General UNICOS® System Administration

S–2301–10011

New Features

General UNICOS® System Administration S–2301–10011

This revision of *General UNICOS System Administration*, supports the 10.0.1.1 release of the UNICOS operating system.

The following changes were made:

- Chapter 8, page 145, Section 8.4.1.2.2, page 175

  The following statement was added: “For sites that have users running at multiple labels, `/usr/preserve/`*LOGIN_NAME* must be created as an MLD or a wildcard directory.”

- Entire Book

  Various minor miscellaneous corrections were made throughout the Book.

Record of Revision

Version	*Description*
9.0	August 1995 Original printing. Documentation supports the administration of UNICOS 9.0 release running on Cray computer systems. This manual contains the contents of and supersedes the information formerly provided in sections “File System Planning,” “Startup and Shutdown Procedures,” “File System Maintenance,” “Basic Administration,” “Crash and Dump Analysis,” UNICOS Multilevel Security (MLS) Feature,” “Administration of Online Documentation,” and appendixes “Job and Process Recovery USENIX Paper,” “Security Attributes in Trusted UNICOS,” and “Covert Channels in Trusted UNICOS” in *UNICOS System Administration*, publication SG-2113 8.0.
9.2	December 1996 Documentation supports administration of the UNICOS 9.2 release running on Cray computer systems.
9.3	August 1997 Documentation supports administration of the UNICOS 9.3 release running on Cray computer systems.
10.0	November 1997 Documentation supports administration of the UNICOS 10.0 release running on Cray computer systems.
10.0.0.3	October 1998 Documentation supports administration of the UNICOS 10.0.0.3 release running on Cray computer systems.
10008	November 2000 Documentation supports administration of the UNICOS 10.0.0.8 release running on Cray computer systems.
10010	October 2001 Documentation supports administration of the UNICOS 10.0.1.0 release running on Cray computer systems.
10011	May 2002 Documentation supports administration of the UNICOS 10.0.1.1 release running on Cray computer systems.

Contents

Page

Page

Page

Page

Page

Page

Page

Page

Page

Page

Page

Figures

Page

Tables

Preface

This guide documents the UNICOS operating system running on Cray computer systems. It provides information needed to perform basic administration tasks as well as information needed for administering the multilevel security (MLS) feature.

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

UNICOS System Administration Publications

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

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

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

Related Publications

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

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

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

- *Cray L7R Release Overview and Software Installation Guide*, contains information on the Cray L7R release and details regarding software installation and configuration for the Cray L7R. This publication contains chapters

documenting an overview of the release, purpose and function of the Cray L7R, system and network configuration requirements, software installation and configuration instructions, and troubleshooting.

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

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

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 Documentation

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

E-mail:
`orderdsk@cray.com`

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

Click on the `Cray Publications Order Form` link.

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

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

Fax:
+1–651–605–9001

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

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items, such as file names, pathnames, man page names, command names, and programming language elements.
`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
8	Administrator commands

	Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.
variable	Italic typeface indicates an element that you will replace with a specific value. For instance, you may replace *filename* with the name `datafile` in your program. It also denotes a word or concept being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a syntax representation for a command, library routine, system call, and so on.
. . .	Ellipses indicate that a preceding element can be repeated.

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

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems.
Cray MPP systems	All configurations of the Cray T3E series. The UNICOS operating system is not supported on Cray T3E systems. Cray T3E systems run the UNICOS/mk operating system.

Reader Comments

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

E-mail:
`swpubs@cray.com`

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

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

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

Introduction to System Administration [1]

This manual is a teaching and reference document for people who manage the operation of Cray computer systems running the UNICOS operating system. This chapter provides an introduction to the manual and includes sections covering the following topics:

- A chapter-by-chapter description of the contents of this manual
- A brief introduction to the UNICOS multilevel security (MLS) feature and *Cray ML-Safe* configuration of the UNICOS system
- A summary description of the user exits available in the UNICOS operating system

1.1 Overview of Contents

This guide provides the following information:

- Descriptions of the general characteristics of UNICOS file systems.
- Descriptions of the startup and shutdown of a UNICOS system, including system initialization, system shutdown, run-level configuration, and multiuser startup.
- Information on the administration and maintenance of file systems, including mounting and unmounting file systems, file system backup and restoration, and file system checking and repair.
- Descriptions of tools and methods used in the daily administration of a UNICOS system. This includes information about executing commands at specified times (the `at` and `cron` commands), communicating with users, monitoring system security, and job and process recovery.
- Description of the user database (UDB). The UDB is the Cray enhancement of the traditional UNIX `/etc/passwd` and `/etc/group` files. The user database contains an entry for each user allowed to log in to and run jobs on your system; it provides faster access to this information than the traditional UNIX password and group files, and it allows safe use of multiple sources when user information is changed.
- Guidelines on crash and dump analysis for troubleshooting and debugging purposes. Although it does not address all potential problems, this guide describes general system crash analysis and recovery.

- Information on the UNICOS multilevel security (MLS) feature (also referred to as *security enhancements*) and on the Cray ML-Safe configuration of the UNICOS system. MLS provides mechanisms to protect both system integrity and sensitive information.
- The administrative procedures required for the online glossary and the online message system.
- Appendixes covering the following topics:
  - User-defined locales, which are collections of culture-dependent information used by an application to interact with a us
  - A description of the development and design of the UNICOS job and process recovery facility

This manual replaces neither experience nor other documents that more fully describe specific system areas. Familiarity with the references listed in the preface, combined with the full-time efforts of an individual, is necessary to effectively manage Cray computer systems running the UNICOS operating system.

1.2 UNICOS Multilevel Security (MLS) Feature and the Cray ML-Safe Configuration

The UNICOS multilevel security (MLS) feature provides specific mechanisms and assurances to protect both system integrity and sensitive information.

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

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

The MLS feature and the Cray ML-Safe configuration are described in Chapter 8, page 145. It is assumed that before you read this chapter, you have read the *UNICOS Multilevel Security (MLS) Feature User's Guide*.

In addition, the MLS feature is referenced throughout this guide as appropriate.

1.3 User Exits

In the UNICOS operating system you can customize the programming environment in accordance with local needs by means of *user exits*. A user exit allows you to introduce local code at a defined point, even if you do not have access to the source code. User exits are compatible from release to release.

User exits are provided in the following areas:

- General Administration

  At system startup, the `brc`(8) script, which invokes system initialization shell scripts, executes the `rc.pre`, `rc.mid`, and `rc.pst` scripts at specified points. These scripts allow for local startup initialization that cannot be accomplished through the existing `rc` script. These user exits are described in "System multiuser startup," Section 3.4, page 77, and on the `brc`(8) man page.

  At system shutdown, the `shutdown`(8) script provides three user exits at specified points: the `shutdown.pre`, `shutdown.mid`, and the `shutdown.pst` scripts, which allow you to provide additional shutdown processing that is not provided by the `shutdown` script. These user exits are described in Section 3.2.1, page 67, and on the `shutdown`(8) man page.

- System Call

  The `uesyscall` system call template allows you to write system calls specific to your site. This features allows you to access kernel structures not otherwise available and to provide kernel support for local UNICOS functions. This feature is documented in Section 5.6, page 127.

- Accounting

  Cray system accounting provides four user exits at specified times during the execution of the `csarun`(8) daily accounting shell script. For instructions for setting up the user exits scripts, see *UNICOS Resource Administration*.

- Fair-share Scheduler

  You can customize the fair-share scheduler's CPU scheduling policy at your site. This allows use of a different scheduling algorithm without modifying the UNICOS kernel. See *UNICOS Resource Administration* for information on this feature.

- Network Queuing Environment (NQE)

  Through the NQS batch processing subsystem, user exits are provided to tailor various NQS functions to your site. NQS user exits allow you to

customize queue destination selection, `qsub` option preprocessing, job startup and termination processing, and NQS startup and shutdown processing. You must keep your UNICOS build and your build for NQS user exits separate.

For more information, see *NQE Administration*.

- MLS

  The MLS feature includes seven user exits as part of its centralized identification and authentication (I&A) mechanism. These user exits allow customers to control some aspects of user I&A, including I&A success and failure processing. Some examples are as follows:

  – Support of a local password format

  – Allowing validation information to be held on a remote (that is, front-end) host

  – Disallowing multiple logins

  – Bypassing password processing

  – Limiting access to a selected group of users or selected network address

  – Disabling `su` to `root`

  Warning: The user exit feature does not meet the requirements of a Cray ML-Safe system.

  For information on user exits in MLS, see Section 8.5.4, page 210.

- Tape subsystems

  User exits are available that allow you to add special routines to communicate with the tape daemon. These user exits allow a system process to examine and modify a structure associated with a tape file. The specific user exits available for the tape subsystem, and instructions for their implementations, are described in the *Tape Subsystem Administration*.

- Unified Resource Manager (URM)

  URM contains code that allows you to customize URM for the specific needs of your site by writing your own job selector. In URM, this locally-written selector code is executed last.

  As shipped, the selector code is empty; it defaults to exiting without doing anything. However, you can write your selector based on any ranking algorithm you choose, and insert your code into the URM user exit.

URM user exits and their implementation are described in *UNICOS Resource Administration*.

File System Planning [2]

All of the files that are accessible from within the UNICOS operating system are organized into *file systems*. File systems store data in formats that the operating system can read and write.

Warning: This chapter contains warnings and information critical to the use of a Cray ML-Safe configuration.

This chapter discusses several aspects of administering file systems. This chapter is organized as follows:

- Introduction to UNICOS file systems
- File system concepts
- Using the `mkspice`(8) command (IOS-E and IPN-1 systems only)
- Creating file system nodes
- Configuring disk arrays
- File system initialization
- Inode allocation strategies
- Inode region allocation
- Labeling a file system
- Mirrored file systems
- Performance considerations

2.1 Introduction to UNICOS File Systems

The following sections introduce some general characteristics of UNICOS file systems. The following topics are discussed:

- File system overview
- File system types
- File system strategies

2.1.1 File System Overview

The file is the logical unit of data storage within the UNICOS operating system. Files are grouped into structures called *file systems.* The root file system contains the base or root of the file system tree. Other file systems are logically attached to (*mounted*) and detached from (*unmounted*) the root file system by the super user.

A file system is typically made up of slices of one or more disk devices. However, a file system can also reside totally or in part in memory.

