  40;
            sector  463698;
            length 260000 sectors;
        }
        pdd 1s10_usr_dm {
            minor  43;
            sector  723698;
            length 72406 sectors;
        }
        pdd 1s10_dump {
            minor  78;
            sector  796104;
            length 156114 sectors;
        }
        pdd 1s10_u89 {
            minor  79;
            sector  952218;
            length 500013 sectors;
        }
        pdd 1s10_OPENa {
            minor  44;
            sector  1452231;
            length 176769 sectors;
        }
        pdd 1s10_usr_utna {
            minor  45;
            sector  1629000;
```

```
            length 260000 sectors;
        }
        pdd 1s10_u67 {
            minor  126;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "12130.1" {
        type DD6S;
        iopath {
            cluster 1;
            eiop 21;
            channel 030;
        }
        unit 1;
        pdd 1s11_OPEN {
            minor  46;
            sector  0;
            length 463698 sectors;
        }
        pdd 1s11_ptmp {
            minor  104;
            sector  463698;
            length 1925302 sectors;
        }
}
disk "13030.0" {
        type DD5I;
        iopath {
            cluster 1;
            eiop 30;
            channel 030;
        }
        unit 0;
        pdd 1b00_swap {
            minor  61;
            sector  0;
            length 723000 sectors;
        }
}
disk "13030.1" {
        type DD5I;
```

```
        iopath {
            cluster 1;
            eiop 30;
            channel 030;
        }
        unit 1;
        pdd 1b01_tmp {
            minor  62;
            sector  0;
            length 723000 sectors;
        }
}
disk "13030.2" {
        type DD5I;
        iopath {
            cluster 1;
            eiop 30;
            channel 030;
        }
        unit 2;
        pdd 1b02_tmp {
            minor  71;
            sector  0;
            length 723000 sectors;
        }
}
disk "13030.3" {
        type DD5I;
        iopath {
            cluster 1;
            eiop 30;
            channel 030;
        }
        unit 3;
        pdd 1b03_tmp {
            minor  72;
            sector  0;
            length 723000 sectors;
        }
}
disk "22050.0" {
        type DD6S;
        iopath {
```

```
            cluster 2;
            eiop 20;
            channel 050;
        }
        unit 0;
        pdd 2s00_swap {
            minor  55;
            sector  0;
            length 463698 sectors;
        }
        pdd 2s00_utmp {
            minor  25;
            sector  463698;
            length 1301742 sectors;
        }
        pdd 2s00_usr_sl {
            minor  47;
            sector  1765440;
            length 123560 sectors;
        }
        pdd 2s00_u01 {
            minor  111;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "22050.1" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 20;
            channel 050;
        }
        unit 1;
        pdd 2s01_src_e {
            minor  48;
            sector  0;
            length 650000 sectors;
        }
        pdd 2s01_usr_upt {
            minor  49;
            sector  650000;
            length 260000 sectors;
```

```
        }
        pdd 2s01_root_k {
            minor  50;
            sector  910000;
            length 240000 sectors;
        }
        pdd 2s01_ram_root {
            minor  57;
            sector  1150000;
            length 14750 sectors;
        }
        pdd 2s01_root_c {
            minor  14;
            sector  1164750;
            length 240000 sectors;
        }
        pdd 2s01_OPEN {
            minor  58;
            sector  1404750;
            length 984250 sectors;
        }
}
disk "22150.0" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 21;
            channel 050;
        }
        unit 0;
        pdd 2s10_swap {
            minor  28;
            sector  0;
            length 463698 sectors;
        }
        pdd 2s10_src_f {
            minor  59;
            sector  463698;
            length 650000 sectors;
        }
        pdd 2s10_src_g {
            minor  60;
            sector  1113698;
```

```
            length 650000 sectors;
        }
        pdd 2s10_OPEN {
            minor  63;
            sector  1763698;
            length 125302 sectors;
        }
        pdd 2s10_u23 {
            minor  113;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "22150.1" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 21;
            channel 050;
        }
        unit 1;
        pdd 2s11_OPEN {
            minor  64;
            sector  0;
            length 463698 sectors;
        }
        pdd 2s11_ptmp {
            minor  65;
            sector  463698;
            length 1925302 sectors;
        }
}
disk "22250.0" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 22;
            channel 050;
        }
        unit 0;
        pdd 2s20_src_b {
            minor  66;
            sector  0;
```

```
            length 650000 sectors;
        }
        pdd 2s20_src_d {
            minor  67;
            sector  650000;
            length 650000 sectors;
        }
        pdd 2s20_OPEN {
            minor  68;
            sector  1300000;
            length 589000 sectors;
        }
        pdd 2s20_u89 {
            minor  115;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "22250.1" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 22;
            channel 050;
        }
        unit 1;
        pdd 2s21_src_h {
            minor  69;
            sector  0;
            length 650000 sectors;
        }
        pdd 2s21_src_i {
            minor  70;
            sector  650000;
            length 650000 sectors;
        }
        pdd 2s21_src_j {
            minor  73;
            sector  1300000;
            length 650000 sectors;
        }
        pdd 2s21_OPEN {
            minor  80;
```

```
            sector  1950000;
            length 439000 sectors;
        }
}
disk "22350.0" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 23;
            channel 050;
        }
        unit 0;
        pdd 2s30_OPEN {
            minor  88;
            sector  0;
            length 1889000 sectors;
        }
        pdd 2s30_u45 {
            minor  117;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "22350.1" {
        type DD6S;
        iopath {
            cluster 2;
            eiop 23;
            channel 050;
        }
        unit 1;
        pdd 2s31_src_k {
            minor  81;
            sector  0;
            length 650000 sectors;
        }
        pdd 2s31_src_upt {
            minor  83;
            sector  650000;
            length 650000 sectors;
        }
        pdd 2s31_src_ISV {
            minor  84;
```

```
            sector  1300000;
            length 650000 sectors;
        }
        pdd 2s31_OPEN {
            minor  86;
            sector  1950000;
            length 439000 sectors;
        }
}
disk "32052.0" {
        type DD6S;
        iopath {
            cluster 3;
            eiop 20;
            channel 052;
        }
        unit 0;
        pdd 3s00_swap {
            minor  90;
            sector  0;
            length 463698 sectors;
        }
        pdd 3s00_utmp {
            minor  38;
            sector  463698;
            length 1301742 sectors;
        }
        pdd 3s00_usr_sl {
            minor  91;
            sector  1765440;
            length 123560 sectors;
        }
        pdd 3s00_u01 {
            minor  119;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "32052.1" {
        type DD6S;
        iopath {
            cluster 3;
            eiop 20;
```

```
    channel 052;
}
unit 1;
pdd 3s01_root_i {
    minor  93;
    sector  0;
    length 240000 sectors;
}
pdd 3s01_usr_f {
    minor  95;
    sector  240000;
    length 260000 sectors;
}
pdd 3s01_usr_g {
    minor  98;
    sector  500000;
    length 260000 sectors;
}
pdd 3s01_usr_h {
    minor  99;
    sector  760000;
    length 260000 sectors;
}
pdd 3s01_usr_j {
    minor  100;
    sector  1020000;
    length 260000 sectors;
}
pdd 3s01_root_ISV {
    minor  102;
    sector  1280000;
    length 240000 sectors;
}
pdd 3s01_src_utna {
    minor  103;
    sector  1520000;
    length 650000 sectors;
}
pdd 3s01_OPEN {
    minor  105;
    sector  2170000;
    length 219000 sectors;
}
```

```
}
disk "32152.0" {
        type DD6S;
        iopath {
            cluster 3;
            eiop 21;
            channel 052;
        }
        unit 0;
        pdd 3s10_swap {
            minor  106;
            sector  0;
            length 463698 sectors;
        }
        pdd 3s10_utmp {
            minor  107;
            sector  463698;
            length 1301742 sectors;
        }
        pdd 3s10_OPEN {
            minor  108;
            sector  1765440;
            length 123560 sectors;
        }
        pdd 3s10_u23 {
            minor  121;
            sector  1889000;
            length 500000 sectors;
        }
}
disk "32152.1" {
        type DD6S;
        iopath {
            cluster 3;
            eiop 21;
            channel 052;
        }
        unit 1;
        pdd 3s11_usr_c {
            minor  15;
            sector  0;
            length 260000 sectors;
        }
```

```
        pdd 3s11_OPEN {
            minor  109;
            sector  260000;
            length 2129000 sectors;
        }
}
disk "32252.0" {
        type DD5S;
        iopath {
            cluster 3;
            eiop 22;
            channel 052;
        }
        unit 0;
        pdd 3s20_src_c {
            minor  16;
            sector  0;
            length 650000 sectors;
        }
        pdd 3s20_OPEN {
            minor  114;
            sector  650000;
            length 131000 sectors;
        }
}
disk "32252.1" {
        type DD5S;
        iopath {
            cluster 3;
            eiop 22;
            channel 052;
        }
        unit 1;
        pdd 3s21_CS_scr {
            minor  116;
            sector  0;
            length 771000 sectors;
        }
        pdd 3s21_tng_1 {
            minor  118;
            sector  771000;
            length 10000 sectors;
        }
```

```
}
disk "32352.0" {
        type DD5S;
        iopath {
            cluster 3;
            eiop 23;
            channel 052;
        }
        unit 0;
        pdd 3s30_CS_scr {
            minor  120;
            sector  0;
            length 771000 sectors;
        }
        pdd 3s30_tng_2 {
            minor  122;
            sector  771000;
            length 10000 sectors;
        }
}
disk "32352.1" {
        type DD5S;
        iopath {
            cluster 3;
            eiop 23;
            channel 052;
        }
        unit 1;
        pdd 3s31_OPEN {
            minor  123;
            sector  0;
            length 781000 sectors;
        }
}
/*
 * Logical devices
 */

sdd bswap {
        minor 2;
        pdd 0b00_swap;
        pdd 1b00_swap;
}
```

```
sdd vbswap {
        minor 3;
        pdd 1s00_swap;
        pdd 2s00_swap;
        pdd 3s00_swap;
        pdd 1s10_swap;
        pdd 2s10_swap;
        pdd 3s10_swap;
}
sdd usr_sl {
        minor 4;
        pdd 3s00_usr_sl;
        pdd 2s00_usr_sl;
}

ldd swap {
        minor 1;
        sdd bswap;
        sdd vbswap;
}
ldd admin {
        minor 46;
        pdd 1s00_admin;
}
ldd ckpt {
        minor 70;
        pdd 0s40_ckpt;
        pdd 1s01_ckpt;
}
ldd core {
        minor 39;
        pdd 0s51_core;
}
ldd dm_jrnl {
        minor 28;
        pdd 0s10_dmjrnl;
}
ldd dmspool {
        minor 29;
        pdd 0s41_dmspool;
}
ldd dump {
        minor 40;
```

```
        pdd 1s10_dump;
}
ldd mail {
        minor 55;
        pdd 0s41_mail;
}
ldd mbin {
        minor 48;
        pdd 1s00_mbin;
}
ldd opt {
        minor 96;
        pdd 0s01_opt;
}
ldd ptmp {
        minor 56;
        pdd 0s50_ptmp;
        pdd 1s11_ptmp;
        pdd 2s11_ptmp;
}
ldd ram_root {
        minor 57;
        pdd 2s01_ram_root;
}
ldd root_a {
        minor 17;
        pdd 0s11_root_a;
}
ldd root_b {
        minor 3;
        pdd 0s00_root_b;
}
ldd root_c {
        minor 14;
        pdd 2s01_root_c;
}
ldd root_d {
        minor 4;
        pdd 0s00_root_d;
}
ldd root_e {
        minor 13;
        pdd 0s11_root_e;
```

```
}
ldd root_f {
        minor 8;
        pdd 0s10_root_f;
}
ldd root_g {
        minor 18;
        pdd 0s11_root_g;
}
ldd root_h {
        minor 27;
        pdd 0s41_root_h;
}
ldd root_i {
        minor 93;
        pdd 3s01_root_i;
}
ldd root_j {
        minor 30;
        pdd 0s50_root_j;
}
ldd root_k {
        minor 50;
        pdd 2s01_root_k;
}
ldd root_upt {
        minor 37;
        pdd 1s01_root_upt;
}
ldd root_ISV {
        minor 102;
        pdd 3s01_root_ISV;
}
ldd root_utna {
        minor 23;
        pdd 0s41_root_utna;
}
ldd spool {
        minor 43;
        pdd 0s41_spool;
}
ldd src_a {
        minor 32;
```

```
        pdd 0s51_src_a;
}
ldd src_b {
        minor 66;
        pdd 2s20_src_b;
}
ldd src_c {
        minor 16;
        pdd 3s20_src_c;
}
ldd src_d {
        minor 67;
        pdd 2s20_src_d;
}
ldd src_e {
        minor 130;
        pdd 2s01_src_e;
}
ldd src_f {
        minor 59;
        pdd 2s10_src_f;
}
ldd src_g {
        minor 60;
        pdd 2s10_src_g;
}
ldd src_h {
        minor 69;
        pdd 2s21_src_h;
}
ldd src_i {
        minor 140;
        pdd 2s21_src_i;
}
ldd src_j {
        minor 73;
        pdd 2s21_src_j;
}
ldd src_k {
        minor 81;
        pdd 2s31_src_k;
}
ldd src_upt {
```

```
        minor 83;
        pdd 2s31_src_upt;
}
ldd src_ISV {
        minor 84;
        pdd 2s31_src_ISV;
}
ldd src_utna {
        minor 103;
        pdd 3s01_src_utna;
}
ldd tmp {
        minor 91;
        pdd 0b01_tmp;
        pdd 1b01_tmp;
        pdd 0b02_tmp;
        pdd 1b02_tmp;
        pdd 0b03_tmp;
        pdd 1b03_tmp;
}
ldd tng_1 {
        minor 118;
        pdd 3s21_tng_1;
}
ldd tng_2 {
        minor 122;
        pdd 3s30_tng_2;
}
ldd u01 {
        minor 123;
        pdd 2s00_u01;
        pdd 3s00_u01;
        pdd 0s40_u01;
}
ldd u23 {
        minor 124;
        pdd 2s10_u23;
        pdd 3s10_u23;
        pdd 0s40_u23;
}
ldd u45 {
        minor 125;
        pdd 1s00_u45;
```

```
        pdd 2s30_u45;
        pdd 1s01_u45;
}
ldd u67 {
        minor 126;
        pdd 0s41_u67;
        pdd 1s10_u67;
        pdd 1s01_u67;
}
ldd u89 {
        minor 127;
        pdd 0s51_u89;
        pdd 2s20_u89;
        pdd 1s10_u89;
}
ldd usr_adm {
        minor 45;
        pdd 0s51_usr_adm;
}
ldd usr_a {
        minor 6;
        pdd 0s00_usr_a;
}
ldd usr_b {
        minor 9;
        pdd 0s10_usr_b;
}
ldd usr_c {
        minor 15;
        pdd 3s11_usr_c;
}
ldd usr_dm {
        minor 53;
        pdd 1s10_usr_dm;
}
ldd usr_d {
        minor 10;
        pdd 0s10_usr_d;
}
ldd usr_e {
        minor 21;
        pdd 0s40_usr_e;
}
```

```
ldd usr_f {
        minor 132;
        pdd 3s01_usr_f;
}
ldd usr_g {
        minor 98;
        pdd 3s01_usr_g;
}
ldd usr_h {
        minor 99;
        pdd 3s01_usr_h;
}
ldd usr_i {
        minor 34;
        pdd 0s51_usr_i;
}
ldd usr_j {
        minor 100;
        pdd 3s01_usr_j;
}
ldd usr_k {
        minor 128;
        pdd 1s10_usr_k;
}
ldd usr_upt {
        minor 49;
        pdd 2s01_usr_upt;
}
ldd usr_ISV {
        minor 24;
        pdd 0s41_usr_ISV;
}
ldd usr_sl {
        minor 47;
        sdd usr_sl;
}
ldd usr_utna {
        minor 129;
        pdd 1s10_usr_utna;
}
ldd utmp {
        minor 95;
        pdd 3s00_utmp;
```

```
        pdd 2s00_utmp;
        pdd 3s10_utmp;
}
ldd dmfclass1 {
        minor 142;
        pdd 0s50_dmfc1;
}
ldd dmfclass2 {
        minor 143;
        pdd 0s50_dmfc2;
}
ldd dmfclass3 {
        minor 144;
        pdd 0s50_dmfc3;
}
ldd dmfclass4 {
        minor 145;
        pdd 0s50_dmfc4;
}
ldd dmfclass5 {
        minor 146;
        pdd 0s50_dmfc5;
}
ldd dmfclass6 {
        minor 147;
        pdd 0s50_dmfc6;
}
/* SN9003 - param.fs.special - Edition 3 [Wed Aug 20 14:37:53 CDT 1997] */
rootdev is ldd root_c;
swapdev is ldd swap;
dmpdev is ldd dump;
}

network {
/* SN9003 - param.network - Edition 6 [Thu Aug 21 11:31:52 CDT 1997] */
    8 nfs_static_clients;
    8 nfs_temp_clients;
    8 cnfs_static_clients;
    8 cnfs_temp_clients;
32768 nfs_maxdata;
  256 nfs_num_rnodes;
 1200 nfs_maxdupreqs;
    3 nfs_duptimeout;
```

```
    0 nfs_printinter;
16000 tcp_nmbspace;
    2 himaxdevs;
    4 himaxpaths;
    1 fdmaxdevs;
    0 npmaxdevs;
    1 enmaxdevs;
    2 atmmaxdevs;
131072 atmarp_recv;
65536 atmarp_send;
 1024 atmarp_entries;

0755 hidirmode;
0666 hifilemode;

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

fddev   0 {
        iopath {
            cluster 2;
            eiop 0;
            channel 040;
        }
}
atmdev  0 {
        iopath {
            cluster 0;
            eiop 0;
            channel 020;
        }
}
atmdev  1 {
        iopath {
            cluster 3;
            eiop 0;
            channel 020;
        }
```

```
}
}
```

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

Note: To complete this procedure, you must be super user; you will see the `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:

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
```

1. Select the following menu:

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

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

Section 5.9.2, page 62, describes the sections of the `/sys/param` file.