2.1.2 File System Types

The exact format of the file system data is determined by the hardware architecture, the `mkfs`(8) options used to make the file system, and the version of the UNICOS operating system under which the file system was made. The following list summarizes the types of file systems:

Type	Description
`NC1FS`	UNICOS file system on Cray PVP systems
`NFS`	Network file system (NFS)
`SFS`	Shared file system
`PROC`	`/proc` file system
`INODE`	`/inode` file system

`NC1FS` file systems are the standard UNICOS file system on Cray PVP systems. The maximum size for an `NC1` file system is 8 Tbytes.

File systems of the type `NFS` reside on a remote server, have been mounted under the UNICOS operating system, and can be accessed through NFS. For information on administering NFS file systems, see the *UNICOS Networking Facilities Administrator's Guide.*

The `SFS` file system is a file system that can be shared among multiple Cray systems. To use the UNICOS Shared File System (SFS) feature you must obtain a software license, which is maintained and administered through the Flexible License Manager (FLEXlm) product. For information on the UNICOS SFS feature, see *Shared File System (SFS) Administrator's Guide.*

The `/proc` file system is intended to be used by the debugging utilities to debug running processes and, to a lesser extent, as an interprocess communication mechanism. The `/proc` file system is a special file system that consists of a directory in which all of the processes present in the system appear as regular

files. Processes can then be read and written as though they were simple disk files.

The chown(1), chmod(1), and link(8) operations are prohibited in the /proc file system. Users may modify the regular files (representing processes) that they own. The super user may modify all files.

The /proc file system does not consume any disk space or kernel buffers. Instead, it is produced directly from information that is maintained in the kernel process table and is constantly changing as processes are created and destroyed in the system. The /proc file system never needs to be checked for damage by any of the file system repair utilities (for instance, fsck(8)) because such damage is impossible.

The output of the df(1) command presents different information for the /proc file system than for other file systems. The number listed under % disk space used refers to the percentage of memory in use. The number listed under % free refers to the percentage of memory available at the present time.

For more information about the /proc file system, see proc(4).

The /inode file system allows privileged processes access to a file or directory when the process knows the device and inode number of a file system. For more information about the /inode file system, see inode(4).

2.1.3 File System Strategies

No one configuration of available disk drives into file systems will prove best for all purposes. In addition, as the needs of users change, the file system layout will most likely need to be reconfigured occasionally. In the absence of a set of absolute rules, the following facts and guidelines should prove useful when you choose the file system layout for your system.

- When organizing disks into file systems, you should first consider attributes of the user population. These attributes can provide a logical division, such as the following:
  - Location
  - Project
  - Applications
  - File-storage requirements
  - Security considerations

By considering these factors, you can significantly enhance system performance.

- A large file system allows the maximum amount of resource sharing. However, a large file system takes far longer to dump and restore than several small file systems. In some cases, smaller file systems are desirable to separate users and reduce disk contention. In addition, a file system that uses a small number of disks has a lower risk of losing data because of disk failures.

  There is no universal solution to deciding on the best file system size. In most cases, a compromise is needed to maximize performance while maintaining file system recoverability.

- Because each file is contained in a single file system, a file can be no larger than the size of its file system.

- A user can take disk space from a file system only if it contains a directory (or file) with write permissions for the user. This usually means that a user can take space only in the file system that contains the user's login directory and in the file systems of shared temporary directories `/tmp` and `/usr/tmp`. Therefore, two users can avoid competition over disk space if they have login directories on different file systems and if they do not use the shared temporary directories. Many UNICOS commands use temporary directory space, but the location of the space can be controlled by the `TMPDIR` shell variable.

- You should also consider *throughput*, or the file transfer rate, when dividing disks into file systems. The two relevant factors are hardware transfer rate and head-positioning overhead.

  The hardware transfer rate is fixed and sets a limit on data transfer speed, but several disks can be combined into a single file system to provide a higher aggregate transfer rate.

  Head-positioning overhead is a problem when a single disk is split into several partitions and those partitions are active simultaneously. The time wasted in moving the head between the two partitions can decrease the effective transfer rate.

- To maintain optimum performance, you should configure your system with no more than one slice per physical device for each file system. This reduces head movement and channel contention.

2.2 File System Concepts

The following sections provide an overview of the basic file system concepts. The information is presented in the following order:

- Disk organization
- Disk flawing (IOS-E and IPN-1 only)
- Disk striping
- Disk mirroring
- Physical devices
- Simple logical devices
- Striped logical devices
- Mirrored logical devices
- Logical device descriptor files

2.2.1 Disk Organization

Disk drives are divided into sectors. The sectors are numbered, starting with 0. Each sector can be identified by its sector number.

As a result of SPR 718787, the disk specification information for UNICOS based Cray systems can be found only in the `diskspec`(7), `disksmpn`(7), and `disksfcn`(7) man pages as follows:

`disksfcn`(7)	Physical specifications of disk devices connected to the Fibre Channel I/O Node (FCN-1).
`diskspec`(7)	Physical specifications of disk devices connected to the IPI-2 I/O Node (IPN-1).
`disksmpn`(7)	Physical specifications of disk devices connected to the Multipurpose I/O Node (MPN-1).

Disk drives are divided into sectors. The sectors are numbered, starting with 0. Each sector can be identified by its sector number.

A *track* is a circle on the disk that the disk head can read in a single revolution of the disk without moving. The density of the disk determines how many sectors are in a track. The tracks are also numbered, starting from 0.

A *cylinder* is the group of tracks, one from each platter surface, with the same track number. The number of tracks per cylinder is determined by the number of surfaces within the disk device.

Note: Disk devices connected to the multipurpose node 1 (MPN-1) and disk devices connected to the Fibre Channel I/O Node (FCN-1) do not organize disks into tracks or cylinders, but only into sectors.

For a summary of the physical characteristics of the disk drives connected to UNICOS based Cray systems, see the `diskspec`(7), `disksmpn`(7), and `disksfcn`(7) man pages.

Disk drives are divided into *slices*, which are contiguous groups of sectors. This division is specified during disk configuration. For each slice there is a corresponding physical device node.

A *partition* is part of a file system that consists of a single slice of a disk device.

A collection of slices from one or more physical drives make up a *logical device*. Logical devices are also specified at disk configuration. For each logical device there is a corresponding logical device node.

Disk drives on IOS-E based systems and disk drives on the IPN-1 must have a slice of cylinders reserved for use by the customer engineer (*CE cylinders*) and a slice reserved as *spares cylinders* to be used in place of other cylinders of the disk that become unusable or *flawed*.

2.2.2 Disk Flawing (IOS-E and IPN-1 Only)

Disk drives have minute defects, or *flaws*, on the recording surface that may interfere with reading and writing data. The number of flaws and their locations vary from drive to drive. Each disk device is shipped with a *factory flaw table* that lists known flawed blocks on the disk. Disk flawing under the UNICOS system is done on a physical device basis, by replacement of bad blocks with alternative blocks from the spares cylinders.

The information for bad blocks is kept separate from the file system on each physical device and is known only to the driver. The advantage of this mechanism is that the logical device always appears contiguous, which allows file systems to be transferred to various logical devices without regard for bad blocks.

Flawing is done twice; once before a physical device is placed in the system and again if blocks go bad during online use. Typically, flawing is done either before

a new drive is placed online for the first time or once for each device when an initial system install is performed.

For information on creating physical disk device inodes that describe the spare sector map, factory flaw map, and diagnostic and customer engineering slices, see Section 2.3, page 15.

2.2.3 Disk Striping

Striping is designed for moving large amounts of data at very high bandwidths, higher than one can normally achieve with existing disk drives. With this technique, several drives are combined into one logical unit. Data in *n*-sector size pieces is written to and read in from the drives in a round-robin fashion, allowing I/O to the individual disks to be overlapped. The drives of a stripe group must all be the same type; for example, you cannot have a stripe group of two DD-49 disk drives and one DD-40 disk drive.

The overlapping of I/O operations causes the increased bandwidth. Each drive in the *stripe group* (group of striped disks) can have an I/O operation active simultaneously at any time. For example, consider a four-drive stripe group to which you want to write several tracks of data. A write request for a track is started for the first drive then a write request for the second track to second drive is started without waiting for the first write to complete, and so on. The I/O operations are asynchronous between drives in the stripe group.

The disadvantage to striping is that sequential data is scattered across several disks. Losing any one of the disks of the stripe group ruins the striped files (for a four-drive stripe group with a striping factor of one track, you would miss every fourth track).

In most cases, the only device that you should use for striping is SWAPDEV, as it is usually the only device with I/O requests large enough for striping to be advantageous. You should not use striping for `/root` or `/usr` file systems.

2.2.4 Disk Mirroring

Mirroring is used to provide data redundancy when data integrity is important. It is implemented by using two to eight slices, usually on as many different physical disks, each of the same size. A write operation to a mirrored device causes separate write operations to be performed on each of the components. A read operation may be performed on any of the component devices.

For instructions on creating mirrored devices, see Section 2.4.4.4, page 26. For a description of mirrored file systems, see Section 2.10, page 46.

2.2.5 Physical Devices

There are four types of physical-level devices, each with its own device driver, as follows:

- Disk drive
- Solid-state disk
- RAM disk
- Network or HIPPI disk device

By convention, disk drive, solid-state disks, and RAM disk device files are kept in the `/dev/pdd` directory and HIPPI disk device files are kept in the `/dev/hdd` directory. Disk drives connected to the GigaRing channel are kept in the `/dev/xdd` directory.

For instructions on creating physical devices, see Section 2.4.1, page 17.

2.2.6 Simple Logical Devices

A *logical disk device* is a collection of blocks on one or more physical disks or other logical disk devices. A *logical direct device* indicates that the logical disk includes exactly one partition or physical slice. A *logical indirect device* indicates that the logical disk includes more than one partition or physical slice.

Logical drivers call the physical drivers by using the device switch mechanism. By convention, logical device files are kept in the `/dev/dsk` directory.

The simple logical device is the highest level driver. Any mountable file system will interface with the driver at this level. Logical device cache is supported only at this level.

For information on creating logical devices, see Section 2.4.4, page 23.

2.2.7 Striped Logical Devices

A striped logical device is a means of combining two or more slices of two different physical devices together to increase bandwidth. This is best for heavily used partitions like the swap device where very large chunks of data are being transferred.

Striped logical devices consist of physical device name lists, and can be combined into simple logical devices. Striped logical device routines can be called directly

with the open(2), close(2), read(2), write(2), and ioctl(2) system calls or from another logical device driver.

By convention, striped logical device files are kept in the /dev/sdd directory.

For information on creating striped logical devices, see Section 2.4.4.3, page 25.

2.2.8 Mirrored Logical Devices

A mirrored logical device is used to provide data redundancy where data integrity is important. It consists of two or more slices, typically on as many different physical devices, each of the same size. A mirrored logical device is similar in configuration to a striped device. A write operation to a mirrored device causes separate write operations to be performed on each of the components. A read operation may be performed on any of the component devices.

Mirrored logical devices are made up of lists of physical device names, and can be combined into simple logical devices. Mirrored logical device routines can be called directly with the open, close, read, write, and ioctl system calls, or from another logical device driver, but not concurrently.

By convention mirrored logical device files are kept in the /dev/mdd directory.

For more information on mdd files, see the mdd(4) man page. For instructions on creating mirrored devices, see Section 2.4.4.4, page 26. For a description of mirrored file systems, see Section 2.10, page 46.

2.2.9 Logical Device Descriptor Files

A logical device descriptor file is used to combine one or more character special disk files to form a single logical disk device. A logical device descriptor file is a list of absolute path names of striped, mirrored, physical, and HIPPI devices. By convention, logical descriptor files are kept in the /dev/ldd directory.

For instructions on creating logical descriptor files, see Section 2.4.4, page 23.

2.3 Using the mkspice(8) Command (IOS-E and IPN-1)

When configuring a disk for the first time, use the mkspice(8) command to create physical disk device inodes that describe the spare sector map, factory flaw map, and diagnostic and customer engineering slices. You name the physical disk devices according to their I/O paths. The following example creates the

spare, ift, diagnostic, and ce slices for DD-XXs on cluster 0, IOP 1, channel 30; cluster 1, IOP 2, channel 32; and cluster 1, IOP 3, channel 34:

```
/etc/mkspice -t ddXX 0130 01232 01334
```

The -i option of the mkspice(8) command initializes the spare maps from the ift nodes. It also initializes the /etc/aft files (ASCII flaw files) which are used with the bb(8) (bad block) command. See aft(5) for information on the aft files and bb for information on creating the bad block files from the aft files.

The -i option of the mkspice(8) command invokes the ift(8) command to read the Factory Flaw table from the /dev/ift node. The following is an example of ift output.

```
*
*  engineering flaw table for DDXX
*
*  factory flaw map date: 10-08-86
*
*       S/N     C2236

#       0 0 0 0

*       count   head    sector  cylinder

        1       3       0       1333
        1       3       1       1333
        1       3       24      1333
```

The -i option of the mkspice(8) command also invokes the spmap(8) command, which generates and writes a physical disk spare map. The spmap command reads flaw information from standard input in the same format written by the ift(8) command. The output from ift can be piped directly into spmap, or the output from ift can be written to an ASCII Flaw table and then piped into spmap.

It is recommended that you create the ASCII Flaw tables. If a new flaw develops, it must be added to the end of the ASCII Flaw table with a text editor. The spmap command then needs to be rerun. If the ASCII Flaw table is lost, you should recreate it, using spmap to ensure that the ordering of the alternate blocks is preserved. For example:

```
/etc/spmap -r /dev/spare/0130 > /etc/aft/0130
```

2.4 Creating File System Nodes

Disk partitions and logical devices are defined with the mknod(8) command. Only the root and swap partitions are defined in the parameter file during startup.

This section provides information on the following areas of creating file systems:

- Creating physical devices
- Examples of physical device creation
- Creating physical devices (GigaRing systems)
- Creating logical devices
- Creating logical descriptor files
- Defining alternate disk paths
- Shared dump and swap configuration

2.4.1 Creating Physical Devices

> **Note:** For information on creating physical devices on GigaRing based systems, see Section 2.4.3.

A disk slice is defined by an I/O path, a unit number, a starting sector number, and a length in sectors. Slices must be aligned on track boundaries, and in practice they are often aligned on cylinder boundaries.

To create a partition, you must first use the mknod(8) command to create a character special file or node and specify a major and minor device number for each slice you want to create. The major device number should be expressed symbolically.

Symbol	Device
dev_pdd	Disk devices.
dev_rdd	RAM disk devices.
dev_ssdd	SSD devices.

The minor device number ranges from 0 to PDDSLMAX for disk devices, 0 to RDDSLMAX for RAM disk devices, and 0 to SSDDSLMAX for SSD devices.

The following minor device numbers have specific designations:

Number	Designation
250	The root partition on the fsload tape.
251	The swap partition in the initial UNICOS kernel.
252	The diagnostic partitions.
253	The ce partitions (customer engineering).
254	The spare partitions (Factory Flaw table).
255	The ift partitions. The number 255 is special in that only nodes with this minor device number can be used on the ioctl(8) command to bring a disk device up after it has gone down.

Only one node of a minor device number can be open at any given time. With the exception of the diagnostic, ce, and ift nodes, do not create two disk nodes with the same minor device number.

The physical device driver fills in internal driver structures and checks for conflicts in the device open routine. An array of slice structures are indexed by the minor device number within a given physical device driver.

The following partitions can be opened by the system without using the nodes in /dev (but may still be accessed through nodes):

- root
- swap

The index, offset, and length of these partitions is passed into the UNICOS kernel through the parameter file. The parameter file index, offset, and length must match the node information. A node for root is needed to run the fsck(8) command; a node for swap is required for defining flaws; and a node for dump is required for snatching dumps.

The following example shows the mknod command needed to create the diagnostic physical device node for a DD-60 disk drive:

```
/etc/mknod /dev/ddd/2230.0 c dev_pdd 252 10 02230 0 120106 0137 0 0
```

Component	Definition
/dev/ddd/2230.0	Device path name
c	Character special
dev_pdd	Major device number

252	Minor device number
10	Disk type
02230	I/O path in octal
0	Starting sector number
120106	Length of slice in sectors
0137	Special purpose flags
0	Alternate I/O path for redundancy
0	Disk unit number

2.4.2 Examples of Physical Device Creation

Note: For information on creating physical devices on GigaRing based systems, see Section 2.4.3.

The following sections give detailed examples of creating the following types of physical devices:

- Disk
- RAM

Note: When working with I/O paths and flags, remember that octal numbers must have a leading 0. Because I/O paths are usually expressed in octal, take care when working with a multicluster system to express I/O paths in the proper form to a command. For example, I/O path `0130` works as expected because of the leading 0, but I/O path `1232` needs a leading 0 when expressed to a UNICOS command.

2.4.2.1 Creating a Physical Disk Device

To create a physical disk device, use the `mknod`(8) command. For information about the physical disk device interface, including the supported disk types, see the `pdd`(4) man page.

Slices normally begin and end on cylinder boundaries.

The I/O path is a concatenation of the cluster, EIOP, and channel numbers. The major number is `dev_pdd`, and the minor number should be less than `PDDSLMAX`, as defined in the parameter file.

The available flags are defined in the `/usr/include/sys/eslice.h` file as follows:

```
/*
 *  flags for physical slice control
 */
#define S_CONTROL       0001    /* control device */
#define S_NOBBF         0002    /* no bad block forwarding */
#define S_NOERREC       0004    /* no error recovery */
#define S_NOLOG         0010    /* no error logging */
#define S_NOWRITEB      0020    /* no write behind */
#define S_CWE           0040    /* control device write enable */
#define S_NOSPIRAL      0100    /* no spiraling */
#define S_NOSORT        0200    /* no disk sorting */
```

Example 1: The following commands define two slices on a DD-XX on cluster 1, EIOP 2, channel 32. The slice `ddXX_0` starts at cylinder 0 and spans 732 (01334) cylinders; its flags field is 0, there is no alternate I/O path, and its unit field is 0. `ddXX_1` starts at cylinder 732 (01334) and spans 150 (0226) cylinders; its flags field is 0, there is no alternate I/O path, and its unit field is 0.

```
/etc/mknod /dev/pdd/ddXX_0 c dev_pdd 50 3 01232      0 245952 0 0 0
/etc/mknod /dev/pdd/ddXX_1 c dev_pdd 51 3 01232 245952  50400 0 0 0
```

Example 2: The following commands define two slices spanning an entire DD-ZZ on cluster 0, EIOP 1, channel 30 (the DD-ZZs are daisy-chained and are on units 0 and 1):

```
/etc/mknod /dev/pdd/ddZZ_0 c dev_pdd 200 10 0130 0 119692 0 0 0
/etc/mknod /dev/pdd/ddZZ_1 c dev_pdd 201 10 0130 0 119692 0 0 1
```

Use the stor(8) command to examine your partitions. The following example shows the stor output that you would receive if you typed in the preceding mknod(8) commands:

```
DDZZ 0130.0

ID                      START               END         LENGTH
----------------------  ------------------  ---------  ----------------
name             minor    block     cyl.hd     cyl.hd    blocks  mbytes

ddZZ_0             200        0       0.00   05052.00    119692   1961.0


DDZZ 0130.1

ID                      START               END         LENGTH
----------------------  ------------------  ---------  ----------------
name             minor    block     cyl.hd     cyl.hd    blocks  mbytes

ddZZ_1             201        0       0.00   05052.00    119692   1961.0


DDXX 1232

ID                      START               END         LENGTH
----------------------  ------------------  ---------  ----------------
name             minor    block     cyl.hd     cyl.hd    blocks  mbytes

ddXX_0              50        0       0.00   01334.00    245952  1007.4
ddXX_1              51   245952   01334.00   01562.00     50400   206.4
```

2.4.2.2 Creating RAM Disks

The amount of core memory available for creating RAM disks is specified in the parameter file, as shown in the following example:

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

To create a RAM disk physical device, use the mknod(8) command. The device type and I/O path parameters are not applicable and should be 0. The major number is dev_rdd, and the minor number should be less than RDDSLMAX,

as defined in the parameter file. The following example shows the creation of a RAM physical device:

```
/etc/mknod /dev/pdd/ram0 c dev_rdd 0 0 0    0 1024
/etc/mknod /dev/pdd/ram1 c dev_rdd 1 0 0 1024 9216
```

2.4.2.3 Creating SSD Slices

The amount of SSD memory available for creating SSD slices is specified in the parameter file, as shown in the following example. (SSD devices are supported only on Cray T90 systems.)

```
SSD ssddev { length 10240 blocks;
        pdd ssd        {minor  3; block 0      ; length 10240 blocks;}
}
```

To create an SSD physical device, use the mknod(8) command. The device type and I/O path parameters are not applicable and should be 0. The major number is dev_ssdd, and the minor number should be less than SSDDSLMAX, as defined in the parameter file. The following example shows the creation of an SSD physical device:

```
/etc/mknod /dev/pdd/ssd0 c dev_ssdd 0 0 0    0 1024
/etc/mknod /dev/pdd/ssd1 c dev_ssdd 1 0 0 1024 9216
```

For information on using the SSD as a logical device, see Section 2.11.3, page 55.