A sample menu screen follows:

```
                        Disk Configuration

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

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

- The `/etc/pddstat` command displays the name of the device, its type, and whether it is up or down.
- 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 J90 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 00 0 0 0
       /dev/pdd/tmp_2 c 32/96 12 01036020 0 201600 00 0 0 1
       /dev/pdd/tmp_3 c 32/97 12 01036020 0 201600 00 0 0 2
```

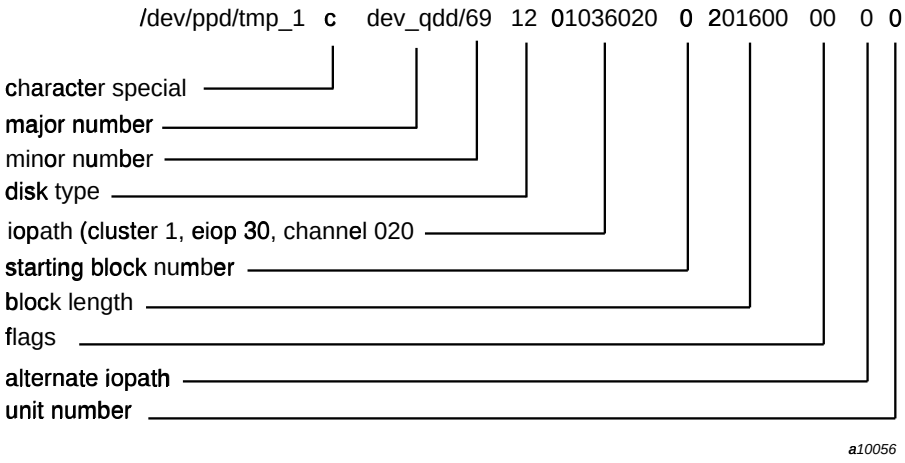

Figure 1. `ddstat` output

Procedure 5: Modifying your configuration file

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

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

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

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

Note: A CRAY J90 IOS-V is case sensitive; enter all lowercase characters for IOS-V commands.

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 Chapter 6, page 133.
2. Create a backup copy of your current configuration file:

```
snxxxx# <CONTROL-A>

(toggles to the IOS)

snxxxx-iosx> cp /sys/param old.param
snxxxx-iosx> <CONTROL-A><RETURN>

(toggles to UNICOS)
```

3. Make sure that you are in `/etc/config` on the UNICOS system. Copy the IOS configuration file `sys/param` from the IOS0 disk drive to a UNICOS disk and a file name of your choice (`new.param` in the following example) by using the `/bin/exdf` command so that you can edit it. The following command specifies that the `/sys/param` file will be read from the IOS system disk (the `-i` input option) and be named `new.param` (the `>` redirection):

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

4. Edit your copy of the parameter file on the UNICOS system (see Section 5.9.2, page 62).

```
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 (`sys/param` on the IOS disk), using the `/bin/exdf`) command. The following command specifies that the `new.param` file will be written to the IOS system disk (the `-o` output option and `<` redirection) and be named `sys/param`:

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

```
snxxxx# exdf -ro sys/param < new.param
```

At this point, the next time the UNICOS system 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. 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 5.13, page 120 describes these additional steps.) You then must restore altered file systems from the back-up tapes you created in step 1 by using the `restore` command.
8. You must create the new `mknod` information for any new disk devices you have defined. To do this, use the `econfig` command to create a file containing the `mknod` information:

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

Remove the existing devices:

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

Generate the new device definitions:

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

5.11 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.

5.11.1 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.

5.11.2 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.

Field	Contents		
	2	default	Use the default file/inode quota that appears in the control file hearder.
	3	infinite	Infinite quota (no quota evaluatin is done).
	4	prevent	No blocks/inodes may be allocated by this ID.
`File warning window`	Number of blocks below the maximum number of file blocks when a warning should be issued.		
`Inode warning window`	Number of inodes below the maximum number of inodes when a warning should be issued.		
`File usage`	Current number of blocks used by the ID.		
`Inode usage`	Current number of inodes used by the ID.		
`Quota hit`	Time when the quota is reached.		

5.11.3 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

5.11.4 Quotas and the user

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

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

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

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

5.11.5 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 */
```

5.11.5.1 Soft quotas

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

Procedure 6: Setting up a quota control file

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

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

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

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

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

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

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

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

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

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

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

```
# vi qudu.out

(Append the following information:)

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

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

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

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

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

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

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

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

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

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

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

5.11.6 Current usage information

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

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

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

5.11.7 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
```

5.11.8 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.

5.11.9 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
```

5.12 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.

5.12.1 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.

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

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

5.12.3 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.

5.12.4 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 the UNICOS system 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.

5.12.5 Apply changes

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

Method 1:

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

Method 2:

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

5.12.6 As you proceed

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

1. Check off items on your detailed plan as you proceed, noting any diversions.
2. Before you format a file system, carefully verify its placement by using the `dmap` 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.

5.12.7 Helpful hints for implementing plan

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

- To copy entire file systems, use the `dump`(8) and `restore`(8) commands; to make partial copies, use `find`(1) and `cpio`(1), or `tar`(1).
- Checkpointed jobs will not continue if the inode numbers of files used by that job change or the minor device number of the holding file system changes; any file system reconfiguration will cause restart failures for checkpointed jobs that have open files on any affected file system.
- The `dump` and `restore` commands change the inode numbers and defragment a file system. You can use the `dd`(1) command only between file systems of the same size and type.
- Moving or reordering slices or adding or removing striping or mirroring changes a file system.
- If you are running in single-user mode, the swap device must exist, but it does not have to be full production size.
- Because the swap device definition comes from the `param` file and not its `/dev` entry, you can move it arbitrarily across system reboots (that is, it needs no preparation before use).
- You can prepare and use the dump device area as a temporary file system; however, be sure that you reinitialize it after you have finished your file system changes.
- If you plan to destroy the `/tmp` file system, notify your users (users like to be warned about changes to the `/tmp` file system).

- When you change a file system that is subject to data migration, you must perform special steps. If you are unsure what is included in these steps, contact your Cray Research service representative.
- Although disk slice names do not have to include the disk number, a logical, ordered naming convention can be useful.
- To tag your data, place a file called `1.am.` *fsname* at the head of each file system (*fsname* represents the name of the file system).
- Do not reuse minor device numbers but keep the highest minor device number under the *type* `MAX` limit for your kernel (in which *type* represents the device type).
- You can use striping only on disk devices that have the same physical type; striping must be between slices of the same size and position on different disks.
- To speed data population, apply logical device cache (`ldcache`) to a destination file system.

5.13 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 section to create file systems on your logical devices. (To determine the devices provided with your system and how they are allocated to file systems, see Section 5.9.3.5, page 66.)

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

The remainder of this section describes the following:

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

- Unmounting a file system by using `/etc/umount`

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

Procedure 7: Create the file system

1. Building the file system

The `/etc/mkfs` command builds the file system structure in the areas of disk that make up the logical device for a given file system. This structure includes designating areas of the logical device to contain the boot block, super blocks, inode region, and so on. On CRAY J90 systems, you always should use the `-q` option when you run `mkfs` to build a structure, which will prevent the disk surfaces from being verified (the IOS `dsurf` and `dslip` commands do this). (When the UNICOS multilevel security (MLS) feature is enabled, `mkfs` provides the new file system with minimum and maximum security levels and authorized compartments.) The format of the `mkfs` command is as follows:

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

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

	`rrf` Round-robin all files (default) `rrd1` Round-robin first-level directories `rrda` Round-robin all directories
`-B` *bytes*	Specifies the number of bytes after which a file is considered to be big. The default is 32,768 bytes (8 blocks) and is defined by the `BIGFILE` argument in `/usr/src/uts/sys/param.h`; you cannot change the definition. The default might be the value you want to use at your site.
`-A` *blocks*	Specifies the minimum number of 4-Kbyte blocks allocated for a file whose size is greater than or equal to `BIGFILE` (see the `-B` option). The default is 21 sectors (blocks) and is defined by the `BIGUNIT` argument in `/usr/src/uts/sys/param.h`; you cannot change the definition. The default might be the value you want to use at your site. For DD-60, DA-62, and DA-301 disk drives, for which the sector size is 16 Kbytes, the allocation unit is rounded up to the nearest multiple of four. For DA-60 disk drives, for which the sector size is 64 Kbytes, the allocation unit is rounded up to the nearest multiple of 16. The interaction of the `-A` and `-B` options is as follows. If a file creation request exceeds the size of `BIGFILE` (8 blocks), the system will allocate `BIGUNIT` (21) more blocks in an attempt to meet the request. The system then checks to see whether the request has been met. If the amount allocated so far is still less than the request, the system will allocate another `BIGUNIT` number of blocks and again check to see whether the request has been met. This cycle of allocation and checking will repeat until the request has been met. You must determine the best settings for the `-A` and `-B` options for your file systems and average allocation requests at your site.
device	Full path name of the block special file (`/dev/dsk/` *filename*). When the disk

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

A basic example follows:

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

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

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

Example 1: round-robin, first-level

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

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

Example 2: round-robin, all-directory

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

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

Example 3: round-robin, all-files

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

Continue with step 2.

2. Labeling the file system

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

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

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

The actual label consists of the following two required fields:

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

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

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

Example 4: assign file system name and volume name to umounted 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 (see Section 5.14.2, page 130). Because of the multipass nature of the `/etc/fsck` command, the file systems must be in an inactive state while being checked. You must ensure that all file systems to be checked are unmounted.

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

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

No matter how many error messages you receive from `fsck`, and no matter how serious the errors may seem, you always can reconstruct your file system

from the last version of your back-up media. Therefore, it is absolutely critical that you have a consistent method of doing backups and that you always follow that method. If you have the backups, you can always restore your file system from the backups if all else fails.

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

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

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

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

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

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

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

Example:

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

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

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

A hardware failure can occur because of the following:

- Disk pack error
- Controller failure
- Power failure

An improper system shutdown can occur because of the following:

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

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

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

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

  1. Allocates an inode from the inode region.
  2. Makes a directory entry, and places the new inode number and file name in the directory.

3. Allocates any data blocks as needed.
4. Increments the link count in the inode for the file. If this is a directory file, the system also increments the link count for the parent directory.

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

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

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

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

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

Continue with step 4.

4. Creating a mount point for the file system

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

Example:

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

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

Continue with step 5.

1. Mounting the file system

A file system is a sequential array of data until it is mounted. When the file system is mounted, the UNICOS kernel interprets the data as a UNICOS file system that is available as part of the system's complete directory structure. To be accessible to the UNICOS system, all file systems except `root` (`/`) must first be explicitly mounted by using the `mount`(8) command. The file system is mounted on an existing directory. The directory may have to be created, using the `mkdir`(1) command (see step 4). By convention, the name of the directory corresponds to the name of the logical device. The fourth field of the `/etc/fstab` file controls the automatic mounting of user file systems when going to multiuser mode. (For steps to configure a file system to be mounted automatically at initialization of multiuser mode, see Procedure 8, page 131.)

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).

5.14 `/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.

5.14.1 `/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.

5.14.2 `/etc/fstab`

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

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

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

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

  you can type one of the following commands instead:

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

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

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

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

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

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

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

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

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

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

(make changes)

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

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

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

2. Uncomment the line for any file system already mentioned in the `/etc/fstab` file that you want to be checked and mounted automatically when the system goes to multiuser mode, or add a line (with the appropriate format) for the desired file system if it is not already mentioned.

3. Edit the fourth field for the desired file system entry to read `CRI_RC=YES`.
4. Save the changes you have made to the `/etc/fstab` file.

Procedure 9: Unmounting file systems

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

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

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

Examples:

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

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

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

```
# /etc/umountem
```

Backing Up and Restoring File Systems [6]

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

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

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

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

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

6.1 Related backup and restore documentation

The following documentation contains information covered in this section:

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

- *CRAY IOS-V Messages*, Cray Research publication SQ–2172: procedure to dump the IOS-V
- *Tape Subsystem Administration*, Cray Research publication SG–2307: tape-related information
- *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271

6.2 Tape devices referenced in `/dev/tape`

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

6.3 Backup and restore utilities

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

6.3.1 `dump` and `restore` utilities

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

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

6.3.2 `rdump` and `rrestore` utilities

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

6.3.3 `dd` utility

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

6.3.4 `tar` and `cpio` utilities

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

6.3.5 `root` and `usr` file systems

There is a minor, but significant change to the disk space allocation scheme for CRAY J90 systems. At initial installation, two sets of production disk partitions named `roota/usra/srca` and `rootb/usrb/srcb` are created. This lets you install into another set of partitions in multiuser mode without disturbing the running system. The CRAY J90 installation utility is designed to perform upgrade installations into the alternative set of partitions. These procedures makes references to `root` and `usr` disk partitions; therefore, if you are using a CRAY J90 system, replace `root` and `bkroot` with `roota` and `rootb`, respectively, and replace `usr` and `bkusr` with `usra` and `usrb`, respectively, in the creation and booting procedures.

At initial installation, the `bkroot` and `bkusr` disk partitions are created automatically for you, except for CRAY J90 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.

For more information see *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271.

Procedure 10: Creating bkroot and bkusr file systems

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

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

1. Create the directories and file structure as follows, replacing values in italics with values appropriate to your system. Use an `mkfs` big file allocation option suitable for the disk types on which `bkroot` and `bkusr` reside for *DEVTYPE* (for example, DD3=20, DD4=20, DD5s=28, and DD5i=28). If you use the UNICOS MLS product, you may have to add security-related options for system levels and compartments (*SECURITY_OPTIONS*).

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

2. Label the file systems, as follows:

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

3. Check file system consistency, as follows:

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

4. Mount the file systems, as follows:

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

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

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

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

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

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

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

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

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

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

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

1. Shut down the UNICOS system by executing the following commands:

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

2. Edit the IOS `/sys/param` file and boot the IOS by executing the following commands (this may be done on the CRAY J90 system console, in which case the prompt will be `sn9` *xxx* `-ios0>`):

```
IOS> cd /sys
IOS> cp param param.prd
IOS> cp param param.bkr

IOS> ed param.bkr
     1
     /rootdev/
     s/ldd root /ldd bkroot/p
        rootdev is ldd bkroot;
     w q
IOS> cp param.bkr param
IOS> cd /
IOS> boot
```

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

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

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

```
# vi /etc/fstab


change root, usr, bkroot and bkusr lines from:


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


to the following:


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

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

```
# vi /etc/config/rcoptions


change ROOTDEV, PIPEDEV, and USRDEV lines from:


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


to the following:


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

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

```
# umount /usr
```

7. Enter multiuser mode by executing the following command:

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

Procedure 12: Backing up the IOS

The following is the procedure for creating a backup copy of the IOS. To back up the Solaris files related to the CRAY J90 installation process and the IOS system software, CRAY J90 users should refer to the "Backup Console Environment" chapter of the *UNICOS Installation Guide for CRAY J90 Model V based Systems*, Cray Research publication SG–5271.

Note: This procedure applies to QIC and DAT devices. It can be run from the IOS console and can be performed regardless of whether the UNICOS system is running. The prompt for CRAY J90 systems will be `sn9` *xxx* `-IOS0`.

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

For quarter-inch cartridge (`QIC`) tapes, a small black dial near one corner of the tape has a raised piece of plastic in the shape of a >. If the point of the shape points at the word SAFE (for example, > SAFE), you **cannot** write to the tape. If you twist the dial so that the point of the shape points away from the word SAFE (for example, < SAFE), you **can** write to the tape.

For 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. Insert the tape into the tape drive.
3. Execute the following commands to copy the files to tape:

For QIC tape devices:

```
IOS> cd /
IOS> tar cvf rpq01 .
```

For DAT devices:

```
IOS> cd /
IOS> tar cvf rpd03 .
```

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 0 dump is the most recent dump with a dump level value less than 8). Wednesday's level 9 dump tapes would contain all changes since Tuesday (Tuesday's level 8 dump is the most recent dump with a dump level less than 9).

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

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

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

- Once per week: You should do a full (level 0) dump.
  - Repeat for each file system you want to copy.

  - Because the `dump` command can read unmounted file systems, you can unmount the file system to be dumped before you begin.

- Daily: You should perform an incremental (level 9) dump:
  - Dump everything that has been modified since the last dump performed with a lower dump level.
  - Repeat for each file system you want to copy.
  - Do this each day that you do not perform a level 0 dump.

6.6 Restoring file systems

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

Options	Description
`-c`	Reads from cartridge tape.
`-f` *file*	Reads the dump from a disk file, rather than tape.
`-i`	Initiates interactive restoration. A shell-like interface is provided that lets the user traverse through the directory tree and select files to be restored.
`-r`	Reads entire tape and loads into current directory. Do this only if you run `mkfs` on the file system first. You usually should do a full `dump` after a full `restore`.
`-t`	Lists the specified file names if the files are on the tape. If no file names are specified, all files on the tape are displayed.
`-v` *vsn*	Causes `restore` to type the name of each file it treats, preceded by its file type.
`-x`	Extracts specified files from the tape (creates subdirectories as necessary).

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

6.7 Increasing and decreasing file system space

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

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

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

6.8 Procedures included in this section

This section includes the following procedures:

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

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

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

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

Note: For this example, a square (`CART`) tape device, with the name `/dev/rss00`, is used.

1. If a `/bin/file` command shows that the UNIX tape logical device to which you want to dump is **not** currently configured in the `/dev` directory,

you must create it by using the mknod command. Typically, these devices have names such as /dev/rss00 (CART), /dev/rmt00 (TAPE), /dev/rpe02 (EXB), /dev/rpq01 (QIC), /dev/rpd03 (DAT), and so on. For more information on the mknod command, see the mknod(8) man page.

For the rest of this example, a square-tape (CART) device with the name /dev/rmt00 is used.

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

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

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

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

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

For quarter-inch cartridge (QIC) tapes, a small black dial near one corner of the tape has a raised piece of plastic in the shape of a >. If the point of the shape points at the word SAFE (for example, > SAFE), you cannot write to the tape. If you twist the dial so that the point of the shape points away from the word SAFE (for example, < SAFE), you can write to the tape.

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

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

3. 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/rss00).

4. Rewind the tape. Round-type tape drives rewind the tape automatically when you push the button to select the tape to be loaded. It is a good

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

```
sn5111# mt -f /dev/rpe02 rew
```

If you intend to write to a tape by using more than one sequential `dump` command, use the nonrewindable versions of each device (such as, `/dev/nrmt00` for round (`TAPE`) tapes, `/dev/nrpe02` for `EXB` tapes, `/dev/nrpq01` for `QIC` tapes, `/dev/nrss000` for square (`CART`) tapes, `/dev/nrpd03` for `DAT` tapes, and so on) in this step and in all subsequent references to the tape device.

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.

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

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

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

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

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

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

```
sn5111# /etc/dump -t 0 -u -f /dev/rss00 /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.

6. Physically alter the tape to prevent the tape from being overwritten (see information included in step 2).
7. Attach a physical label to the tape that states the file systems that have been dumped to the tape and the date the tape was written. It may be useful to add the command that was used to write the tape. It also may be useful to add the commands necessary to restore the tape.

Your file system backup (dump) is now complete.

Procedure 14: Restoring a file system without `tpdaemon`

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

1. If a `/bin/file` command shows that the UNIX tape logical device you want to access is not currently configured in the `/dev` directory, you must create it by using `mknod` commands. Typically, these devices have names such as `/dev/rss00` (`CART`), `/dev/rmt00` (`TAPE`), `/dev/rpe02` (`EXB`), `/dev/rpq01` (`QIC`), `/dev/rpd03` (DAT), and so on.
2. If it was not already unmounted, unmount the file system to be restored by using the `/etc/umount` command. The `/dev/dsk/src` file system is used in this sample procedure.

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

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

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

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

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

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

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

```
sn5111# /etc/mount

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

Warning: Step 3 deletes all information on this file system.

3. **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 Chapter 5, page 51, and the `mkfs`(8) man page.

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

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

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

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

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

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

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

6. Mount the file system on the mount point you have selected by using the `/etc/mount` command. In this example, the mount point `/mnt` is used. If

any user is in the directory or any of its subdirectories, the `mount` command will not be successful (to remove users from a file system forcibly, see the `fuser`(8) man page). If you are the one in the directory or a subdirectory, change to a directory that is not part of the directory tree, including the mount point directory or any of its subdirectories, as follows:

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

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

```
sn5111# cd /mnt
```

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

9. 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 an EXABYTE (`/dev/rpe02`) tape, type the following command line:

```
sn5111# mt -f /dev/rpe02 rew
```

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

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

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

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

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

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

To do a full file system restore :

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

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

Substituting `-f /dev/rmt00` in the preceding example restores to a round (`TAPE`) tape device, `-f /dev/rpe02` restores to an EXABYTE (`EXB`) tape device, `-f /dev/rpq01` restores to a quarter-inch cartridge (`QIC`) device, `-f /dev/rpd03` restores to a digital audio tape (`DAT`), and so on.

To do a partial file system restore :

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

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

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

```
sn5111# /etc/restore -x  -f /dev/rss00 /uts/Nmakefile
```

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

```
sn5111# /etc/restore -x  -f /dev/rss00 /uts/c1/sys
```

Substituting `-f /dev/rmt00` in the preceding partial file system restore examples restores to a round (`TAPE`) tape device, `-f /dev/rpe02` restores to an EXABYTE (`EXB`) tape device, `-f /dev/rpq01` restores to a quarter-inch cartridge (`QIC`) device, `-f /dev/rpd03` restores to a Digital Audio Tape (`DAT`), and so on.

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

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

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

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

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

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

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

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

Procedure 15: 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 the `/etc/udbrestict -r -m R` command to restrict system access, the `/etc/udbrestrict` utility also disables the Network Queuing System (NQS) and `cron` jobs. If an NQS is started in which the `/etc/udbrestict -r` option is set, all checkpointed and all queued NQS jobs of all restricted users will be deleted.

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 (such as the WYSE terminal master console), 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`, `EXB`, `QIC`, 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 QIC           0    0  139
               22 EXB           0    0  139
               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 QIC           0    0  142
               22 EXB           0    0  142
               22 TAPE          0    0  142
               22 CART          1    0  142
jones          14 QIC           0    0    1
               14 EXB           0    0    1
               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 quarter-inch cartridge (`QIC`) tapes, a small black dial near one corner of the tape has a raised piece of plastic in the shape of a >. If the point of the shape points at the word `SAFE` (for example, > `SAFE`), you cannot write to the tape. If you twist the dial so that the point of the shape points away from the word `SAFE` (for example, < `SAFE`), you can write to the tape.

For EXABYTE tapes (type `EXB`) or 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 16: 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 this example is the same file system backed up (dumped) in the backup procedure (`/src`).

Like the `dump` command, you should execute the `restore` command on as idle a system as possible.

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

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

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

Procedure 16(a):

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 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:

```
sn5111# /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 Mon Oct 11 10:41:04
1993
```

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:

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

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

```
sn5111# /etc/mount
/ on /dev/dsk/root read/write on 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: Step 4 **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.

```
sn5111# /etc/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:

```
sn5111# /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:

```
sn5111# /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:

```
sn5111# cd /
sn5111# /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:

```
sn5111# 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:

```
sn5111# /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:

```
sn5111# /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:

```
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.

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
sn5111#
```

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:

```
sn5111# /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:

```
sn5111# rm /mnt/restoresymtabl
```

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

```
sn5111# /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:

```
sn5111# /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:

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

Procedure 17: Restoring a partial file system by using `tpdaemon`

Assumptions

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

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

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

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

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

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

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

Procedure 17(a):

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

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

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

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

There are two methods of doing file system restores. This procedure does not discuss the interactive method in detail, but it is highly effective and very easy to use. To invoke it, use the `-i` option. The `restore`(8) man page gives a good

explanation of how to interact with the interactive shell interface to select files and directory contents for restoration.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
Command: msgd  Page: 1 [delay 10] Thu Oct  7 17:11:10 1993
  Msg #   Time    System Messages
  =====   =====   ===============
  1    17:07   TM122 - mount tape JON1(nl) on a CART device for jones
                  14,  () or reply cancel / device name
  2    17:10   TM011 - enter vsn for tape on device rss000:: display truncated:
> /usr/lib/msg/rep 2 JON1
```

All messages related to your tape job have now disappeared.

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

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

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

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

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

Maintaining Users [7]

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

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

For information on ways to communicate with users, see Chapter 8, page 217.

7.1 Related user accounts documentation

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

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

7.2 The user database (UDB)

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

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

You can modify the UDB in the following ways:

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

- If you do not have a windowing environment, you can use the `/etc/nu` utility (see Section 7.5, page 183).

  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 199). 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). The following section describes a suggested subset of UDB fields (see Section 7.4, page 173). 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 19, page 181).
- 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 20, page 182).

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

- Follow the procedures in section Section 7.5, page 183, to make the new entry in /etc/udb by using the /etc/nu -a command. (Section 7.5, page 183 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 199, to make the desired entry in /etc/udb by using the /etc/udbgen command. (Section 7.6, page 199 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 172.

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 (UIDs) and user names found in the UDB extension files along with their associated `udb_priva` and `udb_pubva` record offsets will 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` will 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` utility uses this script to create a user directory and to change the permissions on this directory.
`/etc/nulib/nu2.sh`	The `nu` utility uses this script to initialize the contents of the user's directory.
`/etc/nulib/nu3.sh`	The `nu` utility uses this script to purge a login account.
`/etc/nulib/nu4.sh`	The `nu` utility uses this script to purge a login without removing the account from the UDB. This action is performed to preserve accounting information.

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.
/etc/nu	Adds, deletes, and modifies login records. `/etc/nu` uses the following 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

The remainder of this section includes information and procedures about using the `/etc/nu` and `/etc/udbgen` utilities to maintain your user records.

Procedure 18: 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 183).

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 "Adding users to `/etc/udb` by using `/etc/udbgen`," page Procedure 26, page 201).

Login name field

user_name:	The user's login name must be a unique 1- to 8-character alphanumeric representation, in which the first character is alphabetic.

Encrypted password field

`passwd`: *encryption*:	Encrypted password to be stored in the user's record. The password content is not validated.

Password aging field

`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.
User ID field	
`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.
Group ID field	
`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.
User comment field	
`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.
Login (home) directory	
`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.
User shell at login	
`shell`: *sh_name*:	Default login shell. You can specify a maximum of 63 characters. Default value for *sh_name* is `/bin/sh`.
Account ID field	
`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.
Login root directory	
`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 value	
`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.

CPU limits

Job CPU time limit
`jcpulim[b]`: *n*:
`jcpulim[i]`: *n*:
: Job CPU time limit (in seconds) for batch (`[b]`) or interactive (`[i]`) jobs. The default is unlimited.

Per-process CPU limit
`pcpulim[b]`: *n*:
`pcpulim[i]`: *n*:
: Per-process CPU limit (in seconds) for batch (`[b]`) or interactive (`[i]`) processes. The default is unlimited.

Memory limits

Job memory limit
`jmemlim[b]`: *n*:
`jmemlim[i]`: *n*:
: Job memory limit in 4096-byte blocks for batch (`[b]`) or interactive (`[i]`) jobs. The default is unlimited.

Per-process memory limit `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.

Process limits

Job process limit
`jproclim[b]`: *n*:
`jproclim[i]`: *n*:
: Job process limit for batch (`[b]`) or interactive (`[i]`) jobs. If you do not specify a value for this

	field, the default is the value of `/MAXUP` in `sys/param.h`.
SDS limits	Secondary data segments (SDS) are not supported on CRAY J90systems; you should ignore these fields and use the default setting.
Tape limits	
Job tape unit limit `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.
File limits	
Per-process file allocation limit `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 19: 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. Section 7.5, page 183, describes how to use `/etc/nu`, and Section 7.6, page 199, describes how to use `/etc/udbgen`. To determine which utility will work best for you in a given situation, you may want to read both sections.

Procedure 20: 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. Section 7.5, page 183, describes how to use `/etc/nu`, and Section 7.6, page 199, describes how to use `/etc/udbgen`. To determine which utility will work best for you in a given situation, you may want to read both sections.

7.5 Using the `/etc/nu` utility

The `/etc/nu` utility is a prompt-driven utility for interactively adding, deleting, and modifying user records. It uses a configuration file called `/etc/nu.cf60`. This section describes the following topics:

- Procedure for changing `/etc/nu` configuration parameters

- Procedure for creating a file system to use with `/etc/nu`
- Procedure for adding user records to `/etc/udb` by using the `/etc/nu` utility
- Procedure for modifying user records by using the `/etc/nu` utility
- Procedure for deleting user records by using the `/etc/nu` utility

Procedure 21: 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 (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 22: 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 21, page 184.)

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 23: 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.

For details on these procedures, see Procedure 18, page 176 through Procedure 22, page 186.

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 the procedure about determining settings for UDB fields, page Section 7.4, page 173.

The user's login name is something you provide when the /etc/nu utility prompts you. The user's login name must be a unique 1 to 8 alphanumeric

representation, in which the first character is alphabetic. Typically, the name is made up of lowercase alphabetic characters.

You may want to change the UDB password aging field to `force` so that the user must change the initial password when logging in for the first time. You must remove the `off` setting for the `DefaultAge` variable in the `/etc/nu.cf60` file (either manually or by using the menu system) so that the password aging field shows up and can be set when executing the `/etc/nu` script.

The `nu` utility has other options that you might want to use when adding a user, such as the following `-p` and `-c` options:

`-p` *dirname*	Modifies UDB files found under the *dirname* directory; lets you maintain private copies or test versions.
`-c` *dirname*	Uses the configuration file, `nu.cf60`, under the *dirname* directory; lets you specify an alternative configuration file.

To determine whether a record exists for a user, use the `udbsee` command.

Example `/etc/nu` session that adds a user

The following `nu` session adds login name `jones` (**`bold`** indicates what you would type; otherwise, you can use the default values):

```
# /etc/nu -a
cmd/nu/nu.c
Login name? (1-8 characters) [quit] jones
Enter password:
Please re-enter password:
Enter actual user name: John R. Jones
User id number? (max = 60000)  [2] 624
Which groups? (names or numbers, use commas, ? for list) [0] cray,test,trng,usrsrc
```

(See procedure for adding groups)

```
You selected groups:
100 0
cray , 100
104 1
test , 104
105 2
trng , 105
98 3
usrsrc , 98
Are these correct? (y or n) [y]
Login directory?  [/hot/u1/jones]
```

(This will be the user's home directory.)

```
Enter shell [/bin/csh]
Which accounts? (names or numbers, use commas, ? for list) [0]
You selected accounts:
```

(See procedure for acids)

```
root , 0
Are these correct? (y or n) [y]
User default login root?  [/]
```

(This will be the user's root directory; set to other than / to restrict acess to file systems.)

```
Resource group ID? (name or number, ? for list) [0] Users
Which permissions? (names or numbers, use commas, ? for list) [none]
You selected permbits:
none
Are these correct? (y or n) [y]
Allocation shares? (min=0) [100]
DEFAULT security compartments? (name1,name2,... or none, ? for list)  [default]
VALID security compartments? (name1,name2,... or none, ? for list)  [default]
Security permissions? (name1,name2,... or none, ? for list)  [default] Security levels?
(default max min)  [0 0 0]
Integrity classes? (default max)  [0 0]
DEFAULT integrity categories? (name1,name2,... or none, ? for list)  [default]
VALID integrity categories? (name1,name2,... or none, ? for list)  [default]
Do you want this user locked? (y or n) [n]
Do you want this user trapped? (y or n) [n]
Per job process limit for batch? (min=0)  [100]
Per job process limit for inter? (min=0)  [100]
Per job MPP PE limit for batch?  [none]
Per job MPP PE limit for inter?  [none]
Per job MPP time limit for batch?  [none]
Per job MPP time limit for inter?  [none]
Per job MPP barrier limit for batch?  [none]
Per job MPP barrier limit for inter?  [none]
Per process MPP time limit for batch?  [none]
Per process MPP time limit for inter?  [none]
Will the user be using the Cray Station? (y or n) [y] n
```

(No for Cray J90 systems.)

```
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  pwage ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624
   6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/rnl/tmp/jones
   8)  shell ...... /bin/csh
   9)  acids ...... root (0)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
```

```
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... default
  18)  valcomps ... default
  19)  permits .... default
  20)  intcls...... 0            21)  maxcls...... 0
  22)  intcat...... default      23)  valcat...... default
  24)  disabled ... 0            25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
  28)  pcpulim[b] ...      none       pcpulim[i] ...      none
  29)  jmemlim[b] ...      none       jmemlim[i] ...      none
  30)  pmemlim[b] ...      none       pmemlim[i] ...      none
  31)  pfilelim[b] ..      none       pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576       jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576       psdslim[i] ...   1048576
  34)  jtapelim[b][type0   ]      99      jtapelim[i][type0   ]     99
       jtapelim[b][type1   ]      99      jtapelim[i][type1   ]     99
       jtapelim[b][type2   ]      99      jtapelim[i][type2   ]     99
       jtapelim[b][type3   ]      99      jtapelim[i][type3   ]     99
       jtapelim[b][type4   ]      99      jtapelim[i][type4   ]     99
       jtapelim[b][type5   ]      99      jtapelim[i][type5   ]     99
       jtapelim[b][type6   ]      99      jtapelim[i][type6   ]     99
       jtapelim[b][type7   ]      99      jtapelim[i][type7   ]     99
  35)  jpelimit[b] ...      none       jpelimit[i] ...      none
  36)  jmpptime[b] ...      none       jmpptime[i] ...      none
  37)  jmppbarrier[b]       none       jmppbarrier[i]       none
  38)  pmpptime[b] ...      none       pmpptime[i] ...      none
Are these values OK? (y or n)  [y]
Entry looks like:
jones:co:John R. Jones
jones:ui:624:di:/hot/u1/rnl/tmp/jones:sh:/bin/csh:dr:/:pw:v0u28k2K1wtX6
jones:gi:100,104,105,98
jones:ai:0
jones:rg:102:as:100
jones:dc:default:cm:default:pm:default
jones:ic:default:vc:default
jones:pj[b]:100:pj[i]:100
jones:js[b]:1048576:js[i]:1048576:ps[b]:1048576:ps[i]:1048576
jones:tp:type0[b]:99:tp:type0[i]:99:tp:type1[b]:99:tp:type1[i]:99
jones:tp:type2[b]:99:tp:type2[i]:99:tp:type3[b]:99:tp:type3[i]:99
jones:tp:type4[b]:99:tp:type4[i]:99:tp:type5[b]:99:tp:type5[i]:99
jones:tp:type6[b]:99:tp:type6[i]:99:tp:type7[b]:99:tp:type7[i]:99+ test 1 -ne 0
```

```
+ rm -rf /hot/u1/jones
+ mkdir /hot/u1/jones
+ test /hot/u1/jones != /hot/u1/jones
+ test 0 -eq 0 -a 1 -ne 0
+ chgrp 100 /hot/u1/jones
+ chown 624 /hot/u1/jones
+ chacid -s root /hot/u1/jones
Do you wish to add more new users? (y or n)  [y] n
execing udbgen - please wait


udbgen complete (at this time, nu executes udbgen)
```

Procedure 24: 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] ...      noneSelect
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 25: 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 `/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 the example on page 180 of the following procedure.

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 section provides the following:

- 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 26: 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 18, page 176 through Procedure 20, page 182. 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:`*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:"
```

Verify that your entry was added by using the udbsee(1) command.

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 (followed by 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 27, page 208).

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 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 27: 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 212.

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 28: 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 6, page 209).
-	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.

Examples of updating user logins by using `udbgen`:

The following examples show how to update user login entries in the UDB by using the `/etc/udbgen` command:

Example 6: 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 7: 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 8: 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 9: 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 10: 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 29: 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 step .

5. 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 section 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 `/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 section.

Procedure 30: 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 31: 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 32: 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 section:

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `mail`(1), `news`(1), `su`(1), `wall`(1), and `write`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `issue`(5) and `motd`(5) man pages

8.2 Issuing emergency messages only

To write (broadcast) emergency messages to all users currently logged in, use the `/etc/wall` command. When a super user executes this command, it overrides message suppression; therefore, use it with discretion.

To run the command, type the /etc/wall command. 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)).

Users who are not currently logged in will never see the message; thus, `wall` is not a suitable method for communicating a message to all users who have accounts on the system.

Typically, the `wall` command is 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 the `login`(1) man page).
- 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).

The following is an example of creating a `wall` message:

```
# /etc/wall
Please log off.  The system will be coming down in 5 minutes for file
system adjustment.  It will be brought back up within the hour.
CONTROL-d.
```

8.3 Issuing critical messages

The `/etc/issue` file is displayed while a user is logging in before he or she 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. All interactive users can determine from the message whether they want to log in to the system.

Messages placed in `/etc/issue` should be brief and so important that users may need the information to decide whether 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` 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 can log in

For example, if the message states that the system is going down for maintenance in 5 minutes, a user may choose not to log in to the system at this time. When messages are no longer applicable, be sure to remove them.

The following is an example of creating an `issue` message:

```
# vi /etc/issue
This machine is being brought down in 5 minutes, 5:30 PM
```

To delete an issue message, enter the following command:

```
# rm /etc/issue
```

8.4 Issuing special messages (message of the day)

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 you may place messages in it by using any UNICOS text editor.

You should place messages in `/etc/motd` that are less immediate than those requiring the use of the `wall` command, but that are important enough that users should be forced to see them. You should remove messages from `/etc/motd` as soon as they are no longer needed. Suitable topics for using the `/etc/motd` file include the following:

- Messages to users of scheduled down time
- Warnings to users to clean up unnecessary files on a particular file system or systems
- Brief explanations of recent problems that may have affected users, often with a pointer to a news item that contains a more detailed explanation

Note: Be sure to remove messages when they are no longer applicable; otherwise, your users may start to ignore them. To remove the message of the day, use the `cp /dev/null /etc/motd` command (rather than `rm /etc/motd`); otherwise, `/etc/profile` tries to `cat` a file that does not exist.

The following is an example of creating a message of the day:

```
# vi /etc/motd
============================================================
This machine will be brought down at 5:30 PM today for
maintenance work.  It should be down for only 1 hour.
============================================================
```

To delete the message of the day, enter the following command:

```
# cp /dev/null /etc/motd
```

8.5 Issuing noncritical communication to all users

The `news` command is the preferred method for delivering noncritical messages to your users. Files you place in the `/usr/news` directory should be ordinary text files you create with any UNICOS text editor. You should create news files that have meaningful names in the `/usr/news` directory.

Note: Be sure to clean out this directory on a regular basis by removing items older than some arbitrary age, such as, 2 months. You can do this by using the `cron` command (for information about using the `cron` command, see the `cron`(8) man page).

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` command), `/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 about 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

Each time a user executes `news`, the `$HOME/.news_time` file is updated.

To display the contents of all current news items, invoke the `news` command without any options.