2.4.3 Creating Physical Devices (GigaRing Systems)

You define a disk slice on a GigaRing based system just as you define a disk slice on an IOS-E based system, with the following considerations:

- The major device number for a disk device connected to the IPN-1 should be expressed symbolically as dev_qdd. See the qdd(4) man page.
- The major device number for a disk device connected to the MPN-1 or the FCN-1 should be expressed symbolically as dev_xdd. See the xdd(4) man page.
- The I/O path for a physical device on a GigaRing based system is a concatenation of the GigaRing number, the node number, and the controller number of the device.
- By convention, xdd device files are kept in the /dev/xdd directory. qdd device files, on the other hand, are kept in the /dev/pdd directory.

The following example shows the mknod(8) command needed to create a physical device node on a DD-318 disk drive connected to the MPN-1:

```
/etc/mknod /dev/xdd/s100a c dev_xdd 2 0 03021  0     100000 0 0 1 0
```

Component	Definition
/dev/xdd/s1000a	Device path name
c	Character special
dev_xdd	Major device number
2	Minor device number
0	Disk type
03021	I/O path in octal
0	Starting sector number
100000	Length of slice in sectors
0	Special purpose flags
0	Alternate I/O path for redundancy
1	Disk unit number
0	Disk subunit number

2.4.4 Creating Logical Devices

Logical devices are categorized into the following types:

- Simple logical
- Striped logical
- Mirrored logical

Each type has its own major number and associated device driver.

Logical devices are created by using the mknod(8) command, which has the following format:

```
/etc/mknod name b major minor 0 0 path
  /etc/mknod name c major minor 0 0 path
```

name Name of the logical device.

b Block special device.

`c`	Character special device.
major	Major device number.
minor	Minor device number in the range `1` through `LDDMAX`. You cannot use `minor 0` for a logical device.
`0 0`	Placeholders for future use.
path	Absolute path name to a physical device, logical device, or a logical descriptor file.

2.4.4.1 Creating Simple Logical Devices

Simple logical devices can point to a physical device or a logical descriptor file and use major number `dev_ldd`. The minor number must be less than `LDDSLMAX`, as defined in the parameter file.

The following example command sequence creates a simple logical direct device and its associated physical slice. The logical direct device has a minor device number of 100, it is on a DD-XX, on IOC 0, IOP 1, channel 30, starting at 0 with a length of 32,768 sectors.

```
/etc/mknod /dev/pdd/x0 c dev_pdd 100 3 0130 0 32768 0 0 0 0
/etc/mknod /dev/dsk/x0 b dev_ldd 100 0 0 /dev/pdd/x0
```

To examine logical device nodes, use the `ddstat`(8) command. The following is an example of `ddstat` output for the preceding `mknod`(8) commands:

```
# /etc/ddstat /dev/dsk/x0
x0 b 34/100 /dev/pdd/x0
        /dev/pdd/x0 c 32/100 3 0130 0 32768 00 0 0 0
```

The following example command sequence creates a simple logical indirect device and its associated physical slices:

```
/etc/mknod /dev/pdd/x0 c dev_pdd 100 3 0130 0 32768 0 0 0 0
/etc/mknod /dev/pdd/y0 c dev_pdd 110 3 0132 0 32768 0 0 0 0
/etc/mknod /dev/ldd/cluster0 L /dev/pdd/x0 /dev/pdd/y0
/etc/mknod /dev/dsk/cluster0 b dev_ldd 30 0 0 /dev/ldd/cluster0
```

The following is an example of the associated `ddstat` output:

```
# /etc/ddstat /dev/dsk/cluster0
cluster0 b 34/30 /dev/ldd/cluster0
        /dev/pdd/x0 c 32/100 3 0130 0 32768 00 0 0 0
        /dev/pdd/y0 c 32/110 3 0132 0 32768 00 0 0 0
```

2.4.4.2 Restrictions on Simple Logical Devices

Simple logical devices have an ldd disk driver restriction. The ldd disk driver requires that the length of each partition within an ldd be a multiple of the largest underlying device sector size. For example, if an ldd has two slices, one from a DD-302 (sector size=1 block) and one from a DA-302 (sector size=4 blocks), the length of the DD-302 slice must be a multiple of 4 blocks.

If the length of any partition in an ldd is not a multiple of the largest underlying device sector size, then the following message from the ldd driver is displayed on the UNICOS console when you attempt to open the device using, for example, mkfs:

```
uts/c1/io/ldd.c-02: WARNING i/o unit incompatible with length on open: devx
```

2.4.4.3 Creating Striped Logical Devices

Striped logical devices must point to a logical descriptor file. Striped logical devices use major number dev_sdd, as defined in the parameter file.

The following example command sequence creates a striped logical device, its associated physical slices, logical descriptor file, and direct logical device:

```
/etc/mknod /dev/pdd/x0 c dev_pdd 100 3 0130 0 32768
/etc/mknod /dev/pdd/y0 c dev_pdd 110 3 0132 0 32768
/etc/mknod /dev/ldd/stripe0 L /dev/pdd/x0 /dev/pdd/y0
/etc/mknod /dev/sdd/stripe0 b dev_sdd 30 0 0 /dev/ldd/stripe0
/etc/mknod /dev/dsk/stripe0 b dev_ldd 30 0 0 /dev/sdd/stripe0
```

The associated ddstat output is as follows:

```
# /etc/ddstat /dev/dsk/stripe0
stripe0 b 34/30 /dev/sdd/stripe0
       /dev/sdd/stripe0 b 39/30 /dev/ldd/stripe0
              /dev/pdd/x0 c 32/100 3 0130 0 32768 00 0 0 0
              /dev/pdd/y0 c 32/110 3 0132 0 32768 00 0 0 0
```

The following procedure provides an example of how to define devices that include SSD devices.

1. Define physical device slices of equal size where each device is on a unique I/O path (argument 6), each is of the same length (argument 8) and each has the same stripe I/O unit (definable on SSD only).

```
/etc/mknod /dev/pdd/ssd0_s1 c dev_ssdd 10 3 001 0 1000000 0 0 0 1024
/etc/mknod /dev/pdd/ssd1_s1 c dev_ssdd 20 3 002 0 1000000 0 0 0 1024
```

```
/etc/mknod /dev/pdd/ssd2_s1 c dev_ssdd 30 3 004 0 1000000 0 0 0 1024
/etc/mknod /dev/pdd/ssd3_s1 c dev_ssdd 40 3 010 0 1000000 0 0 0 1024
```

2. Define a logical node to reference the four stripe components.

```
/etc/mknod /dev/ldd/s1 L /dev/pdd/ssd0_s1
                         /dev/pdd/ssd1_s1
                         /dev/pdd/ssd2_s1
                         /dev/pdd/ssd3_s1
```

3. Define a striped device.

```
/etc/mknod /dev/sdd/s1 c dev_sdd 10 0 0 /dev/ldd/s1
```

4. Define a logical disk device to reference the striped device.

```
/etc/mknod /dev/dsk/s1 b dev_ldd 10 0 0 /dev/sdd/s1
```

You can use the `ddstat`(8) or the `stor`(8) command to verify the device structure as defined. This verification is done by reading the appropriate inodes and files. The device is not opened when `ddstat` or `stor` performs a read operation. Executing `/etc/ddstat` on the configuration example would show the following:

```
/dev/dsk/s1 b 34/10 0 0 /dev/sdd/s1
      /dev/sdd/s1 c 39/10 0 0 /dev/lss/s1
          /dev/pdd/ssd0_s1 c 37/10 3 001 0 1000000 0 0 0 1024
          /dev/pdd/ssd1_s1 c 37/20 3 002 0 1000000 0 0 0 1024
          /dev/pdd/ssd2_s1 c 37/30 3 004 0 1000000 0 0 0 1024
          /dev/pdd/ssd3_s1 c 37/40 3 010 0 1000000 0 0 0 1024
```

Errors in configuration are generally detected by the `ddstat` or `stor` command or by the device drivers when the logical disk device is first opened. Errors at open time are issued to the console and the kernel log.

2.4.4.4 Creating Mirrored Logical Devices

Mirrored logical devices must point to a logical descriptor file and use major number `dev_mdd`. The minor number must be less than `LDDSLMAX`, as defined in the parameter file. The following command sequence creates a mirrored

logical device, its associated physical slices, logical descriptor file, and direct logical device:

```
/etc/mknod /dev/pdd/x0 c dev_pdd 100 3 0130 0 32768
/etc/mknod /dev/pdd/y0 c dev_pdd 110 3 0132 0 32768
/etc/mknod /dev/ldd/mirror0 L /dev/pdd/x0 /dev/pdd/y0
/etc/mknod /dev/mdd/mirror0 c dev_mdd 30 0 077 /dev/ldd/mirror0
/etc/mknod /dev/dsk/mirror0 b dev_ldd 30 0 0 /dev/mdd/mirror0
```

The associated `ddstat` output is as follows:

```
# /etc/ddstat /dev/dsk/mirror0
mirror0 b 34/30 /dev/mdd/mirror0
       /dev/mdd/mirror0 b 40/30 0 077 /dev/ldd/mirror0
              /dev/pdd/x0 c 32/100 3 0130 0 32768 00 0 0 0
              /dev/pdd/y0 c 32/110 3 0132 0 32768 00 0 0 0
```

Each member of a mirrored group (in the previous example `pdd/x0` and `pdd/y0`) contains a *rwmode* parameter. The bits of this parameter control read, write, and initialize privilege for each member; each octet represents (from right to left) r-w-x (read / write / x(init)). For a *rwmode* of `077`, as shown in the preceding example, both `x0` and `y0` are read / write / x(init). A *rwmode* of `073` would indicate initialize `x0` and `y0`, read only from `y0`, and write to both `x0` and `y0`. A *rwmode* of `037` would indicate that `x0` is read-enabled and `y0` is not.