The following options are commonly used with the `news` command:

Option	Description
`-s`	Displays a count of current news items (that is, those created since the last modification of `$HOME/.news_time`)
`-n`	Displays the names of current news items

The `/etc/profile` file usually contains a `news -n` command so that each user receives a list of current news items at login time.

The following is an example of creating a news item:

```
# vi /usr/news/chem_access
We have now added a new group called usrchem to control access to the
chemistry applications.  If you need to belong to this group, call the
help desk for validation.

                        John Doe  (jd@cray)
```

To execute a news item, enter the following command:

```
# /usr/news/filename
```

To delete a news item, enter the following command:

```
# rm /usr/news/chem_access
```

8.6 Using the `write` command

The `write` command initiates immediate person-to-person communication with a user who is logged in 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 who has a login name of `dolores`, for example, you would issue the following command:

```
# write dolores
```

If 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, user `dolores` is currently logged in, a message appears on her screen indicating that you are writing to her. Typically, 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, `write` 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 using the `write` command 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 are 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` command will prove valuable when `write` is the appropriate communication method.

8.7 Using the `mail` command

The `mail` command provides a way to leave messages for specific users, whether or not they are currently logged into the system.

The `mail` command is used as shown in the following example:

`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 specified. 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` command 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`

command to read the message; however, in practice, most users read their mail when they log in.

For more information, see the `mail`(1) and `mailx`(1) man pages.

Log Files [9]

This chapter 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 chapter through normal file manipulation commands such as `tail`(1), `cat`(1), `pg`(1), and `more`(1).

This chapter 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 241.

9.1 Related log files documentation

The following documentation contains information that you will find useful in understanding the material presented in this section:

- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `brc`(8), `cron`(8), `newsys`(8), `reduce`(8), `sar`(8), and `syslogd`(8) man pages
- *CRAY IOS-V Commands Reference Manual*, Cray Research publication SR–2170
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `at`(1), `batch`(1), `cat`(1), `crontab`(1), `date`(1), `logger`(1), `more`(1), `sar`(1), `tail`(1), and `uname`(1) man pages

9.2 /etc/boot.log file

The `/etc/boot.log` file records boot dates and times for a system. When the `/etc/rc` script is executed, it appends a record to the `/etc/boot.log` file. The file is composed of output from the `/bin/date` and `uname -a` commands. The format of the `/etc/boot.log` file includes the system name, node, release, version, and hardware information. To determine the last time a system was booted, see this log. The format is as follows:

`date, uname -a` *yy/mm/dd hh:mm system node release version hardware*

Example:

```
# cat /etc/boot.log
93/09/10 08:07 sn1703c cool 8.0.2 CRAY Y-MP
```

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 the UNICOS system 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 section 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 171).

The following types of user account transactions are recorded into `/usr/adm/nu.log`: `changed`, `added`, `deleted`, and `destroyed`.

An excerpt from a `nu.log` file follows:

```
Text goes  here
# cat /usr/adm/nu.log
jones:co:L B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
        changed to
jones:co:Lauren B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
jones:tp:type0[b]:3:tp:type0[i]:3:tp:type1[b]:3:tp:type1[i]:3
jones:tp:type2[b]:3:tp:type2[i]:3:tp:type3[b]:3:tp:type3[i]:3
        by jones on Mon Sept  13 11:51:00 1993
```

9.5.7 /usr/adm/sa/sa*DD*

The `sar` command uses the `/usr/adm/sa/da`*DD* data collection file to report system activity. The `/usr/lib/sa/sadc` and `/usr/lib/sa/sa1` commands write data to this file; they must be scheduled by `cron` to run at frequent intervals (such as every 15 minutes).

The `/usr/adm/sa/sa`*DD* file is too large and too varied to show a representative example. It is filled with multiple types of reports, each with many different output fields.

For more information about system activity reporting, see the `sar`(1) and `sar`(8) man pages and *UNICOS Resource Administration*, Cray Research publication SG–2302.

9.5.8 /usr/adm/sl/slogfile

The `/usr/adm/sl/slogfile` data file records UNICOS multilevel security (MLS) event information. The `reduce` command, executable only by the

security administrator, reads this data file. The `reduce` command extracts, formats, and outputs entries from UNICOS MLS event files. The MLS event logging daemon, `slogdemon`, collects security-relevant records from the operating system by reading the character special pseudo device `/dev/slog`. An excerpt of the output from the `reduce` command follows:

```
# /etc/reduce -s 04021300 -u jones -p

Apr  2 14:49:21 1993  Validation        o_lvl: 0  s_lvl: 0  jid:0  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]    ********
    Login uid: jones(8863)
  Login to [jones(8863),tng(146)] : Okay   via 128.162.121.20   on /dev/ttyp042
Apr  2 14:49:21 1993  Setuid Syscall    o_lvl: 0  s_lvl: 0  jid:1255  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]         ********
    Login uid: jones(8863)
   Setuid call from root (0) to jones (8863) was successful::
```

9.5.9 /usr/spool/msg/msglog.log

The `/usr/spool/msg/msglog.log` file contains messages and replies to and from the operator. Following is an excerpt from a `msglog.log` file:

```
# cat /usr/spool/msg/msglog.log

Sep 16 09:51:20  Message    1: WARNING THRESHOLD ON /nasc
Sep 16 09:58:59  Message    2: CRITICAL THRESHOLD ON /nasc::

Jun  9 23:34:02  Message daemon stopped
Jun 10 00:43:15  Message daemon started
Jun 10 04:07:49  Message    1: From ghe:  How are you?
Jun 10 04:08:44  Reply    1: good::
Jun 18  12:41:52  Informative: ********** SYSTEM ACCOUNTING RESTARTED for 0618/*
Jun 18 12:48:21  Informative:  *************** SYSTEM ACCOUNTING COMPLETE Thu J*:
```

9.5.10 /usr/lib/cron/cronlog

The `/usr/lib/cron/cronlog` file reports the status of all commands that `cron` executes, including `at`, `batch`, and `crontab`. When the UNICOS system is brought to multiuser mode, the old log file is copied to `/usr/lib/cron/OLDLOG`.

Various types of error messages may be present in the `cronlog` file, including messages about when `cron` was started and stages of job execution. The `cronlog` file has the following format:

`CMD:` *command_executed username process_id job_type start_time username process_id job_type end_time* `rc=`*error return code*

The *job_type* argument can have one of the following values:

`a` = `at`(1) job

`b` = Batch job

`c` = `cron`(8) job

An example of `/usr/lib/cron/cronlog` follows:

```
! *** cron started ***   pid = 3654 Thu Sep 16 17:47:44 1993
! new user (ce) with a crontab Thu Sep 16 17:47:45 1993
! new user (nfs) with a crontab Thu Sep 16 17:47:45 1993
! new user (root) with a crontab Thu Sep 16 17:47:46 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 3687 c Thu Sep 16 17:48:01 1993
<  root 3687 c Thu Sep 16 17:48:02 1993
>  CMD: /usr/lib/acct/ckpacct
>  root 4291 c Thu Sep 16 18:00:01 1993
>  CMD: /usr/lib/sa/sa1 600 1
>  root 4292 c Thu Sep 16 18:00:01 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 4293 c Thu Sep 16 18:00:01 1993
<  root 4293 c Thu Sep 16 18:00:02 1993
<  root 4292 c Thu Sep 16 18:00:02 1993
<  root 4291 c Thu Sep 16 18:00:04 1993
>  CMD: $HOME/scripts/runsequence cpuseq b
>  ce 4731 c Thu Sep 16 19:30:01 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 4732 c Thu Sep 16 19:30:01 1993
<  root 4732 c Thu Sep 16 19:30:01 1993
<  ce 4731 c Thu Sep 16 19:30:12 1993
```

9.5.11 `/usr/tmp/nqs.log`

The Network Queuing System (NQS) log, created by the NQS log daemon, contains NQS activity. Its default location is the ASCII file `/usr/spool/nqs/log` (to change the location of the log file, use the `qmgr`

`set log_file` command; to see where the current log file resides, use the `qmgr show parameters` command). Access to `/usr/spool/nqs` is restricted; however, if you have the correct permissions, you can access the NQS log file by using normal file manipulations, such as `tail`, `cat`, and `more`. If you experience problems with NQS, use a `tail -f` command on this file to observe what NQS is doing.

A sample `nqs.log` file follows:

```
# cat /usr/tmp/nqs.log

NQS(INFO): local mid = 130
I$nqs_boot(): TZ=CST6CDT
NQS(DEBUG): tra_read():0, pid 4033,  state=0, sequence#=0, tid=0
NQS(DEBUG): gen_shrpri_tree(): completed setudb.
NQS(INFO):  gen_shrpri_tree(): Fair Share turned off, Share_wt sched factor set.
NQS(DEBUG): gen_shrpri_tree(): Sh_Decay_usage =  0.0000, Sh_Run_rate =  1.0000
NQS(DEBUG): gen_shrpri_tree(): Share_basis & SHAREBYACCT = 8
NQS(DEBUG): gen_shrpri_tree(): childcnt = 1, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 2, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 3, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 4, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 5, st[0].childsum = 0
NQS(INFO):  nqs_ldconf(): i = 1NQS(INFO): nqs_ldconf(): Pipe queue gale; Dest count: 1
NQS(INFO):  nqs_ldconf(): Creating new destination 0NQS(INFO):  nqs_ldconf(): batch
NQS(INFO):  nqs_upd.c(): Adding new destn batch to head of queue
NQS(INFO):  upd_addquedes(): Updating queue gale destinations
NQS(INFO):  upd_addquedes(): Destination 0;  832NQS(INFO):  nqs_ldconf(): i = 1
NQS(INFO):  upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO):  upp_setlogfil(): Set/Reset command - $$/usr/spool/nqs/log
NQS(INFO):  netdaemon(): Listening on TCP/IP port: nqs
NQS(INFO):  nqs_rbuild(): Set flag for first time thru spawn.
NQS(INFO):  nqs_boot(): BOOTDONE, Database present.
NQS(INFO):  upp_setchkpntdir(): New directory = /usr/spool/nqs/private/root/chkt
NQS(INFO):  upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO):  upp_setlogfil(): Set/Reset command - #$/usr/spool/nqs/log
NQS(INFO):  upp_setsnapfil(): New pathname = /home/swts/cjd/nqs_snapfile
```

9.5.12 `/usr/adm/errfile`

The error log is a binary file that contains error records from the operating system. `errpt` processes error reports from the data. The `/etc/errdemon`

command (see the errdemon(8) man page) reads /dev/error and places the error records from the operating system into either the specified file, or errfile, by default. The /etc/rc (see the brc(8) man page) script starts /etc/errdemon, and /etc/mverr starts a new errfile.

9.5.13 /usr/spool/dm/*

If UNICOS Data Migration Facility (DMF) software is configured on your system, the /usr/spool/dm/dmdlog.*YYMMDD* files record activities that pertain to data migration.

A sample /dm/dmdlog.*YYMMDD* file follows:

```
# cat /usr/spool/dm/dmdlog.930912

        dmdlog.930912

10:55:29 Data Migration daemon 35745 initializing, release level 6100
10:55:29 0 index entries in database
10:55:29 Command request pipe initialized, fd = 7
10:55:29 Kernel request pipe initialized, fd = 8
10:55:29 initmsp: msp fake, pid = 35751, wt_fd = 10, rd_fd = 11
10:55:29 machine id set to 2158163973
10:55:29 First available handle for assignment is 2158163973:1
10:55:30 0 incomplete MSP entries found
10:55:30 0 soft-deleted premigration files found
   .
   .
   .
10:56:35 Counts - permdel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - retrybu,     0,     0,     0,     0,  0,  0
10:56:35 Counts - krecall,    10,    20,    20,     0,  0,  1
10:56:35 Counts - kremove,    28,    28,    28,     0,  0,  1
10:56:35 Counts - kcancel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - invalid,     0,     0,     0,     0,  0,  0
10:56:35 Counts -  pclear,     0,     0,     0,     0,  0,  0
10:56:35 Current mem  = 94437
10:56:35 Stopping daemon processing
10:56:35 Data migration daemon stopped, exit=0
```

Note: The following log files also exist for each file system under data migration:

- `dmloght` (generated by the `dmhit` command)
- `dmlogct` (generated by the `dmmctl` command)
- `dmlogsm` (generated by the `dmfree` command)

9.6 Cleaning up system logs

Some log files are recycled during each reboot, some logs accumulate content slowly and must be cleaned up only occasionally, and some log files accumulate content quickly and should be monitored and cleaned up frequently. This section describes each group of log files.

9.6.1 Log files recycled during each reboot

The following log files are recycled during each reboot; therefore, you do not have to monitor them for space consumption. If any of the log files must be saved, however, you should copy them to a location of your choice before shutting down the system. If you forget their location, most of the log files are linked to `/usr/spool/ccflogs` .

Log files that recycle are as follows:

- `/etc/rc.log` (log file from `init 2` function)
- `/usr/adm/sulog` (including all `su` records)
- `/usr/spool/msg/msglog.log` (messages and replies from and to an operator)
- `/usr/lib/cron/log` (all `cron` entries since reboot)
- `/usr/tmp/nqs.log` (all NQS entries)

9.6.2 Small accumulative log files

The following log files accumulate content slowly, but you should clean them up occasionally so that they do not consume space needlessly:

- `/etc/boot.log` (records boot dates and times for a system)
- `etc/dump.log` (records crash dump dates and dump file locations)

- `usr/adm/nu.log` (records all `/etc/nu` output)

9.6.3 Large accumulative log files

The system activity report (`sar`) data and report log files accumulate content quickly; therefore, you should monitor these and clean them up frequently. If not managed promptly, these log files could potentially saturate the `/usr` file system. All `sar` data is saved up to 31 days in `/usr/adm/sa/sa`*DD*. At the end of each month, you should dump them to a file server or to tape; otherwise, newer collected data will overwrite them. The `sar` reports (stored in `/usr/adm/sa/sar`*DD*) are kept only up to 7 days, because the reports usually are not backed up. To change the number of days you want to keep `sar` data or `sar` reports, modify `/usr/lib/sa/sa2`.

You also should monitor the following log files:

- Email log file
- User mail files if not read and cleared
- NQS log files (these can grow quickly)
- `errpt` files when active disk errors or tape activity exists
- MLS log files, which are located in `/usr/adm/sl`

Accounting [10]

The UNICOS system 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 243, which describes the unique features of CSA.

This chapter 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`. A sample file is provided at the end of this chapter; the sample file may differ slightly from the one included with your system.

For information on using standard UNIX System V accounting, see *UNICOS Resource Administration*, Cray Research publication SG–2302.

10.1 Related accounting documentation

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

- *UNICOS Resource Administration*, Cray Research publication SG–2302, "Accounting" chapter
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `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) man pages
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `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*, Cray Research publication SG–2302.
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 (for NQS and the tape subsystem); accounting information 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*, Cray Research publication SG–2302.

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*, Cray Research publication SG–2302.

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 51.)

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*, Cray Research publication SG–2302.) 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*, Cray Research publication SG–2302.)

`/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*, Cray Research publication SG–2302. 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 login and logout records for users. Records tty, process ID, type of process, and connect-time accounting data. For information about fixing `wtmp` errors, see Section 10.9, page 254.
`/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 10.8, page 252.

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 2, page 250. 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 10.7, page 251)

   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.

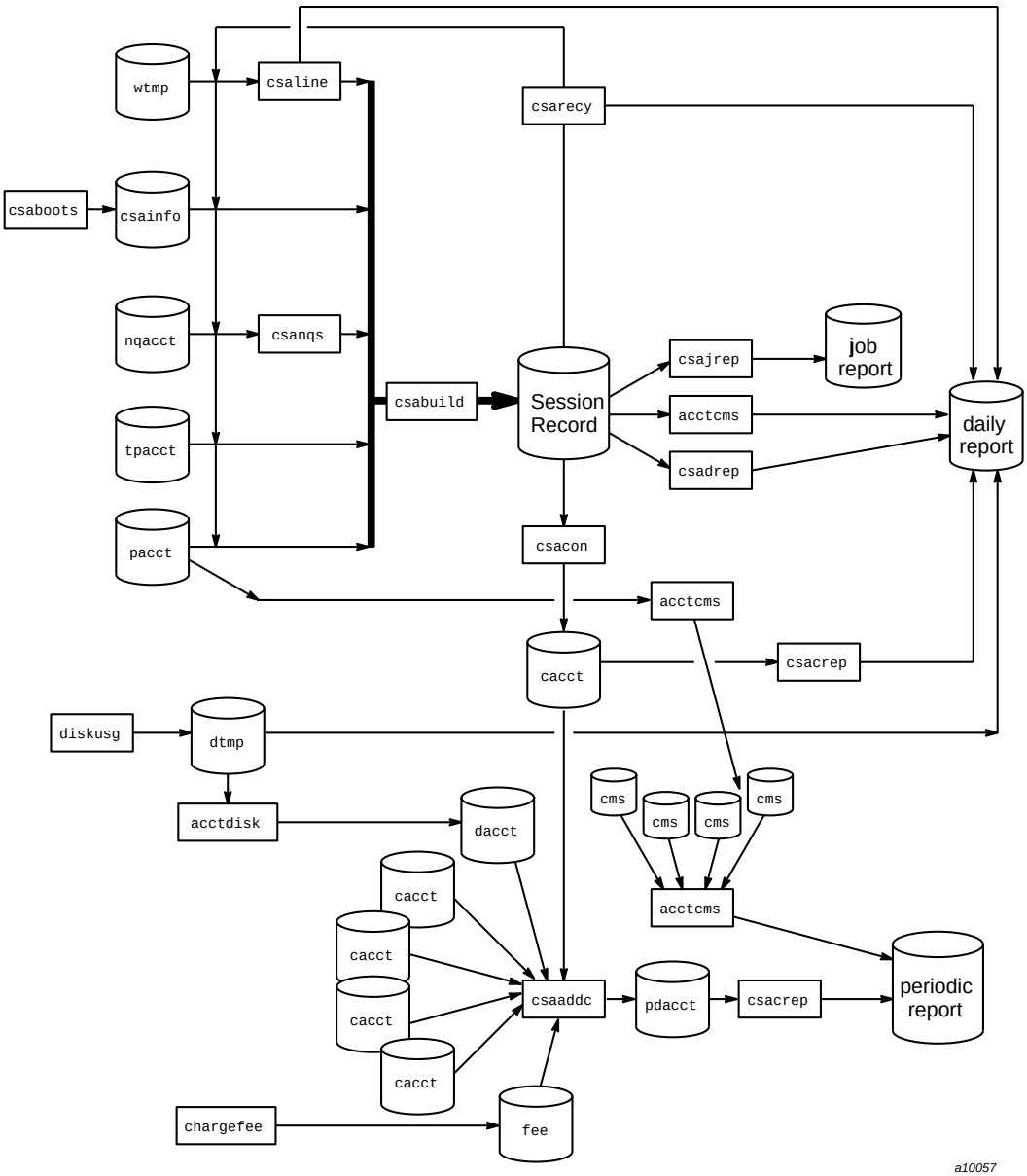

Figure 2. Daily operation overview of CSA

10.6 Customizing your system billing procedure

UNICOS system accounting has been designed to be easy for you to customize to meet your site's requirements. For instance, you may want to change the way the system charges system billing units (SBUs) for various kinds of services, to define the weighting factor used in calculating SBUs for various system services (such as tape requests, NQS requests, or connect time), to change the definition of the memory integral to be used in connection with memory charges for multitasking programs, and so on. You may want to set prime time equal to nonprime time and charge based on NQS queue usage. To do so, you must modify the `/etc/config/acct_config` file (a sample file is included beginning on page Section 10.13, page 262 of this chapter). By default, all SBUs are set to 0.0. Read the commented instructions throughout the file to determine which lines of the `acct_config` file you want to modify for your site's needs. You also can use your local SBU calculations by modifying the default algorithms defined in `/usr/src/cmd/acct/lib/acct/user_sbu.c`, compiling, and relinking the accounting programs.

10.7 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*, Cray Research publication SG–2302).

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.

10.8 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 Section 10.9, page 254.
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.

10.9 Fixing wtmp errors

The wtmp files generally cause the highest number of errors in the day-to-day operation of the accounting subsystem. When the date is changed, and the UNICOS system is in multiuser mode, a set of date change records is written into the /etc/wtmp file. When a date change is encountered, the wtmpfix (see the fwtmp(8) man page) program adjusts the time stamps in the wtmp records.

Some combinations of date changes and reboots, however, slip by wtmpfix and cause csaline to fail. The following steps show how to repair a wtmp file:

```
$ cd /usr/adm/acct/work/MMDD/hhmm
$ /usr/lib/acct/fwtmp < Wwtmp > xwtmp
$ ed xwtmp


(delete corrupted records)

$ /usr/lib/acct/fwtmp -ic < xwtmp > Wwtmp


(restart csarun at the wtmpfix state)
```

If the wtmp file is beyond repair, create a null Wwtmp file. This prevents any charging of connect time.

10.10 Verifying data files

To verify data files, use the csaedit, csapacct, and csaverify commands. csaedit and csapacct verify and delete bad data records; csaverify only flags bad records. By default, csaedit and csaverify are invoked in csarun to verify the data files.

These commands may allow files that contain bad data, such as very large values, to be verified successfully.

10.11 Editing data files

You can use the csaedit and csapacct commands to verify and remove records from various accounting files. The following example shows how you can use csapacct to verify and remove bad records from a per-process (pacct) accounting file.

In this example, csapacct is invoked with verbose mode enabled (valid data records are written to the pacct.NEW file):

```
$ /usr/lib/acct/csapacct -v pacct pacct.NEW
```

The output produced by this command line is as follows:

```
Bad record - starting byte offset is 077740 (32736)
   invalid pacct record - bad base parent process id 97867
Found the next magic word at byte offset 0100130, ignored 120 bytesFound 394 BASE records
Found 4 EOJ records
Found 1 MTASK (multitasking) records
Found 0 ERROR records
Found 0 IO records
Found 0 SDS records           # not on CRAY-2, J90 systems
Found 0 MPP records           # not on CRAY-2, J90 systems
Found 0 PERFORMANCE records   # not on CRAY-2 systems
Outputted records for 398 processes
Ignored 120 bytes from the input file
```

You can use csaedit and csapacct in conjunction with csaverify , by first running csaverify and noting the byte offsets of the first bad record. Next, execute csaedit or csapacct and remove the record at the specified offset.

The following example shows how you can verify and then edit a bad `pacct` accounting file:

1. Verify the `pacct` file by using the following command line; the following output is received:

```
$ /usr/lib/acct/csaverify -P pacct

/usr/lib/acct/csaverify: pacct: invalid pacct record - bad base parent process id
    97867 byte offset: start = 077740 (32736)  word offset: start = 07774 (4092)
/usr/lib/acct/csaverify: pacct: invalid pacct record - bad magic word 03514000
   byte offset: start = 0100070 (32824)  word offset: start = 010007 (4103)
```

2. Delete the record found at byte offset 32736, as follows (valid records are written to `pacct.NEW`):

```
$ /usr/lib/acct/csapacct -o 32736 pacct pacct.NEW
```

3. Reverify the new `pacct` file to ensure that all the bad records have been deleted, as follows:

```
$ /usr/lib/acct/csaverify -P pacct.NEW
```

You can use `csaedit` to produce an abbreviated ASCII version of some of the daemon accounting files and `acctcom` to generate a similar ASCII version of `pacct` files.

10.12 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*, Cray Research publication SG–2302.

Procedure 33: 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*, Cray Research publication SG–2302.

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; parameters that pertain to USCP, MPP, and SDS accounting do not apply to CRAY J90:

- 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*, Cray Research publication SG–2302.)
- 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`. If necessary, add the following lines to the `/etc/rc` script in the Accounting script section, just before the `/usr/lib/acct/startup` script section:

```
# Accounting.
#
if [ -x /etc/csaboots ]; then
        echolog "Writing accounting boot time."
        x /etc/csaboots -v >> $RC_LOG
fi
```

3. Ensure that the following lines are included in `/etc/shutdown` to turn off per-process accounting and daemon accounting before the system is brought down:

```
if [ -x /usr/lib/acct/shutacct ]
then
        /usr/lib/acct/shutacct
        echo "Process accounting stopped."
fi
```

4. (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.

5. (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.

6. 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
```

7. 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}
```

8. 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 Research, 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
```

9. 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.

10.13 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*, Cray Research publication SG–2302.
- 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*, Cray Research publication SG–2302.
- 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`.

A sample `/etc/config/acct_config` file follows.

```
#
#   SN5228 - acct_config - Edition 40  [Mon Jan  3 14:34:41 CST 1994]
#   Created by Configuration Generator Rev. 80.60
#
#
#   SN5228 - acct_config - Edition 22  [Mon Nov 15 13:15:11 CST 1993]
#   Created by Configuration Generator Rev. 80.60
#
#
#   SN5147 - acct_config - Edition 8  [Mon Jun  7 14:58:30 CDT 1993]
#   Created by Configuration Generator Rev. 80.60
#
# USMID @(#)skl/etc/config/acct_config   70.8    04/20/93 17:17:19
#
#    (C) COPYRIGHT CRAY RESEARCH, INC.
```

```
#    UNPUBLISHED PROPRIETARY INFORMATION.
#    ALL RIGHTS RESERVED.
#
#
#    This file contains the parameter labels and values used by the
#    accounting software.
#
################################################################################
#
#    Connect time SBUs.
#  The following section contains the labels and values which pertain to
#  connect time accounting.
#
################################################################################
#
# The CON_PRIME parameter defines the weighting factor for prime time
# connect time. This is in billing units per second.
CON_PRIME    0.0
# The CON_NONPRIME parameter defines the weighting factor for nonprime
# time connect time.  This is in billing units per second.
CON_NONPRIME    0.0
################################################################################
#
#    Device accounting SBUs (available only on non-Cray2 systems).
#  The following section contains the labels and values which pertain to
#  device accounting.
#
#
################################################################################
#
# The System Administrator's Guide (SG-2113) describes device accounting
# configuration in detail.
#
# For each device type to be billed, 3 fields must be filled out.
#   1) Logical I/O sbu  -  unit per logical I/O request.
#   2) Characters xfer sbu  -  unit per character transferred.
#   3) Device name  -  Device type name. This field must be surrounded by
#                      double quotes if the name contains embedded spaces.
#    Block devices: The name should match the type assigned to
#      the block device. For example, if the device to be
#      mounted on /tmp is a dd49 with logical device cache,
#      then this device should be labeled as "dd49 with ldcache"
#      or numerically 7.
```

```
#    Character devices: The name should match the major device numbers
#      in /usr/src/uts/c1/cf/devsw.c.
#
# Sites may add new types as long as the number assigned to that type does
# not exceed MAXBDEVNO (for block devices) or MAXCDEVNO (for character
# devices). A site may also change the meaning of the default device types.
#
# MAXBDEVNO and MAXCDEVNO may be increased (see /usr/include/sys/param.h),
# but this will increase the size of the largest possible accounting record.
# This in turn may necessitate more disk space for the pacct files and/or
# a larger kernel stack.
#
#
#    Block device SBUs.
# The numeric suffixes for the "BLOCK_DEVICE" labels must be ascending from
# 0 to MAXBDEVNO - 1. MAXBDEVNO is currently 10.
#
#             Logical       Characters           Device
#   label     I/O Sbu        Xfer Sbu        Name
BLOCK_DEVICE0    0.0        0.0        "dd29"
BLOCK_DEVICE1    0.0        0.0        "dd39"
BLOCK_DEVICE2    0.0        0.0        "dd40"
BLOCK_DEVICE3    0.0        0.0        "dd49"
BLOCK_DEVICE4    0.0        0.0        "dd29 with ldcache"
BLOCK_DEVICE5    0.0        0.0        "dd39 with ldcache"
BLOCK_DEVICE6    0.0        0.0        "dd40 with ldcache"
BLOCK_DEVICE7    0.0        0.0        "dd49 with ldcache"
BLOCK_DEVICE8    0.0        0.0        "ssd"
BLOCK_DEVICE9    0.0        0.0        "bmr"
#
#    Character device SBUs.
# The numeric suffixes for the "CHAR_DEVICE" labels must be ascending from
# 0 to MAXCDEVNO - 1. MAXCDEVNO is currently 35. The suffixes must match
# the minor numbers in /dev. These minor numbers are defined in
# /usr/src/uts/c1/cf/devsw.c in the cdevsw[] array.
#
#             Logical       Characters           Device
#   label     I/O Sbu        Xfer Sbu        Name
CHAR_DEVICE0    0.0        0.0        "hpm"
CHAR_DEVICE1    0.0        0.0        "ios-tty"
CHAR_DEVICE2    0.0        0.0        "systty"
CHAR_DEVICE3    0.0        0.0        "memory"
CHAR_DEVICE4    0.0        0.0        "hyperchannel"
```

```
CHAR_DEVICE5    0.0        0.0        "expander tape"
CHAR_DEVICE6    0.0        0.0        "expander printer"
CHAR_DEVICE7    0.0        0.0        "hsx"
CHAR_DEVICE8    0.0        0.0        "error-device"
CHAR_DEVICE9    0.0        0.0        "expander disk"
CHAR_DEVICE10    0.0        0.0        "low speed channel"
CHAR_DEVICE11    0.0        0.0        "bmx tape"
CHAR_DEVICE12    0.0        0.0        "pty-master"
CHAR_DEVICE13    0.0        0.0        "pty"
CHAR_DEVICE14    0.0        0.0        "?"
CHAR_DEVICE15    0.0        0.0        "bmx daemon"
CHAR_DEVICE16    0.0        0.0        "net"
CHAR_DEVICE17    0.0        0.0        "net"
CHAR_DEVICE18    0.0        0.0        "disk control"
CHAR_DEVICE19    0.0        0.0        "secded"
CHAR_DEVICE20    0.0        0.0        "security log"
CHAR_DEVICE21    0.0        0.0        "cpu control"
CHAR_DEVICE22    0.0        0.0        "logger"
CHAR_DEVICE23    0.0        0.0        "disk maintenance"
CHAR_DEVICE24    0.0        0.0        "data migration"
CHAR_DEVICE25    0.0        0.0        "?"
CHAR_DEVICE26    0.0        0.0        "?"
CHAR_DEVICE27    0.0        0.0        "?"
CHAR_DEVICE28    0.0        0.0        "?"
CHAR_DEVICE29    0.0        0.0        "?"
CHAR_DEVICE30    0.0        0.0        "?"
CHAR_DEVICE31    0.0        0.0        "?"
CHAR_DEVICE32    0.0        0.0        "?"
CHAR_DEVICE33    0.0        0.0        "?"
CHAR_DEVICE34    0.0        0.0        "?"
##############################################################################
#
#    Multitasking CPU time SBUs.
#  The following section contains the labels and values which pertain to
#  multitasking.
##############################################################################
#
# The MUTIME_WEIGHT variables define the weighting factors that
# are used to bill user CPU time for multitasking programs. It
# is used in conjunction with the ac_mutime array (see
# /usr/include/sys/acct.h), which defines the amount of user
# CPU time the multitasking program spent with i + 1 CPUs connected.
#
```

```
# MUTIME_WEIGHTi defines the marginal cost for getting the i-th + 1
# CPU at one instant. If the MUTIME_WEIGHT values are set to less
# than 1.0, there will be an incentive for multitasking. If the
# values are set to 1.0, multitasking programs will be charged for
# user CPU time just as all other programs.
#
# There must be an MUTIME_WEIGHT variable for each of the cpus
# available on the machine.
#
MUTIME_WEIGHT0        1.0        # 1 CPU
MUTIME_WEIGHT1        1.0        # 2 CPU
MUTIME_WEIGHT2        1.0        # 3 CPU
MUTIME_WEIGHT3        1.0        # 4 CPU
MUTIME_WEIGHT4        1.0        # 5 CPU
MUTIME_WEIGHT5        1.0        # 6 CPU
MUTIME_WEIGHT6        1.0        # 7 CPU
MUTIME_WEIGHT7        1.0        # 8 CPU
##############################################################################
#
#    NQS SBUs.
#  The following section contains the labels and values which pertain to
#  NQS accounting.
#
##############################################################################
#
#    Set the values to 1 if jobs, or portion of jobs, which
#    terminate with the specified termination code are to be billed.
#    Otherwise, set the value to 0. By default, all portions of a
#    request will have sbus calculated for them.
#
NQS_TERM_EXIT        1        # Request exited
NQS_TERM_REQUEUE    1        # Request requeued for a restart
NQS_TERM_PREEMPT    1        # Request preempted
NQS_TERM_HOLD        1        # Request held
NQS_TERM_OPRERUN    1        # Request rerun by operator
NQS_TERM_RERUN        1        # Request non-operator rerun




#
```

```
#    Set NQS_NUM_QUEUES to be the number of queues for which you want
#    to set sbus.
#
NQS_NUM_QUEUES        3
#
#    Set the sbus associated with each queues. There must be
#    NQS_NUM_QUEUES sbu/queue pairs. The labels' numeric suffixes
#    must be ascending from 0 to NQS_NUM_QUEUES. Thus, if
#    NQS_NUM_QUEUES is 0, no NQS_QUEUEx values need be defined.
#
#    If an sbu value is set to less than 1.0, there is an incentive
#    to run jobs in this queue. If the value is set to 1.0, the
#    jobs will be charged as though it were a normal, non-NQS job.
#    If the sbu is set to 0.0, there is no charge for jobs running
#    in this queue. For queues not listed below, the sbu is set
#    to 1.0.
#
#  label    sbu    queue_name
NQS_QUEUE0    1.0    b_30_5
NQS_QUEUE1    1.0    b_600_1
NQS_QUEUE2    1.0    b_1200_1
#
#    Set NQS_NUM_MACHINES to the number of originating machines for
#    which you want to set sbus.
#
NQS_NUM_MACHINES        2
#
#    Set the sbus associated with each originating machine. There must
#    be NQS_NUM_MACHINES sbu/machine pairs. The sbu values are set
#    in the same manner as those for the queues. Once again, the
#    numeric label suffixes must be ascending from 0 to NQS_NUM_MACHINES.
#    Thus, if NQS_NUM_MACHINES is 0, no NQS_MACHINEx values need be
#    defined.
#
#  label    sbu    machine_name
NQS_MACHINE0    1.0    sn1405
NQS_MACHINE1    1.0    sn2024
##############################################################################
#
#    Pacct SBUs.
#  The following section contains the labels and values which pertain to
#  pacct (kernel) accounting.
#
```

```
#############################################################################
#
#    Set the prime time weighting factors.
#    On non-Cray2 systems:
#       If P_STIME is nonzero, then P_SCTIME and P_INTTIME must be zero.
#       If P_SCTIME and P_INTTIME are nonzero, then P_STIME must be zero.
#       This is so there won't be multiple billing of system cpu time.
#
P_BASIC        0.0    # Basic prime time weighting factor
P_TIME        0.0    # General time weighting factor
P_STIME        0.0    #   System CPU time weighting factor (unit/sec)
P_UTIME        0.0    #   User CPU time weighting factor (unit/sec)
P_ITIME        0.0    #   I/O wait time weighting factor (unit/sec)
#
#    P_SCTIME and P_INTTIME are used only on non-Cray2 systems.
#
P_SCTIME    0.0    #   System call weighting factor (unit/sec)
P_INTTIME    0.0    #   Interrupt time weighting factor (unit/sec)
P_MEM        0.0    # General memory weighting factor
P_XMEM        0.0    #   CPU time memory weighting factor (unit/Kw-min)
P_IMEM        0.0    #   I/O wait time memory weighting factor (unit/Kw-min)
P_IO        0.0    # General I/O weighting factor
P_BYTEIO    0.0    #   I/O char xfer weighting factor (unit/char xferred)
P_PHYIO        0.0    #   Physical i/o req weighting factor (unit/phy i/o req)
P_LOGIO        0.0    #   Logical i/o req weighting factor (unit/log i/o req)
#
#    The following 3 SDS weighting factors are used only on non-Cray2
#    machines.
#
P_SDSMEM    0.0    # SDS memory integral weighting factor (unit/Mw-sec)
P_SDSLOGIO    0.0    #   SDS logical i/o req weighting factor (unit/log req)
P_SDSBYTEIO    0.0    #   SDS char xferred (unit/char transferred)
#
#    Set the non-prime time weighting factors.
#    On non-Cray2 systems:
#       If NP_STIME is nonzero, then NP_SCTIME and NP_INTTIME must be zero.
#       If NP_SCTIME and NP_INTTIME are nonzero, then NP_STIME must be zero.
#       This is so there won't be multiple billing of system cpu time.
#
NP_BASIC    0.0    # Basic non-prime time weighting factor
NP_TIME        0.0    # General time weighting factor
NP_STIME    0.0    #   System CPU time weighting factor (unit/sec)
NP_UTIME    0.0    #   User CPU time weighting factor (unit/sec)
```

```
NP_ITIME    0.0    #   I/O wait time weighting factor (unit/sec)
#
#    NP_SCTIME and NP_INTTIME are used only on non-Cray2 systems.
#
NP_SCTIME    0.0    #   System call weighting factor (unit/sec)
NP_INTTIME    0.0    #   Interrupt time weighting factor (unit/sec)
NP_MEM        0.0    # General memory weighting factor
NP_XMEM        0.0    #   CPU time memory weighting factor (unit/Kw-min)
NP_IMEM        0.0    #   I/O wait time memory weighting factor (unit/Kw-min)
NP_IO        0.0    # General I/O weighting factor
NP_BYTEIO    0.0    #   I/O char xfer weighting factor (unit/char xferred)
NP_PHYIO    0.0    #   Physical i/o req weighting factor (unit/phy i/o req)
NP_LOGIO    0.0    #   Logical i/o req weighting factor (unit/log i/o req)
##############################################################################
#
#    Tape SBUs.
#  The following section contains the labels and values which pertain to
#  tape accounting.
#
##############################################################################
#
#    The following section sets the sbu values for each of the
#    TP_MAXDEVGRPS tape device groups. TP_MAXDEVGRPS is defined
#    in /usr/include/acct/dacct.h. At this time, only 2 device
#    groups are used: TAPE and CART. However, there must
#    be TP_MAXDEVGRPS "TAPE_SBU" variables defined. The TAPE_SBU
#    numeric suffix must be ascending from 0 to TP_MAXDEVGRPS - 1.
#
#    The fields are:
#      Device_group    Device group name
#      Mount        Billing unit per mount
#      Reserve    Billing unit per reserve second
#      Read        Billing unit per byte read
#      Write        Billing unit per byte written
#
#    Note:  On Cray2 systems, TAPE_SBU0 is always for tape devices,
#      and TAPE_SBU1 is always for cart devices.
#
#
#        Device
#        Group    Mount        Reserve        Read        Write
TAPE_SBU0    TAPE    0.0        0.0        0.0        0.0
TAPE_SBU1    CART    0.0        0.0        0.0        0.0
```

```
TAPE_SBU2    SILO    0.0        0.0        0.0        0.0
TAPE_SBU3    UNUSED    0.0        0.0        0.0        0.0
TAPE_SBU4    UNUSED  0.0             0.0             0.0            0.0
TAPE_SBU5    UNUSED  0.0             0.0             0.0            0.0
TAPE_SBU6    UNUSED  0.0             0.0             0.0            0.0
TAPE_SBU7    UNUSED  0.0             0.0             0.0            0.0
##############################################################################
#
#    USCP SBUs.
#  The following section contains the labels and values which pertain to
#  USCP accounting.
#
##############################################################################
#
#
#    The USCP_MAXMF parameter defines the number of mainframes
#    for which sbus are to be set. This value must be at least 1.
#
USCP_MAXMF    2
```

(Set this to 0 for CRAY J90 systems.)