See the `mdd`(4) man page for more information on `mdd` files. For a more complete overview of mirrored file systems, see Section 2.10, page 46.

2.4.4.5 Restrictions on Striped and Mirrored Logical Devices

The following restrictions and guidelines apply to the configuration of striped and mirrored groups:

- All slices must be of the same length.
- All devices on a striped group must be up. When configuring a mirror group, only one member needs to be up.
- The starting sector of each slice must be a multiple of the stripe factor.
- The length of each slice must be a multiple of the stripe factor.
- When a stripe group is opened, the driver allocates a number of PBUF headers equal to nine times the number of slices. For mirror groups, the number is three times the number of slices. If there are not enough PBUF headers free,

the driver waits. If not enough PBUF headers are configured, the system may hang.

2.4.5 Creating Logical Descriptor Files

A logical descriptor file can contain up to 64 absolute path names, each which can be up to 48 characters in length. Each absolute path name is a *member* of the logical disk device. The members are combined in a manner prescribed by the character or block special device referencing the logical descriptor file. The members can be physical or logical devices.

You can use the mknod command to create a logical descriptor file, using the following format:

```
/etc/mknod name L member0 [member1 member2] ...
```

Example 1: To create a logical descriptor file x containing two physical slices of type x0 and x2, specify the following:

```
/etc/mknod /dev/ldd/x L /dev/pdd/x0 /dev/pdd/x1
```

Example 2: To create a logical descriptor file y containing two logical striped devices of type y0 and y2, specify the following:

```
/etc/mknod /dev/ldd/y L /dev/sdd/y0 /dev/sdd/y1
```

2.4.6 Defining Alternate Disk Paths

The UNICOS operating system allows you to define an alternate path to a disk when you define the device node by using the mknod(8) command. The alternate path provides a backup for the primary path to a device. If you configure a device with an alternate path, it is initialized when you open the device. The system then can attempt error recovery by means of the alternate path when it detects a hard failure on the active disk path.

A *path* is defined as the hardware resources between the CPU and the disk device. On IOS-E based systems, these consist of the low-speed channel to the mainframe, the multiplexing I/O processor (MUXIOP), the low-speed channel from the MUXIOP to a given model E I/O processor (EIOP), and the EIOP itself with its channel adapter.

You can access a disk through the alternate path if any element of the primary path is not functional and hardware elements exist to connect to a second port of

the disk device. Optimally, on an IOS-E based system, the alternate path should include a different I/O cluster and EIOP.

You create an alternate path when you define a physical device interface with the `mknod`(8) command. The following example defines an alternate I/O path of 1130 to device 0236.7.

```
mknod /dev/pdd/device.7 c dev_pdd 15 10 0236 1472 32768 0 01130 7 0
```

A path of 1130 defines IOC 1, IOP 1, and channel 30.

The alternate path is used as part of error recovery. If the standard five retries and micro-sequencing error recovery fails, the disk driver attempts recovery on the alternate path. This can recover errors such as high-speed channel (HISP) errors and errors on the channel adaptors or in the EIOP. Errors that are on the disk itself remain as errors, just as on the primary path.

A device running on the alternate path does not switch back to the primary path if normal error recovery fails. Errors that occur during write-behind, when the CPU no longer has the data, cannot be recovered by switching to a different path, because data cannot be retrieved from the IOS.

If a device is configured with an alternate path, the system, by default, switches to the alternate path during error recovery. You can disable and enable the error recovery switch with the `autoswitch` option of the `sdconf`(8) command, as follows.

`sdconf` *device* `autoswitch on | off`

You can control both the primary and alternate path by using the `sdconf` command. Executing `sdconf` *device pripath* and `sdconf` *device altpath* resets the primary or alternate path.

You can also change the path to a device with the `inform`(8) command on the OWS-E. If the `pddinform` function in the driver is notified that a given EIOP has died, any device running on that EIOP that has an alternate path switches to the alternate path to complete any outstanding I/O.

2.4.7 Configuring Alternate Paths on FCN Devices

FCN devices can be dual-ported. If these devices are dual-ported, the `XDD` driver has the capability to dynamically switch to the alternate path when the primary path fails. If the alternate path fails, the `XDD` driver will not switch back to the primary path.

An FCN device can be dual-ported in any of three configurations:

- 2 Fibre Channel Loops on the same FCN device
- 2 FCN devices on the same GigaRing channel
- 2 FCN devices, each on a different GigaRing channel

These configurations are illustrated in "Configuring Alternate Paths on FCN Devices," Section 2.4.7.1.

2.4.7.1 Configuring Alternate Paths on FCN Devices

The following diagram illustrates a system in which no devices are dual-ported and no alternate paths are defined.

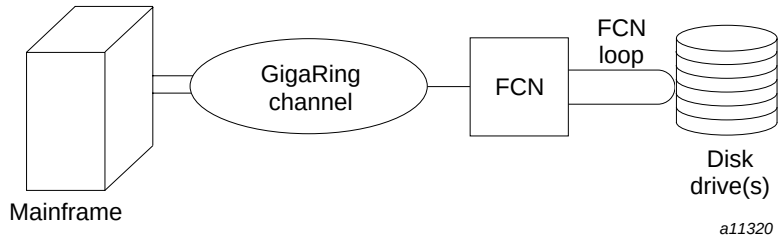


Figure 1. Configuration with No Alternate Path

The following diagram illustrates a system configured with two Fibre Channel Loops on a single FCN device.

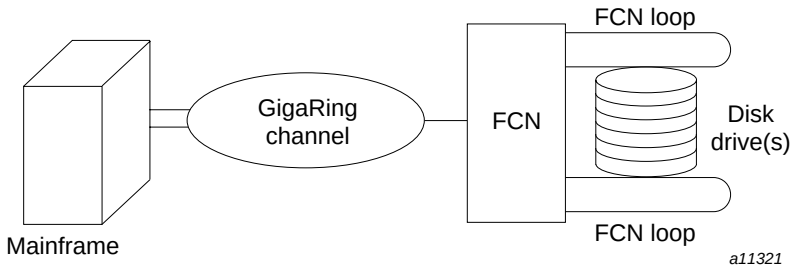


Figure 2. Configuration 1: One FCN Device, Two Fibre Channel Loops

Configuration 1 shows the following path configuration:

primary path	Ring 1 Node 1 Channel 1 Unit 1
alternate path	Ring 1 Node 1 Channel 2 Unit 1

Note: When configuring the primary and alternate paths, avoid configuring them on the same ring. If the ring has a node failure, the failure will bring the ring down.

The following command creates a device node with the primary and alternate path definitions of Configuration 1:

```
/etc/mknod dd308 c 33 40 0 01011 0 2340000 0 01012 01 0
```

The following diagram illustrates a system configured with two FCN devices on the same GigaRing channel.

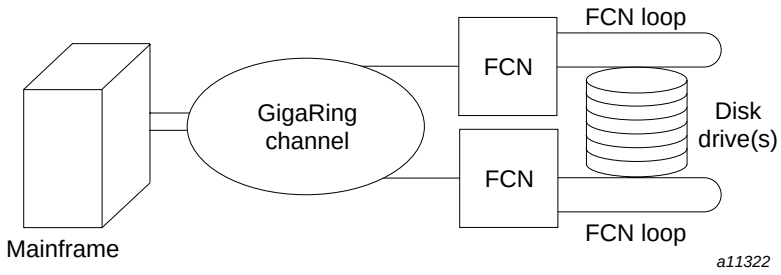


Figure 3. Configuration 2: Two FCN Devices, One GigaRing Channel

Configuration 2 shows the following path configuration:

primary path	Ring 1 Node 1 Channel 1 Unit 1
alternate path	Ring 1 Node 2 Channel 1 Unit 1

The following command creates a device node with the primary and alternate path definitions of Configuration 2:

```
/etc/mknod dd308 c 33 40 0 01011 0 2340000 0 01021 01 0
```

The following command creates a device node with the primary and alternate path definitions of Configuration 1:

```
/etc/mknod dd308 c 33 40 0 01011 0 2340000 0 01012 01 0
```

The following diagram illustrates a system configured with two FCN devices on different GigaRing channels.

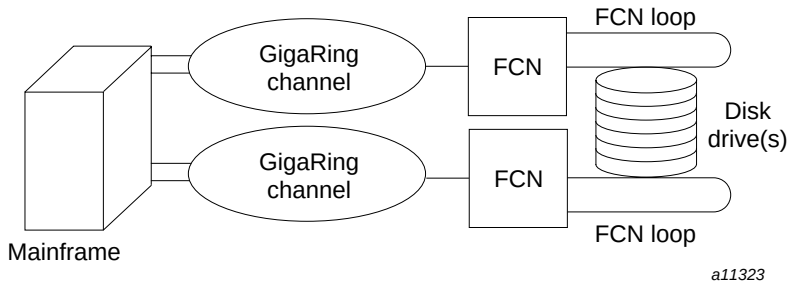


Figure 4. Configuration 3: Two FCN Devices, Two GigaRing Channels

Configuration 3 shows the following path configuration:

primary path	Ring 1 Node 1 Channel 1 Unit 1
alternate path	Ring 2 Node 2 Channel 1 Unit 1

The following command creates a device node with the primary and alternate path definitions of Configuration 3:

```
/etc/mknod dd308 c 33 40 0 01011 0 2340000 0 02021 01 0
```

2.4.7.2 Failure Modes

Dynamic switching to the alternate path takes place under the following conditions:

- Fibre Channel Loop failure
- FCN failure
- GigaRing channel failure

The following sections describe the system response to each of these conditions.

2.4.7.2.1 Fibre Channel Loop Failure

When the mainframe detects a failure of a request to the FCN that is associated with the Fibre Channel Loop, and if there is an alternate path to the device through another Channel Loop on the same FCN, another FCN on the same GigaRing channel, or another GigaRing channel, the dynamic path switch algorithm will be executed.