```
#
#    The following parameters set the sbu values for each of the
#    USCP_MAXMF mainframes. Sbus must be set for each of the
#    US_MAXTTYPE transfer types. US_MAXTTYPE is defined in
#    /usr/include/acct/dacct.h.
#
#    Mainframes not listed below are given sbu values of 0.0.
#
#      USCP_MAINFRAME sets the 2 character mainframe identifier.
#
#      The following parameters set the runtime and sectors transferred
#      sbus for each of the various transfer types. Runtime sbus
#      are in units per second. Sectors transferred (xfer) sbus
#      are in units per sectors transferred.
#
#      USCP_INTER sets the sbus for interactive disposes and fetches
#        (obsolete).
#      USCP_DISPOSE sets the sbus for disposes.
#      USCP_FETCH sets the sbus for fetches.
#      USCP_GET sets the sbus for gets.
#      USCP_PUT sets the sbus for puts.
#      USCP_SAVE sets the sbus for saves.
```

```
#
USCP_MAINFRAME0        SB
#            runtime        xfer
USCP_INTER0        0.0        0.0
USCP_DISPOSE0        0.0        0.0
USCP_FETCH0        0.0        0.0
USCP_GET0        0.0        0.0
USCP_PUT0        0.0        0.0
USCP_SAVE0        0.0        0.0
USCP_MAINFRAME1        YJ
#            runtime        xfer
USCP_INTER1        0.0        0.0
USCP_DISPOSE1        0.0        0.0
USCP_FETCH1        0.0        0.0
USCP_GET1        0.0        0.0
USCP_PUT1        0.0        0.0
USCP_SAVE1        0.0        0.0
################################################################################
#
#    Miscellaneous parameters which are sometimes reset.
# The following section contains miscellaneous parameters that can be
# reset by the site.
#
################################################################################
#
# The ACCT_FS parameter defines the file system on which /usr/adm/acct
# resides. It is used when checking the amount of free space on /usr/adm/acct.
#
ACCT_FS        /usr
#
# The HOLIDAY_FILE parameter defines the location of the holidays file.
# This parameter should be an absolute pathname.
#
HOLIDAY_FILE    /usr/lib/acct/holidays
#
# The MAIL_LIST parameter is a list of users to whom mail is sent
# if errors are detected in the various shell scripts.
#
MAIL_LIST    "root adm"
#
# The MEMINT parameter is used to select the memory integral.
#
MEMINT        2
```

```
#
# The MIN_BLKS parameter sets the minimum number of free blocks on the
# ACCT_FS filesystem that need to be available. If less than MIN_BLKS
# free blocks is available, accounting is disabled, or processing via
# runacct or csarun is halted.
#
MIN_BLKS    500
#
# The NQS_START parameter enables or disables NQS accounting when
# /usr/lib/acct/startup is executed. Valid values are "on" and "off".
# If NQS accounting is enabled here, it must also be enabled by NQS
# via the qmgr(8) "set accounting on" command.
#
NQS_START    on
#
# The NUM_HOLIDAYS parameter sets the upper limit on the number of
# holidays that can be defined in HOLIDAY_FILE.
#
NUM_HOLIDAYS    20
#
# The PERF_NAME0 parameter sets the type which is to be specified with
# devacct(1M) when enabling and disabling performance accounting.
#
PERF_NAME0    perf_01
#
# The TAPE_START parameter enables or disables tape accounting when
# /usr/lib/acct/startup is executed. Valid values are "on" and "off".
# If tape accounting is enabled here, the "-c" option must be used
# when starting the tape daemon, tpdaemon(8).
#
TAPE_START    on
#
# The USCP_START parameter enables or disables uscp accounting when
# /usr/lib/acct/startup is executed. Valid values are "on" and "off".
# Uscp accounting does not need to be enabled by the uscp daemon.
#
USCP_START    on
```

(Set this to off for CRAY J90 systems.)

```
##############################################################################
#
#    Miscellaneous parameters generally not reset.
# The following section contains miscellaneous parameters that are not
```

```
# generally changed by a site. Care must be used when some of these are
# modified.
#
##############################################################################
#
# The A_SSIZE parameter is the maximum number of sessions in 1
# accounting run that can be processed by acctprc1(1M).
#
A_SSIZE        10000
#
# The A_TSIZE parameter is the maximum number of tty line names
# in 1 accounting run that can be processed by acctcon1(1M) and
# csaline(1M).
#
A_TSIZE        1000
#
# The A_USIZE parameter is the maximum number of distinct login
# names in 1 accounting run that can be processed by acctprc1(1M)
# and acctprc2(1M).
#
A_USIZE        5000
#
# The ACCTOFF string is written to /etc/wtmp when accounting is
# turned off by shutacct(1M). This string should be a maximum
# of 11 characters.
#
ACCTOFF        acctg off
#
# The ACCTON string is written to /etc/wtmp when accounting is
# turned on by startup(1M). This string should be a maximum of
# 11 characters.
#
ACCTON        acctg on
#
# The BUILD_MAXFILES parameter sets the upper limit on the number
# of files which can be processed by csabuild(1M).
#
BUILD_MAXFILES    200
#
# The MAX_CPUS parameter sets the upper limit on the number of
# cpus a machine can have. This value must be at least as large
# as the number of cpus on your machine.
#
```

```
MAX_CPUS    32
#
# The MAXICYLS parameter sets the upper limit on the number disk
# cylinders involved in the inode region that must be read by the
# Cray2 version of diskusg(1M). This is an expected worse case
# based on 8000 inode sectors / 16 regions = 512 sectors with the
# dd49's being the smallest at 360 sector/cylinder, requiring 2
# reads per area for an average distribution.
#
MAXICYLS    32
#
# The MAXILIST parameter sets the maximum number of ilist expected
# in a filesystem. This parameter is used only by the Cray2
# version of diskusg(1M).
#
MAXILIST    200
#
# The MAXUSERS_DISK parameter sets the upper limit on the number
# of user id/account id pairs that can be handled by the Cray2
# version of diskusg(1M).
#
MAXUSERS_DISK    5000
# The NCLUSTER parameter sets the upper limit on the number of
# partitions in a filesystem. This parameter is used only by the
# Cray2 version of diskusg(1M).
#
NCLUSTER    100
# The NSYS parameter sets the upper limit on the number of different
# reasons a wtmp record can be written. Generally, these records are
# written by acctwtmp(1M) when accounting is turned on or off.
#
NSYS            20
##############################################################################
#
#    User defined labels.
# The following section contains user defined labels and values.
# These labels are used in the site tailored sbu routines. The
# format of the following lines must be:
#    label        value
#
##############################################################################
```

Adding Your Cray Research System to Your Network [11]

This chapter describes the minimal steps you must take to place your CRAY J90 system on an existing TCP/IP network. After your CRAY J90 system is a functional member of the network, you can do additional configuration. This chapter also includes a table that describes some of the most common TCP/IP configuration files.

After you have placed your CRAY J90 system on an existing TCP/IP network by using the information in this chapter, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, for additional networking configuration information; this publication also includes which menus to use for various networking tasks.

11.1 Related network information

The following publications contain additional information that will be of use to you:

- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022 (man pages):

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

- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011 (man pages):

  `ftp`(1B)

  `hostname`(1)

  `netstat`(1B)

  `telnet`(1B)

- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014 (man pages):

  `gatedconfig`(5)

  `hosts`(5)

  `hosts.equiv`(5)

  `inetd.conf`(5)

  `lo`(4)

  `networks`(5)

  `protocols`(5)

  `rhosts`(5)

  `services`(5)

  `shells`(5)

- The following publications also are recommended, but Cray Research does not provide them:
  - *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture*, Douglas Comer. Prentice Hall, 1991.
  - *UNIX Networking*, S. Kochan and P. Wood. Hayden Books, 1989.
  - *Introduction to the Internet Protocols*, Charles Hedrick, Rutgers University, 1987. Anonymous `ftp` from `cs.rutgers.edu`.
  - *Introduction to the Administration of an Internet-based Local Network*, Charles Hedrick, Rutgers University, 1988. Anonymous `ftp` from `cs.rutgers.edu`.

Procedure 34: Adding a CRAY J90 system to an existing TCP/IP network

To place a CRAY J90 system on an existing TCP/IP network, you should complete the following steps:

Note: To add your CRAY J90 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 system to your existing TCP/IP network.

1. Verify that the network section of the `/sys/param` file contains the proper configuration for your system. See the Network section, Section 5.9.3.5.4, page 69, for details.

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 CRAY J90 system (one per interface), and an entry for at least one other host on the same network as the CRAY J90 system. The entry format is as follows:

   IPaddress host_name annotations

   Example:

```
# cat /etc/hosts

127.0.0.1      localhost loghost

(local host)

128.192.16.8    cray cray-eth

(your CRAY J90 system)

128.192.16.125  cyclone cyclone-eth1

(other host)
```

   Note: Contact your network administrator for the internet address of the Cray system.

3. Compile a binary hosts file.

Cray Research systems support a binary `/etc/hosts` file called `/etc/hosts.bin`. Create this file by using the `/etc/mkbinhost` command, as follows:

```
# /etc/mkbinhost
/etc/hosts.bin: 3 entries written
```

4. Update the `/etc/config/interfaces` file.

   The `/etc/config/interfaces` file defines all network interfaces on the Cray Research system. Change the host name for each interface on your system to match those you chose in step 2; for additional information, see the `initif`(8) man page and the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304. The entry format is as follows:

   interface_name _ family address ifconfig parameters

   Example:

   Note: All host names must be defined in your `/etc/hosts` file.

```
# 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 `/etc/initif` script; an example follows:

```
# /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 Research system. To reach hosts that

are not on the same FDDI or Ethernet network as the CRAY J90 system, you must have a route. To create a route, execute the /etc/route command, as shown in the following example:

```
# /etc/route add default otherhost
add net default: gateway otherhost
```

The *otherhost* is the IP address (or *name* as shown in the `/etc/hosts` file) of a host that is on the same network as the CRAY J90 system and connects to one or more additional networks.

Place this command in the `/etc/tcpstart.mid` script so that it will be run automatically at system startup.

7. Test the network.

   Test the network by using the `ping` command and view the configuration by using the `netstat` command. The `ping` command tests whether you can reach another host on the network. If `ping` succeeds, you can be confident that the hardware and routing works on all hosts and gateways between you and the system to which you are sending `ping`. The `netstat` command has many options. The `-i` option lets you view a table of cumulative statistics for transferred packets, errors, and collisions for each interface that was autoconfigured. The interfaces that are statically configured into a system but are not located at boot time are not shown; the `-r` option lets you view the routing table. The network address (currently Internet-specific) of the interface and the maximum transmission unit (mtu) in bytes also are displayed. You should become familiar with how these displays look on your system so that you will recognize changes and problems immediately.

   Examples:

```
# /etc/ping otherhost
PING otherhost : 56 data bytes
64 bytes from 123.123.12.13: icmp_seq=0. time=10. ms
CONTROL-c
```

```
# netstat -i
Name     Mtu   Network          Address         Ipkts      Ierrs    Opkts     Oerrs
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 CRAY J90 system cannot access that network.

8. Reboot the system.

   Reboot the system to verify that all of your changes are handled automatically during system startup. (For booting procedures, see Chapter 3, page 19.) At some point, you should again test the network by using `ping` and view the configuration by using `netstat`.

9. Transfer full configuration files from another system, as shown in the following example.

Save copies of your original files and add the new entries for the CRAY J90 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.
# /etc/mkbinhost
/etc/hosts.bin: 2675 entries written
```

Note: You should check for entries for the J90 system in the `/etc/hosts` file. If they are not there, see Procedure 34, step 2, page 277.

Your CRAY J90 system should now be on the network.

11.2 Domain name service (DNS)

If you want to use domain name service (DNS) to perform host name lookup, you should configure your CRAY J90 system as a caching-only server. This should be done for the following reasons:

- A caching-only server is more efficient than a remote server (resolver only) because it maintains a cache of data and, therefore, requires less frequent network access.
- A caching-only server does not have authority over a particular zone, therefore, it does not have to answer queries from other authoritative servers.

- A caching-only server does not have to load configuration files from disk like a primary master server or across the network like a secondary master server, which gives it a faster start-up time.

To configure your Cray Research system as a caching-only server, you may configure both resolver and the local name server. You can perform most of this configuration by using the UNICOS Installation / Configuration Menu System.

Procedure 35: Configuring a caching-only server by using the menu system

Note: If you have not completed the previous procedure, "Adding a CRAY J90 system to an existing TCP/IP network," you should complete steps 3, 4, and 5 of that procedure before configuring a caching-only server.

To configure your CRAY J90 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 the UNICOS system will use DNS, rather than the `/etc/hosts` file to look up host names. A sample `TCP/IP Host/Address Lookup Configuration` menu screen follows:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
```

```
                TCP/IP Host/Address Lookup Configuration

    Use Domain Name (DN) service ?              YES
S-> DNS lookup (resolver) ==>
    Local DN server (named) ==>
```

2. Configure the resolver, which consists of creating the `/etc/resolv.conf` file, which is created if you place information in the `DNS lookup (resolver)` menu. When you have a local name server (`named` process) running, you should have the local host address (127.0.0.1) as the first name

server; otherwise your local named will be bypassed, resulting in decreased performance and increased network traffic.

The following is an example of the menu screen:

```
       TCP/IP Domain Name Service Lookup (resolver) Configuration
    Local domain name                          cray.com
    Address for Domain Name server #1:         127.0.0.1
S-> Address for Domain Name server #2:         128.162.19.7
    Address for Domain Name server #3:         128.162.19.13
    Address for Domain Name server #4:
    Address for Domain Name server #5:
    Address for Domain Name server #6:
    Address for Domain Name server #7:
    Address for Domain Name server #8:
    Address for Domain Name server #9:
```

3. Configuring a caching-only local name server consists of creating the named.boot, root.cache, and localhost.rev files. You can do this by using the Local DN server (named) menu. Perform the following steps to create these files:

   a. The named.boot file is read when named starts up. It tells the server what kind of server it is, over which zones it has authority, and where to get its initial data. The following is an example of the menu screens:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
```

```
                TCP/IP Host/Address Lookup Configuration

    Use Domain Name (DN) service ?              YES
    DNS lookup (resolver) ==>
S-> Local DN server (named) ==>
```

```
            TCP/IP Local Domain Name Server Configuration

S-> Directory for name server files              /etc/named.d
    Address of forwarding name server #1         128.162.19.7
    Address of forwarding name server #2         128.162.19.13
    Address of forwarding name server #3         128.162.1.1
    Slave server?                                NO
    Root name server cache file                  root.cache
    Root name server cache ==>
    Primary zones ==>
    Secondary zones ==>
```

b. The local name server also needs configuration information for the zones for which it is the primary server (the zones for which it has authority). On a caching-only server, the only zone for which the local `named` has authority is the `0.0.127.IN-ADDR.ARPA` zone. This information is stored in the `localhost.rev` file; you can configure its name, but not its contents, by using the menu system:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
                    ->TCP/IP Root Nameserver Cache Config
```

```
              TCP/IP Root Nameserver Cache Configuration

     Server name     Server address  Time To Live
     --------------  --------------  ------------
E->  earth.cray.com  128.162.3.55    1000000
```

```
              TCP/IP Root Nameserver Cache Configuration

S-> Server name                              earth.cray.com
    Server address                           128.162.3.55
    Time to live                             1000000
```

c. The local name server must know the name of the server that is the authoritative name server for the domain. The `root.cache` file is used

to "prime the cache" with this information, and you can configure it by using the menu system:

```
Configure System
    ->Network Configuration
        ->TCP/IP Configuration
            ->TCP/IP Host/Address Lookup Configuration
                ->TCP/IP Local Domain Name Server Config
```

```
               TCP/IP Domain Name Service Primary Zones

    Name                  File           Account  Serial #
    --------------------  -------------  -------  --------
E-> 0.0.127.IN-ADDR.ARPA  localhost.rev
```

```
               TCP/IP Domain Name Service Primary Zones

    Zone name                                  0.0.127.IN-ADDR.ARPA
    File to contain zone information           localhost.rev
S-> Account name of responsible party
    Serial number for zone
```

4. Start the `named` daemon by executing the following command:

```
# /etc/sdaemon -s named
```

Procedure 36: Configuring a caching-only server without using the menu system

Note: If you have not completed the previous procedure, "Adding a CRAY J90 system to an existing TCP/IP network," you should complete steps 3, 4, and 5 of that procedure before configuring a caching-only server.