The FCN return a status indicating that this is a failure of the Fibre Channel Loop.

You can force this type of failure by disconnecting the cable to the port or disconnecting the Fibre Channel Loop cable.

2.4.7.2.2 FCN Failure

Failure of the FCN is detected by the PEER to PEER message protocol layer. This condition is detected after failure of any communication from the FCN with the mainframe after 30 seconds. At this time, the XDD driver is informed of a PEER down message condition.

If there is an alternate path for these devices through another FCN on the same GigaRing channel or another GigaRing channel, the XDD driver will execute the alternate path switching algorithm.

You can force this type of failure by disconnecting the FCN from the GigaRing channel.

2.4.7.2.3 GigaRing Channel Failure

Failure of the GigaRing channel is detected by the PEER to PEER message protocol layer. This condition is detected after failure of any communication from the FCN with the mainframe after 30 seconds. At this time, the XDD driver is informed of a PEER down message condition. The system cannot return a status that distinguishes between a GigaRing channel failure and an FCN failure.

If there is an alternate path for these devices either through another FCN on the same GigaRing channel or another GigaRing channel, the XDD driver will execute the alternate path switching algorithm.

If the alternate path is on another FCN on the same GigaRing channel, the path switch will fail. If the alternate path is on a different GigaRing channel, the path switching will be successful if the FCN is functional.

You can force this type of failure by disconnecting the GigaRing cable.

2.4.7.2.4 Alternate Path Switching Restrictions

The following restrictions apply to the use of the alternate path:

- On first open, both paths must be valid and functional; that is, the XDD driver must be able to open both paths on first open.
- Once the device is being used on the alternate path, failure of the alternate path will not cause the XDD driver to switch to the primary path.

2.4.8 Shared Dump and Swap Configuration

Under the UNICOS operating system, you can configure the same physical slice that you use for a dump partition as part of the swap device. The dump header is preserved permanently and space used by a system dump will be allocated as needed for system dumps. The space is released back to the swap device when the dump is processed by cpdmp(8). A single slice swap device may be shared with the dump device.

The following restrictions apply when configuring a shared slice:

- The slice must be disk-based.
- SSD or RAM slices may not be used as the shared slice.
- The slice cannot be a member of a striped or mirrored device.
- The slice must be defined as a pdd type member of the swap device.

In the following swap configuration, slice dump may be chosen as the shared slice, but slices ssd_swap, diskswap1, and diskswap2 may not.

```
SSD  ssd      {
     length 1024 Mwords;
     pdd ssd_sds    {minor    4; block         0; length  1835008 blocks;}
     pdd ssd_swap   {minor    6; block   1835008; length  262144  blocks;}

disk d0130.1  {type DD60; iopath{cluster 0; eiop 1; channel 030;}
     unit 1;
     pdd dump      {minor   22; sector       0; length  119692 sectors;}

disk d0236.0  {type DD60; iopath{cluster 0; eiop 2; channel 036;}
     unit 0;
     pdd diskswap1  {minor   67; sector       0; length  119692 sectors;}
}
disk d1236.0  {type DD60; iopath{cluster 1; eiop 2; channel 036;}
     unit 0;
     pdd diskswap2  {minor   68; sector       0; length  119692 sectors;}


   sdd stripe_swap { minor 10; pdd diskswap1;
                               pdd diskswap2;

   ldd swap      { minor  10; pdd ssd_swap;
                              pdd dump;
                              sdd stripe_swap;
```

```
   swapdev is ldd swap;
```

Both a swap logical device and a dump logical device must be defined with the shared slice as a component. The minor number of the dump logical device is different than the minor number of the swap logical device, but they are not required to be the same as the examples shown. For the above example the dump device definition would be as follows:

```
ldd dump      { minor  19; pdd dump    ;
              }
```

The dumpdev statement is provided below. To initialize and use the shared partition, 3 different boot parameter files must be used.

For dump device initialization, the UNICOS system must be booted one time in a nonshared configuration. The mkdmp command should be run as shown in the mkdmp(8) man page. To initialize the dump device in the above example, use the following parameter file definitions:

```
SSD  ssd      {
     length 1024 Mwords;
     pdd ssd_sds    {minor    4; block         0; length  1835008 blocks;}
     pdd ssd_swap   {minor    6; block   1835008; length  262144  blocks;}

disk d0130.1  {type DD60; iopath{cluster 0; eiop 1; channel 030;}
     unit 1;
     pdd dump      {minor   22; sector       0; length  119692 sectors;}

disk d0236.0  {type DD60; iopath{cluster 0; eiop 2; channel 036;}
     unit 0;
     pdd diskswap1  {minor   67; sector       0; length  119692 sectors;}
}
disk d1236.0  {type DD60; iopath{cluster 1; eiop 2; channel 036;}
     unit 0;
     pdd diskswap2  {minor   68; sector       0; length  119692 sectors;}


   sdd stripe_swap { minor 10; pdd diskswap1;
                               pdd diskswap2;

   ldd swap      { minor  10; pdd ssd_swap;
                              sdd stripe_swap;

   ldd dump      { minor  19; pdd dump    ;
                 }
```

```
swapdev is ldd swap;
dmpdev  is ldd dump;
```

After the dump device is initialized, the shared configuration should be used. Reinitialization of the dump device is only necessary if the dump device is changed or if flaws are added or removed on the dump device. If reinitialization is necessary, the nonshared configuration must booted for just the mkdmp(8) processing.

To boot the UNICOS system in a shared dump/swap configuration, the dmpdev statement must indicate that the dump device is a pdd device:

```
dmpdev is pdd dump;
```

To perform a system dump, the dumpsys(8) command requires that the dmpdev statement indicate that the dump device is an ldd device, so the boot parameter file cannot be used:

```
dmpdev is ldd dump;
```

It is suggested that the shared slice be named dump. This allows the cpdmp(8) command to use defaults and will be easier to implement on-site.

2.5 Configuring Disk Arrays

This section contains information on how to configure DD-331 into a DA-331 array.

2.5.1 Installing an Array

Perform the following steps to install and configure a disk array:

- Copy any data to be saved to another media
- Initialize the device as a RAID-3 array
- Write data to the parity drive
- Modify configuration information

After you perform these steps, the array is ready for I/O. These steps are described in detail below.

1. Copy any data to be saved to another media.

   It is not possible to move a file system on 4 single drives onto an array of 4+1 drives without first dumping, then restoring the data.

   There are two factors that may affect the restore of the data:

   - If there are many small files, the restored data could take more space. This is because the minimum sector size is four blocks, so small files may take more space.
   - For extended files, the space allocation may be more efficient after the restore and take less space.

   Unless the existing file system is nearly full, it is unlikely that it will not restore.

2. Initialize the device as a RAID-3 array:

   ```
   xdms -a init -m 35 02010.011 # 02010.011 in octal
   ```

   See the `xdms`(8) man page for details.

   The syntax `-m 35` means raid 3, 5 units and does not indicate a mask, but a hex value that identifies the raid type (mode) to initialize.

3. Write data to parity drive.

   ```
   xdms -a scrub 02010.011
   ```

   A scrub of a DA-331 takes about one hour and 50 minutes.

   If the scrub fails or is interrupted with Control-C, you will see errors when you execute `mkfs`(8) on the file system

4. Modify configuration information:

   `/etc/mknod /dev/xdd/`*`name`* `c 33` *`minor`* `4` *`pripath`* `0 2340000 0` *`altpath unit`*

   The *dtype* field needs to be 4 (to indicate a RAID-3 device); the 4 specifies that there are four 512-word blocks per sector. *Loop_ID* is sometimes used instead of *unit* number. Only one `/dev/xdd/`*`XXXX`* node is needed to represent an entire array.

   The following example shows the output from a `ddstat -m disk*` command on an array:

```
            Name  Type  Maj.  Min. Sector  Path  Start  Length  Flags  Alt Path  Unit  I Field
/etc/mknod disk1  c    33    39      4    02010   0    2340000   0       0       001     0
/etc/mknod disk2  c    33    40      4    02010   0    2340000   0       0       011     0
```

```
/etc/mknod disk3  c    33    41      4    02020   0    2340000   0       0       001     0
/etc/mknod disk4  c    33    42      4    02030   0    2340000   0       0       001     0
/etc/mknod disk5  c    33    43      4    02030   0    2340000   0       0       011     0
```

The last line in the above example represents a RAID-3 slice on ring 2, node 3, FCN channel / loop 0, involving Loop-IDs / units 011-015 (see the values under Path and Unit for this information).

Note: The following describes some of the column headings: Maj. is major number, Min. is minor number, and Alt Path is alternative path. IField applies only to ND-4x HIPPI disks.

Configure the `/dev/dsk` entries:

```
/etc/mknod disk1 b 34 39 0 0 /dev/xdd/disk1
/etc/mknod disk2 b 34 40 0 0 /dev/xdd/disk2
/etc/mknod disk3 b 34 41 0 0 /dev/xdd/disk3
/etc/mknod disk4 b 34 42 0 0 /dev/xdd/disk4
/etc/mknod disk5 b 34 43 0 0 /dev/xdd/disk5
```

The following example shows the output from a `ddstat -m *` command:

```
/etc/mknod disk1 b 34 39 0 0 /dev/xdd/disk1
        /etc/mknod /dev/xdd/disk1 c 33 39 4 02010 0 2340000 0 0 01 0
/etc/mknod disk2 b 34 40 0 0 /dev/xdd/disk2
        /etc/mknod /dev/xdd/disk2 c 33 40 4 02010 0 2340000 0 0 011 0
/etc/mknod disk3 b 34 41 0 0 /dev/xdd/disk3
        /etc/mknod /dev/xdd/disk3 c 33 41 4 02020 0 2340000 0 0 01 0
/etc/mknod disk4 b 34 42 0 0 /dev/xdd/disk4
        /etc/mknod /dev/xdd/disk4 c 33 42 4 02030 0 2340000 0 0 01 0
/etc/mknod disk5 b 34 43 0 0 /dev/xdd/disk5
        /etc/mknod /dev/xdd/disk5 c 33 43 4 02030 0 2340000 0 0 011 0
```

After performing the above steps, the array is ready for I/O:

```
# /etc/mkfs -q -A96 /dev/dsk/disk1

/etc/mkfs: *** NC1FS file system initialized on /dev/dsk/disk1 ***
        *** Lower security level = 0   Upper security level = 0
        *** Valid security compartments = 0
                none
        *** Big file: 32768 bytes  big allocation unit: 96 blocks
        *** Allocation strategy: Round robin all files(rrf)
        *** 1 partitions / 9360000 total blocks / 9351704 free
        *** 131072 total inodes / 131068 free
        *** 1 primary partitions / 4 blocks per alloc. unit
        *** File system partitions:
                part 0: primary blocks 0 - 9359999  on device disk1
        *** Panic on error option selected
```

2.5.2 Replacing a Failing Spindle

Use the following procedure to replace a failing spindle when unrecovered errors are occurring.

1. Determine the failing spindle from the failing spindle mask provided in the `errpt` output or system console log. See "Spindle to unit number mapping", Table 1.

Table 1. Spindle to Unit Number Mapping

Unit	Spindle	Spindle mask
0?1	4	020 (parity)
0?2	3	010
0?3	2	004
0?4	1	002
0?5	0	001

2. Disable the spindle:

   `xdms -a disable` *iopath.unit-spindle*

For example:

```
xdms -a disable 02010.01-3 # disable spindle 3
```

This step, disabling the spindle, may take place automatically, depending on the type of errors encountered.

3. Spin down the spindle:

   `xdms -a spindown` *iopath.unit-spindle*

   If you try to spin down a drive that is not disabled, the next I/O to that array/spindle will cause the array to spin up as part of normal error recovery.

4. Replace the failing spindle.

   Note: Pulling or insertion of a spindle will cause the FCN software to re-initialize the DSF. This DSF initialization may take a few minutes and will affect not just the DSF containing the pulled/pushed spindle (which will suspend I/O to other drives in the DSF), but will affect other DSFs on the same daisy-chained channel. Some error/console message may appear at this time.

   Before reconstructing the array, check that the serial number of the drive/unit (on the I/O path) is correct:

   `xdms -a info` *iopath.unit-spindle*

5. Reconstruct the spindle.

   The replacement drive that is to be used to reconstruct the full 4+1 array does not need to be initialized.

   Execute the following command:

   `xdms -a reconstruct` *iopath.unit*

   A reconstruct can take 1.5 hours, minimum, depending on other I/O to the same disk array or other I/O to the same DSF. Some reconstructs have taken over 13 hours to complete, when running heavy I/O to the same array (for example, by using `fstest`(8)).

   A message appears on the console (and in the `ion_syslog.info`) when the reconstruct is complete:

   ```
   10/16/97 17:39:17 NOTICE sdisk_admin_r3.c line 774
   Array Reconstruction is complete on FC Loop 0 Target 1
   ```

2.5.3 Converting RAID Members to Single Spindles

The following example initializes all RAID members to a single spindle:

```
# ./xdms -ainit -m 1 4044.21

4044.21 appears to be a member of a RAID-3S 4+1    .
To init JUST THIS MEMBER to Single Spindle enter "y"
To init ALL RAID MEMBERS to Single Spindle enter "a"

Please be sure all activity to device is idled before continuing with init.
Continue with Init to Single Spindle now? (y, a, or n)
a
xdms: Initialized iopath 4044 unit 21
xdms: Initialized iopath 4044 unit 22
xdms: Initialized iopath 4044 unit 23
xdms: Initialized iopath 4044 unit 24
xdms: Initialized iopath 4044 unit 25
```

Note: If the `-z` option is used with the `-a init` action and the device mode is a `-m 1` only the RAID member specified will be initialized. The net affect of the `-z` option is the same as in previous example.

2.5.4 Software Limitations

The following limitations are in effect when configuring and restoring disk arrays.

- Executing a Control-C after issuing a `xdms -a reconstruct` command does not halt the reconstruct. After the initial reconstruct request, the FCN performs the reconstruct. However, `xdms`(8) can resume monitoring status of the reconstruct by reissuing the reconstruct or issuing an info request on the array device. The info request will indicate the percent of reconstruct complete if there is a reconstruct currently in process.
- The addresses of array members cannot be changed. If the array was initialized as targets / units 1–5, it cannot be moved to 9–13 (011–015), 17–21 (021–025), etc. It can be moved to another channel as long as the target addresses are the same. When you want to change the addresses, you must execute `xdms -a init` and remake the `/dev/xdd` nodes.

2.6 File System Initialization

Use the mkfs(8) command to initialize file systems. The mkfs command builds the file system with a boot block, a super block, a root inode and a bit map of free blocks. By default, it also performs a surface check and zeroes the disk data blocks before initialization. When the UNICOS multilevel security (MLS) feature is enabled, mkfs provides the new file system with minimum and maximum security levels and authorized compartments. See Chapter 8, page 145, for more information on using mkfs on a UNICOS MLS system.

NC1FS file systems include a secondary allocation area. The secondary allocation area is a means of segmenting file data by usage. The secondary allocation area contains only user file data and may be allocated in different allocation units than primary allocation areas. If a secondary allocation area exists, default allocation of user data will occur there once a file has grown to "big file" size, as defined in the sys/param.h file or with the mkfs(8) or setfs(8) commands.

For a complete list of the options available with mkfs, see the mkfs(8) man page. For further information on allocation of inode regions, see Section 2.8, page 44.

2.7 Inode Allocation Strategies

You can specify inode allocation strategies at system configuration time by using the mkfs(8) command with the -a option, as follows:

```
mkfs -a strategy
```

You can also change the allocation style after configuration is complete by using the setfs(8) command.

The allocation strategies are as follows:

Strategy	Description
rrf	Round-robin files (default)
rrd1	Round-robin first-level directories
rrda	Round-robin all directories

Round-robin allocation is a process of allocating files and directories to partitions in sequence. When a file or directory is created, it is assigned to the next partition in sequence after the partition to which the previous file or directory was assigned.

When a file or directory is created in the last partition, the next partition in sequence is partition 0.

Each allocation strategy specifies the preferred location for data blocks, directory blocks, and inodes; however, if the preferred locations are full, the kernel places these items wherever possible in the file system.

If you use the logical device cache with your file systems, you can reduce system overhead by using the `mkfs` command to make the allocation unit equal to the logical device cache block size or a multiple of the same. Alternatively, you can use partition cache with some or all of the partitions of a file system to reduce system overhead.

You can improve performance on individual jobs and by pre-allocating space using the `setf`(1) command, the `assign`(1) command, or the `ialloc`(2) system call. These commands and system call perform all the allocation in one step and allocate storage in contiguous or nearly contiguous areas. Additionally, you can use `setf`, `assign`, and `ialloc` to force allocation from particular file system slices, so that different files can be placed on different physical disk devices.

2.7.1 `rrf` Allocation

The default allocation strategy, `rrf`, is the recommended strategy. The inodes and directories are assigned to the first 25% of partition 0 whenever possible. When new files are created, the data blocks are round-robined. As data blocks are added to a file, they remain on the same partition as the preceding data blocks whenever possible. This allocation style provides good recovery and a good distribution of file blocks for performance.

2.7.2 `rrd1` Allocation

The `rrd1` allocation strategy round-robins all first-level directories among the available partitions. (*First level directories* are directories defined in the root (top-level) directory of the file system.) The inode and the directory data blocks for each directory are in the assigned partition. Within a first-level directory, all files and subdirectories remain in the same partition as the first-level directory whenever possible.

This allocation scheme allows for better recovery and performance by limiting the impact of system crashes. For example, if a first-level directory corresponds to a single user, as in the typical home directory case, then all the files for one user are in the same partition and on the same disk. If one disk of a multidisk crashes, only a subset of all the users are affected. However, the top-level

directory data blocks and inode are on partition 0. If the first disk is lost, the entire file system must be restored as before.

The disadvantage to this allocation strategy is that there can be severe performance penalty for assigning all of the files for one user on the same disk. If, for example, you use the `cp`(1) command to copy a file within the directory to another file within that directory, you would be reading from and writing to files on the same disk drive.

2.7.3 `rrda` Allocation

The `rrda` allocation strategy round-robins each new directory that is created. The disk files within a directory have their inodes and data blocks in the same partition as the parent directory.

This strategy produces bad performance and recovery. If one disk of a three disk site were lost, even though the root directory is in partition 0, one third of its directory entries will have been on the bad disk. Of the remaining two thirds, the files within those directories will be recoverable, but one third of the subdirectories will be unrecoverable. The more complex the directory tree is, the worse this situation is. Using the `fsck`(8) command on this file system is not always productive.

The same performance problems of the `rrd1` allocation also affect the `rrda` strategy. If you copy one file to another in one directory, the command is always doing I/O on one drive only. Also, the data blocks for files within a directory are not distributed among the disks in the system.

2.8 Inode Region Allocation

Before creating a file system, you must understand how inode regions are allocated and how the `mkfs -i` option can be used. Up to 64 partitions can exist in an `NC1FS` file system. Each partition can have up to four inode regions. The size of an inode region is limited by the number of bits contained in the first block that is used as a bit map for free inodes. For disks with 512-word sectors, each inode region can contain at most 32,768 inodes. The size of an inode region is defined when the region is created and cannot later expand or contract.

When the `mkfs`(8) command creates a file system, exactly one inode region is created in each partition. When an inode region in a partition is full, a new one is created in that partition unless there are already four regions. Then, an inode region in the next partition is tried. If all four regions in all partitions are full, no more files can be created until some files (and inodes) are released.

The `mkfs -i` option specifies the desired ratio of blocks to inodes; the default value is 4.

Consider some examples in which the file systems have been created with the `mkfs -i 2` command and they contain 100,000 blocks. The `-i 2` option and argument indicate that 50,000 inodes (100,000 / 2) should be created. It is assumed that the default `rrf` allocation strategy is used in the following examples.

File system 1 contains one partition. Because one inode region is created and an inode region can contain at most 32,768 inodes, only 32,768 inodes are created for this file system. If all 32,768 are used, the file system will create another inode region if there is room on the device. A maximum of four inode regions can be created per partition. Therefore, at most, four inode regions can exist on file system 1.