To configure your CRAY J90 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 the UNICOS system 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
```

11.3 Common TCP/IP configuration files

After you have the Cray Research system on the network, you should do a few additional things to make sure it is a fully functional member of your network, including configuring `inetd`, adding additional routes, and updating other configuration files. If you are using the menu system, you should update these files by using the menu system options. Table 3 describes some of the most common TCP/IP configuration files. The *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, describes all of these files.

Table 3. TCP/IP configuration files

File (relative to `/etc`)	Description	Change
`config/daemons`	Lists system and network daemons to start at system boot.	Probably
`gated.conf` or `tcpstart.mid`	Contains routes to be installed at system boot.	Yes
`config/hostname.txt`	Contains text host name for TCP/IP.	Probably not
`hosts`	Maps Internet addresses to host names.	Yes
`hosts.equiv`	Lists trusted hosts for `rlogin`, `rsh`, and so on.	Optional
`hycf.` *xxx*	Maps physical addresses to Internet addresses for nonbroadcast media (for example, HYPERchannel and HIPPI).	Yes, if you have HYPERchannel or HIPPI
`inetd.conf`	Lists network services to be handled by `inetd`.	Probably not
`config/interfaces`	Lists network interfaces and their characteristics.	Yes
`networks`	Maps network names to network Internet addresses.	Optional
`protocols`	Maps protocol names to protocol numbers.	No
`$HOME/.rhosts`	Lists trusted users for `rlogin`, `rsh`, and so on.	Optional
`services`	Maps protocol and port numbers to service names.	Probably not
`shells`	Lists shells allowed for `ftpd`.	Probably not

Configuring NIS [12]

12.1 Related NIS documentation

The following documentation contains information covered in this chapter:

- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `netstart`(8), `udbgen`(8), `ypbind`(8), `ypinit`(8), `yppasswdd`(8), `yppush`(8), `ypserv`(8), `ypstart`(8), and `ypxfr`(8) man pages
- *UNICOS User Commands Reference Manual*, Cray Research publication SR–2011: `domainname`(1), `udbsee`(1), `yppasswd`(1), and `ypwhich`(1) man pages

12.2 What is NIS?

The Network Information Service (NIS) is a network service that allows information such as passwords and group IDs for an entire network to be held in one database. (NIS was formerly known as Yellow Pages.)

Cray Research also supports a new naming service called Network Information Service Plus (NIS+), developed by SunSoft, Inc., a Sun Microsystems company. NIS+ is one of a suite of technologies that make up Open Network Computing Plus (ONC+), a SunSoft product and technology concept. NIS+ is separately licensed (as part of ONC+). The NIS product continues to be provided with the UNICOS release under the UNICOS license. Only the new NIS+ product requires the separate ONC+ license. For more information on NIS+, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

Implemented with the Remote Procedure Call (RPC) and eXternal Data Representation (XDR) library routines, UNICOS NIS has the following features:

- Look-up service: UNICOS NIS maintains a set of databases that can be queried through the use of pointers, or "keys." Programs can request the value associated with a particular key, or all of the keys, in a database.

- Network service: Programs do not have to know the location of data or how it is stored. Instead, they use a network protocol to communicate with a database server that contains the information.
- Distributed service: Databases are fully replicated on several machines, known as "NIS servers." The servers propagate updated databases among themselves, ensuring consistency.

The UNICOS NIS environment includes at least one Cray Research system and one or more other hosts that also run NIS.

NIS databases contain maps; a *map* contains information that is usually found in an ASCII configuration file. Each map contains a set of keys and associated values. For example, the `passwd` map contains user names (the keys) and their associated `/etc/passwd` file entries (the values). The NIS maps are stored in `dbm` format. The `makedbm` command converts an ASCII file into a `dbm` format file that NIS can use. Usually, you do not have to worry about `dbm` format or the `makedbm` command. To generate the maps, you will use the makefile in the `/etc/yp` directory. For further information on the internal map format, see the `dbm`(3) and `makedbm`(8) man pages.

Cray Research supports the following maps on systems running the UNICOS operating system:

Map	Description
`group`	Performs the function of the `/etc/group` file: mapping group names to group IDs.
`netgroup`	Defines networkwide groups used for permission checking when doing remote mounts, remote logins, and remote shells.
`passwd`	Performs a few selected functions of the UDB on UNICOS systems; namely, password, home directory, and shell lookup.
`publickey`	Used for secure RPC, the `publickey` map contains public key/private key pairs for users and hosts on the network. For more information about running secure RPC, see the NIS chapter in the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

An important distinction to make is that a Cray Research system running the UNICOS system can **serve** other NIS maps for its domain; however, the Cray Research system (as a client) **consults** only the preceding maps. For more information about supported maps, see the NIS chapter in the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

An *NIS domain* is a specified set of NIS maps. The set of maps for a given domain is stored in a directory named after the domain. To assign hosts to a particular domain, use the `domainname` command.

Servers provide resources; clients use them. There are two types of NIS servers: master servers and slave servers. The *master server* contains the NIS maps. You can change NIS maps only on the master server. A *slave server* contains copies of the NIS maps that it obtains from the master server for its domain.

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

Note: You should configure your CRAY J90 system as an NIS slave server.

This section contains procedures for the following:

- Using the menu system to configure your CRAY J90 system as an NIS slave server
- Configuring your CRAY J90 system as an NIS slave server without using the menu system
- Configuring user accounts to use NIS

UNICOS NIS differs from the NIS facility used on other systems based on the UNIX system. Administration of an NIS domain that includes a Cray Research system is different from administration of an NIS domain that does not. For example, UNICOS NIS does not support the broadcast feature. If you are unfamiliar with NIS, you should first read the NIS documentation for your other systems. After you have familiarized yourself with the general NIS mechanism, read the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304, to familiarize yourself with UNICOS NIS before you configure NIS on your CRAY J90 system.

Procedure 37: Using the menu system to configure your CRAY J90 system as an NIS slave server

Note: This procedure assumes that another host on the network is already configured as an NIS master server and that your CRAY J90 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. For information about starting system daemons, see Chapter 4, page 45. If you have an ONC+ license, start the `rpcbind` daemon command by executing the `/etc/rpcbstart`(8) or `/etc/rpcbind` command. This daemon provides support for universal addressing. For more information on the `rpcbind` daemon, see the *Remote Procedure Call (RPC) Reference Manual*, Cray Research publication SR–2089, and the `rpcbind`(8) and `rpcbstart`(8) man pages.

Before you configure NIS on your CRAY J90 system, read the cautionary note and other important information on pages Section 12.2, page 289.

The following steps use the menu system to configure your CRAY J90 system as an NIS slave server:

1. Enable the menu system to configure NIS. To give the menu system permission to change the configuration for NIS, ensure that the `NIS configuration` option is set to `YES` in the `Configure System->Configurator Automation Options` menu. Also, ensure that the `Configure System->Major Software Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

2. Assign your CRAY J90 system to an NIS domain.

   Select the `Configure System->Network Configuration->NIS Configuration` menu. Enter the NIS domain name, and then activate the NIS configuration. A sample menu screen follows:

```
Configure System
    ->Network Configuration
        ->NIS Configuration
```

```
                        NIS Configuration

S-> NIS domain name
    Import the NIS configuration ...
    Activate the NIS configuration ...
```

3. Run the `/etc/yp/ypinit` script with the `-s` option to configure the CRAY J90 system as an NIS slave server. Include the host name of the NIS master for your domain on the command line, as follows:

```
# /etc/yp/ypinit -sNISmasterserver
```

Running `ypinit -s` *NIS_masterserver* causes a copy of the NIS maps to be transferred from the master to the slave server (your Cray Research system running the UNICOS system) and placed in the `/etc/yp/`*domainname* directory. Running this command also adds the slave server to the `ypservers` map for your domain.

Note: You must run `ypinit` only once, when you first install the host as a slave server. After that, you can perform map updates by using either the `yppush` command from the master server or the `ypxfr` command from the slave server.

4. Start the NIS daemons, `ypserv` and `ypbind`.

Note: The procedure for starting NIS daemons differs from the procedure for starting other system daemons.

An administrator (`root`) can start all daemons manually from the command line. To start the `ypbind` daemon, use the `-h` option, as follows:

```
# /etc/ypserv
# /etc/ypbind -h yourCRAY J90ohostname
```

You may start the daemons manually when you first install NIS to verify that everything is working. After that, the daemons will be started automatically each time the system boots because when you activated the menu in step 1, your NIS domain name was written into the `/etc/config/ypdomain.txt` file. At system startup, the `/etc/ypstart` script accesses the `/etc/config/ypdomain.txt` file, automatically sets the NIS domain name, and then starts the `ypserv` and `ypbind` daemons.

You **must not** add the daemons to the `/etc/config/daemons` file. The `ypstart` script assumes that you are running the Cray Research system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

5. Verify that the CRAY J90 system has bound to itself by using the `ypwhich` command. It is normal to see that the domain has not bound the first time you execute `ypwhich`; simply enter it a second time, as follows:

```
# ypwhich
Domain domainname not bound.
# ypwhich
yourCRAY J90system
```

Procedure 38: Configuring your CRAY J90 system as an NIS slave server without using the menu system

Note: This procedure assumes that another host on the network is already configured as an NIS master server and that your CRAY J90 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. For information about starting system daemons, see Chapter 4, page 45.

Before you configure NIS on your CRAY J90 system, read the cautionary note and other important information on pages Section 12.2, page 289.

The following steps explain how to configure your CRAY J90 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 CRAY J90 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 CRAY J90 system as an NIS slave server. Include the host name of the NIS master for your domain on the command line, as follows:

```
# /etc/yp/ypinit -s NISmasterserver
```

   Running `ypinit -s` *NISmasterserver* transfers a copy of the NIS maps from the master to the slave server (your Cray Research system running the UNICOS system) and places it in the `/etc/yp/`*domainname* directory. Running this command also adds the slave server to the `ypservers` map for your domain.

   Note: You must run `ypinit` only once, when you first install the host as a slave server. After that, you can perform map updates by using either the `yppush` command from the master server or the `ypxfr` command from the slave server.

4. Start the NIS daemons, `ypserv` and `ypbind`.

   Note: The procedure for starting NIS daemons differs from the procedure for starting other system daemons.

   An administrator (`root`) can start all daemons manually from the command line. To start the `ypbind` daemon, use the `-h` option, as follows:

```
# /etc/ypserv
# /etc/ypbind -h yourCRAY J90hhostname
```

   You may start the daemons manually when you first install NIS to verify that everything is working. After that, you should configure the daemons so that they are started automatically each time the system boots; see "To have NIS start automatically when you start UNICOS" at the end of this procedure.

5. Verify that the CRAY J90 system has bound to itself by using the `ypwhich` command. It is normal to see that the domain has not bound the first time you execute `ypwhich`; simply enter it a second time, as follows:

```
# ypwhich
Domain domainname not bound.
# ypwhich
yourCRAY J90system
```

To have NIS start automatically when you start UNICOS

To start NIS automatically when you start the UNICOS system, 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 the UNICOS system in the future, the `/etc/ypstart` script will access the `/etc/config/ypdomain.txt` file, set the NIS domain name automatically, and then start the `ypserv` and `ypbind` daemons. You **must not** add the daemons to the `/etc/config/daemons` file. The `ypstart` script assumes that you are running the Cray Research system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

Procedure 39: Configuring user accounts to use NIS

Note: This procedure assumes that your CRAY J90 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 171). The `yppasswd` command works only if you have started the `yppasswdd` daemon on the NIS master server machine.

Configuring NFS [13]

13.1 Related NFS documentation

The following documentation contains information covered in this chapter:

- *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022: `automount`(8), `biod`(8), `cnfsd`(8), `exportfs`(8), `mount`(8), `mountd`(8), `nfsd`(8), `nfsidmap`(8), and `sdaemon`(8) man pages
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014: `exports`(5) and `fstab`(5) man pages

13.2 What is NFS?

The network file system (NFS) is a Cray Research software product that allows users to share directories and files across a network of machines.

NFS users can use standard I/O system calls, commands, and permission controls to access files from any file system. Similarly, other NFS users can make use of file systems by using remote commands from anywhere in the local network environment. You can use NFS in diverse administrative environments through the use of the ID mapping facility (see Section 13.3, page 300). 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 J90 system as an NFS client and as an NFS server.

For additional information about NFS, including information about the following topics, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304:

- NFS automounter (`automount`(8)(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 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 NIS, a given user or group ID always refers to the same user or group within the administrative domain. This allows all hosts in the NFS group to interpret the authentication information passed in the NFS requests in the same way. Traditional NFS environments make use of NIS (formerly called Yellow Pages) to achieve a flat administrative space. NIS is a distributed look-up service that maintains a common database of UID and GID information for members of an administrative domain. An NIS domain is one administrative domain.

If your Cray Research system resides entirely within one NIS domain and does not interact with hosts outside that domain, you probably will not have to configure NFS ID mapping. However, Cray Research systems are often shared by many different administrative domains, making the creation of a single, flat administrative space for user and group identification technically and/or organizationally difficult. Because a given ID can refer to different users or groups in different administrative domains, this would prevent NFS from being used in such an environment, or would cause serious security problems.

The Cray Research system NFS ID mapping facility allows different administrative domains to participate in cross-mounting NFS file systems without creating a single, flat administrative space.

Following is a description of circumstances in which it is desirable and circumstances in which it is necessary for the Cray Research NFS server to access ID mapping information:

- Account IDs, or ACIDs, which are unique to the UNICOS system, 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) chapter in the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304.

Procedure 40: Configuring a CRAY J90 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 J90 system as an NFS client:

1. Make entries in the `/etc/fstab` file that describe the file systems you want mounted using NFS. You can use the menu system to do this step or you can do this manually.

   If you are using the menu system, you must first enable the menu system to configure NFS. To give the menu system permission to change the configuration for NFS, change the `NFS configuration` option to `YES` in the `Configure System->Configurator Automation Options` menu. Also, ensure that the `Configure System->Major Software`

`Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

Then, select the `Configure System->File System (fstab) Configuration->NFS File Systems` menu, add your entries, and update the form file. Then activate your changes through the `File Systems (fstab) Configuration` menu. A sample `Network File System Configuration` menu screen follows:

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

```
   Network File System Configuration
     Host    Name              Mount         RW  Quota  Suid  Auto  Bg  So
    -------  --------          ------------  --  -----  ----  ----  --  -- >
E-> tngmoon  /usr/bin          /UTNA/sunbin  ro                     bg  so
```

If you are not using the menu system, edit the `/etc/config/rcoptions` file and set the `RC_NFS=` parameter to `YES`. Then make entries in the `/etc/fstab` file that describe the file systems you want mounted using NFS. The following example shows sample entries; for more information about the options, see the `fstab`(5) and `mount`(8) man pages and *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304:

```
# cat fstab
#
# Mainframe file system table (fstab)
#
#       There are six fields per line, separated by white space.
#       1.  device name
#       2.  filesystem name
#       3.  filesystem type
#       4.  mount options
#       5.  dump frequency
#       6.  pass number to check file system
#
/dev/dsk/root   /                   NC1FS  rw    1    1
/dev/dsk/home   /home               NC1FS  rw    1    2
/dev/dsk/core   /core               NC1FS  rw    1    2
/dev/dsk/usr    /usr                NC1FS  rw    1    2
/dev/dsk/src    /usr/src            NC1FS  rw    1    2
#
# NFS file systems
#
tngmoon:/home/tngmoon/user1     /UTNA/user1     NFS     ro,soft,bg
tngmoon:/usr/bin                /UTNA/sunbin    NFS     ro,soft,bg
```

2. Start the `biod` daemon, which is an optional client daemon that handles write-behind and read-ahead requests. Although this daemon is optional, you should run it to improve NFS performance. By default, four `biod` daemons are started; you might improve client performance by running more `biod` daemons. Ensure that the `biod` daemon is started by using the menu system or by doing it manually.

   Note: To configure NFS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group. For information about starting system daemons, see Chapter 4, page 45.

   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 the UNICOS system. 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 the UNICOS system in the future.) Then execute the `/etc/sdaemon` script to start `biod` now, as follows:

```
# /etc/sdaemon -s biod
```

Note: You cannot do the remaining steps to this procedure by using the menu system.

3. If you do not want to configure UNICOS NFS ID mapping at this time, you should disable this feature by executing the following command (for information on UNICOS NFS ID mapping, see the *UNICOS Networking Facilities Administrator's Guide*, Cray Research publication SG–2304:

```
# /etc/uidmaps/nfsidmap -d
NFS ID mapping is disabled.
```

Note: To disable NFS ID mapping permanently, place the `/etc/uidmaps/nfsidmap -d` command in the `/etc/uidmaps/Set.domains` file; otherwise, if you want NFS ID mapping disabled, you must execute 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 the UNICOS system, 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 the UNICOS system in the future, the script will be run automatically:

```
# /etc/mountnfs
[1] 85260
[2] 85261
```

8. Verify that the file systems have been mounted by using the `/etc/mount` and `df` commands, as shown in the following examples:

```
# /etc/mount
/ on /dev/dsk/root read/write on Mon Jan 10 08:45:33 1994
/tmp on /dev/dsk/tmp read/write on Mon Jan 10 08:46:11 1994
/usr on /dev/dsk/usr read/write,rw on Mon Jan 10 08:46:12 1994
/home on /dev/dsk/home read/write,rw on Mon Jan 10 08:46:13 1994
/usr/src on /dev/dsk/src read/write,rw on Mon Jan 10 08:46:13 1994
/proc on /proc read/write on Mon Jan 10 08:46:14 1994
/els_src on /dev/dsk/els_src read/write on Mon Jan 10 09:14:58 1994
/UTNA/sunbin on tngmoon:/usr/bin read only,ro,soft,bg on Mon Jan 10 18:13:41 1994
/UTNA/user1 on tngmoon:/home/tngmoon/user1 read only,ro,soft,bg on Mon Jan 10 18:13:41 1994
```

```
# df
/UTNA/user1  (tngmoon:/home/tngmoon/user1):
                                 100955  1K blocks ( 48.2%)
/UTNA/sunbin (tngmoon:/usr/bin ):  65879  1K blocks ( 50.7%)
/els_src     (/dev/dsk/els_src ): 178106  4K blocks ( 59.4%)*   57677 I-nodes
/proc        (/proc            ): 119400  4K blocks ( 95.5%)      412 procs
/usr/src     (/dev/dsk/src     ):  27132  4K blocks ( 18.1%)*   25654 I-nodes
/home        (/dev/dsk/home    ): 671031  4K blocks ( 97.8%)*  169883 I-nodes
/usr         (/dev/dsk/usr     ): 152964  4K blocks ( 59.0%)*   26694 I-nodes
/tmp         (/dev/dsk/tmp     ): 495065  4K blocks ( 98.8%)*   98292 I-nodes
/            (/dev/dsk/root    ):  13417  4K blocks ( 17.9%)*   16169 I-nodes
```

Procedure 41: Configuring a CRAY J90 system as an NFS server

The following steps explain how to configure a CRAY J90 system as an NFS server:

1. Describe the file systems you want to allow other systems to mount using NFS by placing entries in the `/etc/exports` file. (For a complete list of export options, see the `exports`(5) man page.) You can use the menu system to place your entries in the `/etc/exports` file or you can do this manually.

   If you are using the menu system, ensure that the menu system has permission to change the configuration for NFS by verifying the `NFS configuration` option is set to `YES` in the `Configure System->Configurator Automation Options` menu. Also, ensure that the `Configure System->Major Software Configuration` menu

has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

Then select the `Configure System->Network Configuration->NFS Configuration->List of Exported File Systems` menu, add your entries, and update the form file. Then activate your changes through the `NFS Configuration` menu. A sample `NFS Exported File Systems Configuration` menu screen follows:

```
Configure System
    ->Network Configuration
        ->NFS Configuration
            ->List of Exported File Systems
```

```
  NFS Exported File Systems Configuration
    File System  Clients  ro  rw clients  Anon  UID  Root Clients  Sum  Ker
    -----------  -------  --  ----------  ----  ---  ------------  ---  --- >
E->  /home                 rw              anon  -2   edge
```

If you are not using the menu system, ensure that the `RC_NFS=` parameter is set to `YES` in the `/etc/config/rcoptions` file. Then edit the `/etc/exports` file to describe the file systems you want to allow other systems to mount using NFS.

2. Look at the list of these file systems by using the `cat /etc/exports` command, as shown in the following example:

```
# cat /etc/exports
/nasc   -root=edge:sn1234
/home   -rw
/tmp
/UTNA/goodstuff
```

3. Make all file systems described in the `/etc/exports` file available to NFS client systems by using the menu system or doing it manually.

   If you are using the menu system, select the `Configure System->System Daemons Configuration->System Daemons Table` menu, set the `Start up at boot time?` option to `YES`, and update the form file. Then activate your changes through the `System`

`Daemons Configuration` menu. A sample `System Daemons Table` menu screen follows:

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

```
                System Daemons Table

S-> Group                                          NFS
     Name                                           exportfs
     Start up at boot time?                         YES
     Kill action                                    *
     Executable pathname                            /etc/exportfs
     Command-line arguments                         -av
     Additional command-line arguments
     Additional command-line arguments
```

If you are not using the menu system, you can make all file systems described in the `/etc/exports` file available to NFS client systems by executing the `/etc/exportfs -av` command, as follows:

```
# /etc/exportfs -av
```

Caution: If you are not using the menu system, you **must** run the `/etc/exportfs -av` command each time your system is rebooted. You should automate the execution of this command by using the menu system.

4. Enable the NFS server daemons; at a minimum, the TCP/IP `portmap` daemon and the NFS `nfsd` and `mountd` daemons must be started on an NFS server. You can use the menu system to enable the daemons or you can do this manually. The `cnfsd` daemon is necessary **only** when you have more than one Cray Research system; it is intended for use only between Cray Research systems. For more information, see the `exports`(5) and `cnfsd`(8) man pages.

   If you are using the menu system, select the `Configure System->System Daemons Configuration->System Daemons`

`Table` menu, set the NFS server daemons to `YES`, and update the form file. Then activate your changes through the `System Daemons Configuration` menu. When you activate this change, the daemons will be started automatically each time you start the UNICOS 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 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
```

Frequently Used Commands [A]

This appendix provides a brief description of several frequently used system administration commands, scripts, and files. 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*, Cray Research publication SR–2011
- *UNICOS System Calls Reference Manual*, Cray Research publication SR–2012
- *UNICOS File Formats and Special Files Reference Manual*, Cray Research publication SR–2014
- *UNICOS Administrator Commands Reference Manual*, Cray Research publication SR–2022
- *CRAY IOS-V Commands Reference Manual*, Cray Research publication SR–2170

A.1 Commands available from the IOS console

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 CRAY J90 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 CRAY J90 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`	The UNICOS system configuration file (`/ios-param` is a copy of this file). A CRAY J90 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`.

A.2 Commands available from the UNICOS console

The following commands and scripts are available from the UNICOS console:

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/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/init`	A command that signals the `init` process to change to a different run level.
`/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 . 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 [B]

In the course of normal system administration, occasions exist when it is important to make a new version of a file, but it is also important to keep an old file. A sequence of operations often used to replace a real production file with a new one and keep an old version of a file is as follows:

```
cp file.REAL file.NEW
edit file.NEW
cp file REAL file.OLD
mv file.NEW file.REAL
```

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
```

Cleaning Tape Units [C]

This appendix contains information about standard use and care procedures for the tape hardware on CRAY J90 systems.

Note: For complete information, see the documentation provided with your tape subsystem.

C.1 Cleaning the digital audio tape (DAT)

When the right Clean/Attention light on the digital audio tape (DAT) flashes amber, you should clean the tape heads. To clean the heads, use Cleaning Cartridge 90334800, which you can order from Logistics (or you can use the HP 92283K Cleaning Cartridge). The Cray Research cleaning cartridge package includes two tapes.

To use the cleaning cartridge, follow these steps:

1. Insert the cleaning cartridge into the drive. The drive automatically takes the cartridge, loads, it, and cleans the heads.
2. After about 30 seconds, the drive ejects the cartridge.

   If the cartridge is ejected after only about 14 seconds, this means the cartridge has reached the end of its useful life, and no cleaning has occurred. Discard the cartridge, and repeat the cleaning operation with a new cleaning cartridge.
3. Take the cartridge out of the drive, and write the date on the label on the cartridge. A cartridge usually has a life of 25 cleaning cycles.

If the Cleaning Needed signal reappears, the cartridge is nearing the end of its useful life; copy the data on the cartridge onto a new one and discard the old cartridge. After you have cleaned the heads successfully, the Cleaning Needed signal will be cleared.

Rather than waiting for the Cleaning Needed signal to appear on the front panel, you should clean the heads according to the following table:

Tapes used per day	<1	1	2	3	4	5
Cleaning interval	weekly	weekly	twice weekly	twice weekly	daily	daily

C.2 Cleaning the 3480 (StorageTek 4220)

The StorageTek 4220 subsystem displays a message on the LED when cleaning is required (the word CLEAN appears on the display). After the host processor unloads the current data cartridge, insert the cleaning cartridge (shipped with the system). The device recognizes the cleaning cartridge and initiates the cleaning cycle. The cleaning cartridge is unloaded after the 15-second cleaning cycle completes.

You should replace the cleaning cartridge after 500 uses.

C.3 Cleaning the 9-track tape (StorageTek 9914)

The StorageTek 9914 streamer requires no preventative maintenance, but it does require routine cleaning. You should clean the heads daily (if the system is used continuously). To clean, pull the device fully out from its rack, release the three thumbscrew fasteners, and raise the tape path cover to its fullest. Use a cleaning material and solvent as recommended in the subsystem documentation.

Disk Capacities and Transfer Rates [D]

This appendix contains information about disk device capacities and transfer rates. The UNICOS `diskel`(7) man page also contains these specifications.

D.1 DD-5I disk drives

The DD-5I disk drive , supported on CRAY J90 systems, 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.

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

D.2 DD-5S disk drives

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

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

D.3 DD-6S disk drives

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

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

D.4 DD-314 disk drives

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

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

D.5 DD-318 disk drives

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

Table 8. DD-318 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	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	3,711

Logical Device Cache Process [E]

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 J90 systems, logical cache is defined in central memory (DDRAM).

When a process issues a read request of data on a file system, the action taken to access the data depends on whether the data is currently in the UNICOS system buffer cache or logical device cache. The process is described as follows:

1. If the data is found in the system buffer cache (central memory), it is copied to the user area. If the requested data is not found, step 2 is taken.
2. If 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.

E.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.

E.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
- ---- ----  ---------- --------  --------  -------  -----  -----
B  300   27    16727    30354      34865     1799     95.09  /dev/dsk/root
B  300   27    1729      4703       1399      254     84.63  /dev/dsk/home
B  250   27    6702     20794       6191      263     95.93  /dev/dsk/tmp
#
M  200   10      47        11         27        4     87.10  /dev/dsk/src

# ldcache -bCache to user    Cache to disk     Cache/disk ratio
Reads  Writes    Reads  Writes    Read  Write  Total    Name
-----  ------    -----  ------    ----  -----  -----    ----
839155 334505    65016  28772    12.9  11.6  12.5    /dev/dsk/root_b
301039  26871    25616   1628    11.8  16.5  12.0    /dev/dsk/usr_b
 68947  74725    13424  13824     5.1   5.4   5.3    /dev/dsk/spool
   183   1678    18416   1743     0.0   1.0   0.1    /dev/dsk/usr_tmp

# ldcache -l /dev/dsk/tmp /dev/dsk/tmp   Fri Sep 24 14:52:12 1993
```

A hit rate of under 97% probably indicates that the file system is not a good candidate for ldcaching or that you should enlarge the size of that file system's ldcache area if possible. In the preceding display, you should examine the file system usage and ldcache configuration aspects of the `/dev/dsk/home` and `/dev/dsk/src` file systems. You also should examine the ratio of logical reads to physical reads, as shown in the preceding display of the `ldcache -b` example.

An example of displaying ldcache statistics for an individual file system follows:

```
                          Read data     Write data
                          ---------     ----------
Blocks transferred:           689            1296
Avg request length:         1 blks           1 blks
Lst transfer rate:    0.008192 Mbs     0.061236 Mbs
Max transfer rate:    0.135680 Mbs     0.208438 Mbs
Cache hits:                   597             677
Cache misses:                   0              73
Cache hit rate:        1000.000000       90.266667
```

E.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 19.

Power Up and Down Procedures [F]

This appendix includes the following procedures:

- How to power up and power down a CRAY J90 system mainframe cabinet

F.1 Powering up/down a CRAY J90 system

This section contains procedures for powering up and powering down a CRAY J90 system.

F.1.1 Powering up a CRAY J90 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 3, page 337).
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 3) 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 4, page 338).

10. Enter `load` at the boot prompt (>). ACT "first load" then scrolls up.

11. Close doors of all cabinets.

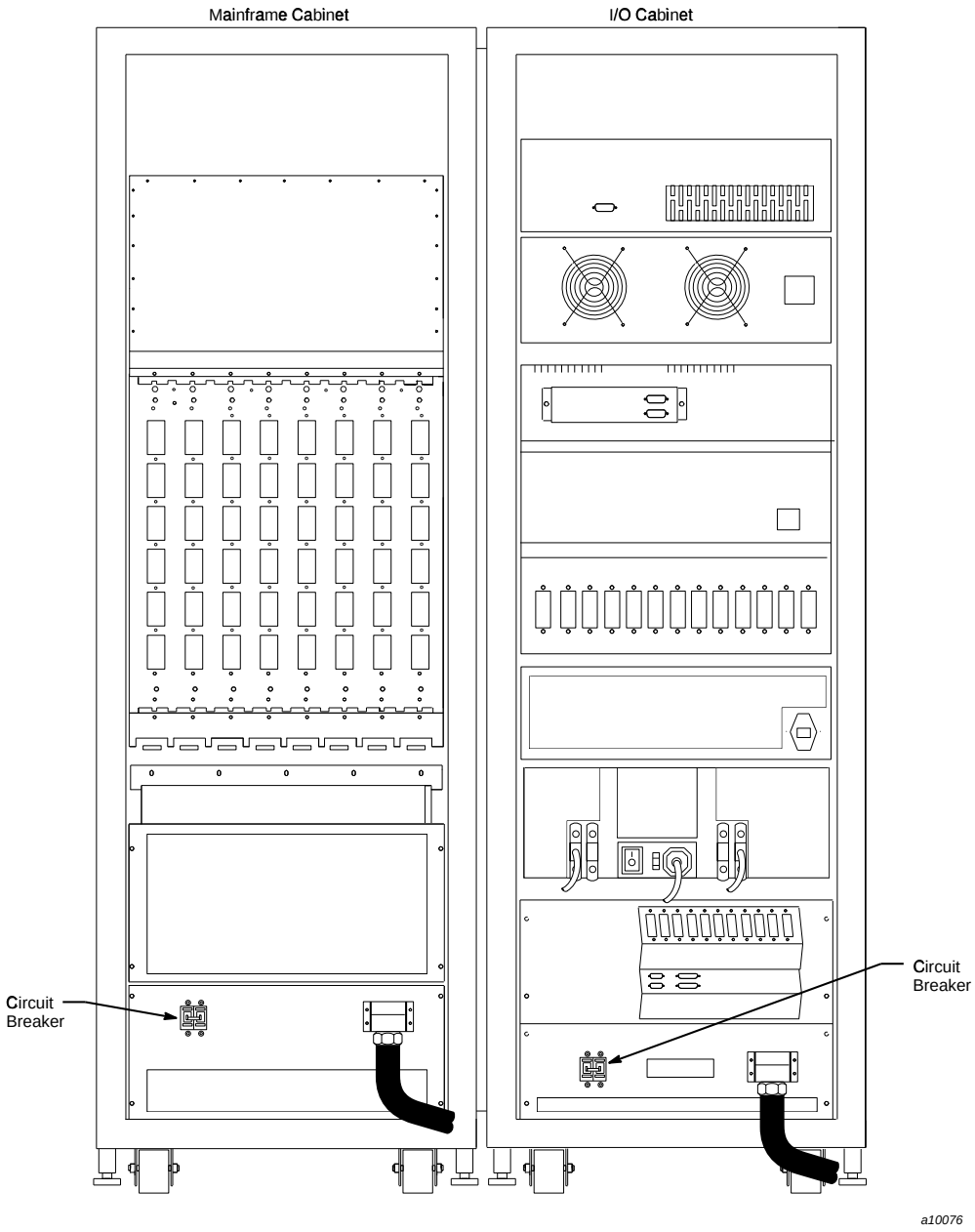

Figure 3. AC circuit breakers

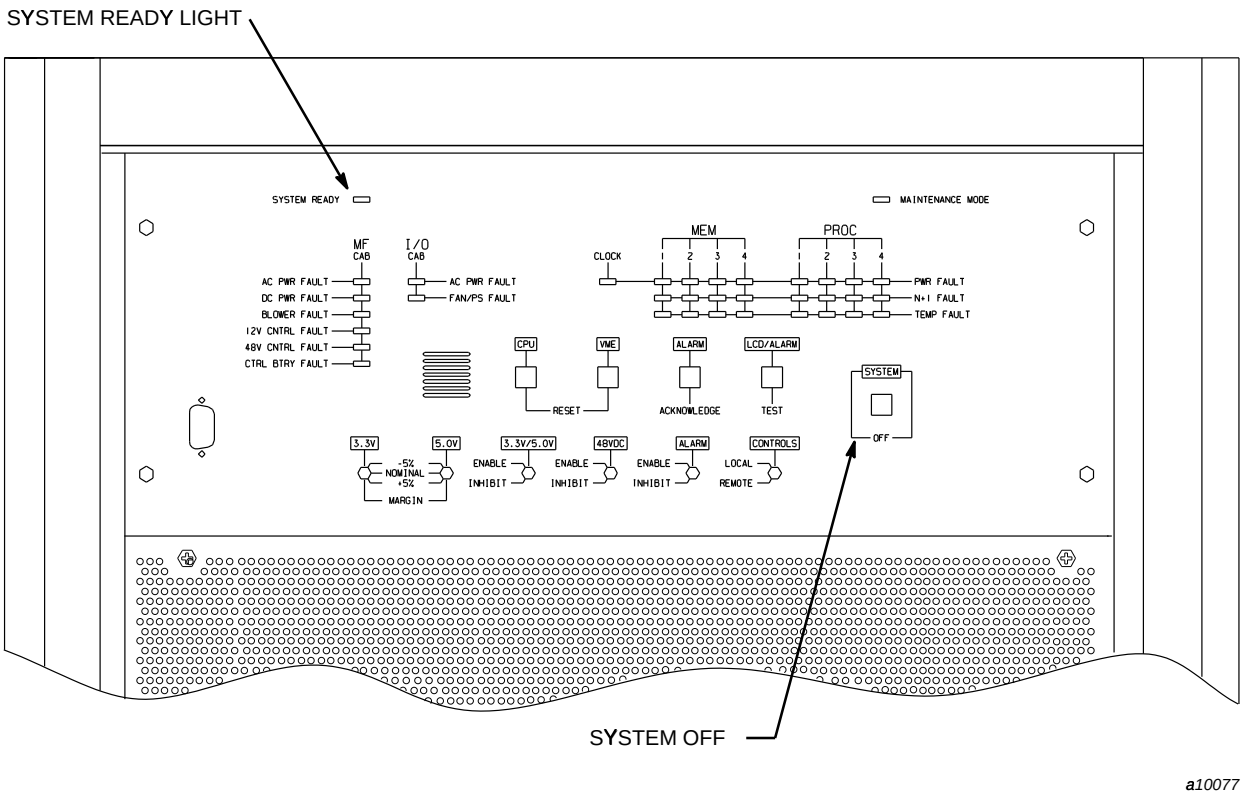

Figure 4. CCU

F.1.2 Powering down a CRAY J90 system

To power down a CRAY J90 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 [G]

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

IOS and Mainframe Dump [H]

If your CRAY J90 series system experiences an IOS assertion panic, IOS processor panic, or a UNICOS system panic, you should perform an IOS and mainframe dump so that the data in your system's memory is saved into a file. These dumps are very important and useful for determining probable causes for software or hardware problems. This chapter explains the procedures for capturing memory dumps on CRAY J90 systems, both from the IOS and from the mainframe (UNICOS).

H.1 Send dump results to Cray Research

Place dumps on digital audio tapes (DAT) media in `tar`(1) or `cpio`(1) format and send them to Software Product Support at Cray Research corporate headquarters in Eagan, Minnesota, USA for analysis. For tracking purposes, include the CRUISE ticket number and/or the SPR number for the incident. The `unicos` and crash files that are created at the same time as the dump file become the core file system and also should be forwarded for analysis.

H.2 IOS dump for IOS-V

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>
```

H.2.1 Dumping a slave IOS

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
```

H.3 UNICOS dump

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 345. 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*, Cray Research publication SQ–2172, for message explanations.

See the *CRAY IOS-V Commands Reference Manual*, Cray Research publication SR–2170, or the online man pages for descriptions of IOS commands.

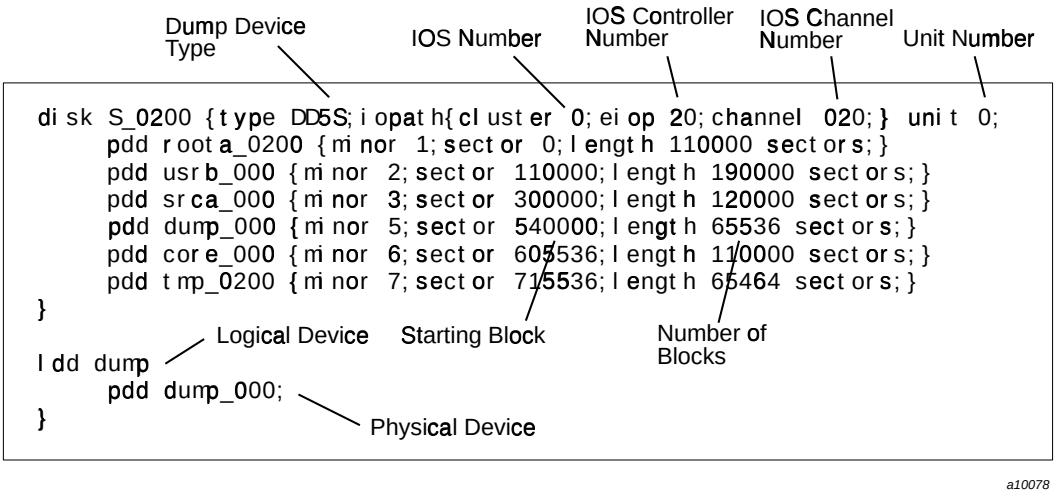

Figure 5. Dump entry example from IOS `/sys/param` file

H.4 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.

H.4.1 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
```

H.5 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 Section H.5, page 347 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.

```
/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
```

Example 11: Sample console output when the UNICOS system is booted in multiuser mode

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

Index

D

E

F

S

T

U

V

W