File system 2 contains two partitions. The inode region on the first partition contains 32,768 inodes. The inode region on the second partition contains 17,232 inodes, so there are 50,000 inodes available in the file system. If the inode region in the first partition fills, a second, then a third, and finally a fourth are created before the inode region in the second partition is used. An exception to this rule would occur if there were not enough contiguous blocks to create additional regions in the first partition. In that case, the kernel would switch immediately to the inode region in the second partition.

File system 3 contains three partitions. The inode region on the first partition contains 32,768 inodes, the region on the second partition contains 17,232 inodes, and the region on the third partition contains 16 inodes. (One block contains 16 inodes; this is the smallest region that is created.) File system 3 contains 50,016 inodes. As with file system 2, the kernel will create more regions in the first partition (up to four total regions) before using the inode region in the second partition. The kernel attempts to create all four regions in the second partition before using the inode region in the third partition.

The size of the inode regions created in these example file systems when the first region is full do not depend on what had been specified with the `mkfs -i` option. Rather, the size depends on the ratio of the number of blocks actually in use to the number of inodes (used and unused) in all of the existing inode regions.

For example, the first inode region fills up in file system 2 and 80,000 blocks are used in the file system. Because there is already room for 50,000 inodes (in the regions in both partitions), the blocks-to-inode ratio is 80,000 / 50,000 = 1 (integer division). Therefore, a blocks-to-inode ratio of 1 is used for the next region. Because there are 20,000 free blocks in the file system, room is allocated for 20,000 / 1 = 20,000 inodes in the second inode region in the first partition.

2.9 Labeling a File System

A label on a newly created file system should be created through the `labelit`(8) command. It is optional, but when a label is not given to a file system, a warning message is issued in the following format when the file system is mounted; *mntpt* is the mount point of the file system:

`mount: warning: <> mounted as </`*mntpt*`>`

The basic format of the `labelit` command is as follows:

```
/etc/labelit device [fsname volname]
```

The *device* is the name of the device file that you want to label. The variables *fsname* and *volname* specify the file system name and volume name to be written in the label. The `labelit` command takes several options. See the `labelit`(8) man page for a description of the options to `labelit`.

The following example shows how to label the device `/dev/dsk/usr` as the file system `usr` with a volume name of `usr-6.1`. After the file system is labeled, a `sync`(1) command is issued.

```
/etc/labelit /dev/dsk/usr usr usr-6.1sync
```

2.10 Mirrored File Systems

A mirrored file system resides in two or more component devices, each of which contains a full copy of the mirrored information. A write operation to the mirrored device causes separate write operations to be performed on each of the components. A read operation may be performed on any of the component devices.

The multiple write operations provide for redundancy and for recovery in the case of a failure by a single component. The multiple read paths provide more options for scheduling the read operation, leading to faster completion

2.10.1 Creating a Mirrored File System

Example 1: The following example creates a mirrored file system consisting of identical areas on two different physical disks.

```
# for each component
/etc/mknod /dev/pdd/m0 c dev_pdd 100 10 0130 0 119692 0 0 0
```

```
/etc/mknod /dev/pdd/m1 c dev_pdd 101 10 0132 0 119692 0 0 0
# grouping the physical devices into a logical device
/etc/mknod /dev/ldd/log_mir L /dev/pdd/m0 /dev/pdd/m1
# making the mirror
/etc/mknod /dev/mdd/mir c dev_mdd 100 0 07777 /dev/ldd/log_mir
# making the fs
/etc/mknod /dev/dsk/fs_mir b dev_ldd 100 0 0 /dev/mdd/mir
/etc/mkfs  /dev/dsk/fs_mir
```

Example 2: The following example creates a mirrored file system consisting of a slice of the SSD paired with a slice of disk. This file system is intended to create an "all-cached-spindle" in which you perform read operations from the SSD and write operations to both the SSD and the non-volatile disk; this combines the speed of SSD with the permanence of a non-volatile disk. Because of the nature of the default path through system startup, it is important that the SSD component be declared second in the mirror.

```
# for each component
/etc/mknod /dev/pdd/acs_disk c dev_pdd 100 10 0130 0 4600 0 0 0
/etc/mknod /dev/pdd/acs_ssd c dev_ssdd 10 10 0 4600
# for the logical device
/etc/mknod /dev/ldd/acs_ldd L /dev/pdd/acs_disk /dev/pdd/acs_ssd
# for the mirror
/etc/mknod /dev/mdd/acs_mir c dev_mdd 100 0 037 /dev/ldd/acs_ldd
# for the file system
/etc/mknod /dev/dsk/acs b 100 0 0 /dev/mdd/acs_mir
/etc/mkfs /dev/dsk/acs
```

2.10.2 Configuring a Mirrored Device

While the mountable file systems are usually found in the `/dev/dsk` directory, the mirrored devices are usually in the `/dev/mdd` directory. At this level, before it is a file system, the mirrored device may be configured to enable reading and/or writing on selected components of the device.

For an open mirrored device that is known to the `mdd` driver in the kernel, the configuration is carried in a kernel table. For a closed mirrored device, the configuration is in one of the device-dependent fields of the inode.

Just as a standard file is characterized by a read/write/execute mode, each component of a mirrored device can be summarized by `r-w-x` bits, or by a single octal digit. The rightmost field describes the state of the first component.

The meaning of the bits are as follows:

Bit	Meaning
04 (r-bit)	This component is enabled for reading
02 (w-bit)	This component is enabled for writing
01 (x-bit)	This component is undamaged and physically available

You can use the `/etc/mddconf` program to display or change the configuration. If you specify the `-p` option with a new configuration, the value is written to the permanent disk-resident inode. Following are four sample configurations.

Example 1: This example shows a two-component mirrored file system that is enabled in all components for reading and for writing:

```
# /etc/mddconf /dev/mdd/mir
      device name                        r/w mode
--------------------------      ------------------------
/dev/mdd/mir                    ------------------rwxrwx
```

Example 2: This example shows a two-component mirrored file system that reads from a single device but writes to both:

```
# /etc/mddconf /dev/mdd/acs_mir
      device name                        r/w mode
--------------------------      ------------------------
/dev/mdd/mir                    ------------------rwx-wx
```

No matter what is written to the following file system, the data that is read will not change:

```
# /etc/mddconf /dev/mdd/hard_head
      device name                        r/w mode
--------------------------      ------------------------
/dev/mdd/mir                    -------------------wxr-x
```

Example 3: This example shows a two-component mirror; both components are being synchronized with the data in the read-enabled component:

```
# /etc/mddconf /dev/mdd/new
      device name                        r/w mode
--------------------------      ------------------------
```

```
/dev/mdd/mir                        ------------------wx-rwx
```

Though the previous examples show a file system as composed of a single logical device that is a mirror of physical devices, other configurations are possible.

2.10.3 Default Configuration

In software provided by Cray, the default configuration for a mirrored device is `rwx` in each component. For file systems such as the `acs` file system described in one of the previous examples, the default configuration may be changed by a `mirror=` entry in the options field of the `/etc/fstab` description for the file system. For the `acs` example with an SSD in the second mirrored component, the `/etc/fstab` description would be:

```
/dev/dsk/acs  /mount_point NC1FS  rw,mirror=(acs_mir:073)
```

The mirror configurations are specified within the parentheses. The first field is the base name of the mirrored device. The second field, separated from the first by a colon, is the configuration specification given numerically. A leading zero is recommended so that the configuration will be interpreted in octal.

If there is more than one mirrored device in a file system, the semicolon separates entries, as in the following example:

```
mirror=(zero:0337;one:0373;two:0733)
```

If the existing configuration of a mirror does not contain an exercise (`x`) bit in each component, any configuration found in `/etc/fstab` is ignored. The tuned configuration used by Cray software will enable read and write operations in every component that is marked with the `x` bit.

2.10.4 Mirrored Devices during Startup

If a file system containing a mirrored device has been cleanly dismounted, it may be remounted without examination. If a mirrored file system is in an unknown state as the result of a system crash, it must be handled with special processing to prevent different components from being matched in a mirror. There are three programs that help in bringing up a mirrored device.

The first program, `/etc/mdd_pre`, runs before the `fsck`(8) utility. If the file system needs to be checked, it configures all mirrors to a single component read / write configuration; for example, `0117`, `0171`, or `0711`. The configuration permissions ensure that `fsck` sees a consistent set of data.

The second program, /etc/fsck, is used by all file systems, mirrored or not.

The third program, /etc/mdd_post, is used after fsck. Though the file system may be mounted and used as soon as fsck completes, the mdd_post program performs the following steps to finish bringing up the mirrored parts of the file system:

1. Reconfigures the mirror to a wide-write state. In this state, all read operations are completed by the mirrored component used during the fsck step, but the write operations go to every component.
2. Executes /etc/mddcopy. This program uses ioctl calls at the mdd driver level to copy identical information to all components of the mirror.
3. Reconfigures the mirror to the tuned state.

The mdd_pre and mdd_post programs should be placed in the start-up scripts before and after the mfsck(8) command. Experience in the field suggests that mdd_pre fits in /etc/inittab and mdd_post fits in rc.pst.

2.10.5 Manual Startup of Mirrored File Systems

It is possible to bring up a mirrored file system without using the /etc/mdd_pre and /etc/mdd_post programs. This technique might be valuable when there are more recently updated mirror components than the one chosen by mdd_pre.

First, use the /etc/mddconf command to restrict the I/O to a single component of each mirrored device in the file system. Then run the fsck(8) utility. You can repeat the mddconf and fsck step using the -n option on fsck to survey the status of the file system.

When fsck has run to completion and performed the necessary corrections, the file system may be mounted. You can perform the mdd_post processing manually, as outlined above, or you can start the /etc/mdd_post program with a single file system as a parameter.

2.11 Performance Considerations

The following sections discuss device and file system configuration that can affect the performance of the system:

- Logical device cache
- System buffer cache

- Using SSD as a file system
- Secondary data segments (SDS)
- File system placement

2.11.1 Logical Device Cache

The logical device cache provides an excellent means of utilizing the SSD, by providing the capability to assign SSD cache to specified logical devices. The benefit of this type of cache is that predictable and high-speed I/O can be assigned to specific file systems, based on the needs of a particular group or individual and at the discretion of the administrator. It is suggested that you assign cache to `/tmp` and other heavily used file systems.

Cache for logical devices is assigned by the `ldcache`(8) command or with the installation and configuration menu system. Cache may be changed dynamically.

The cache for a logical device is specified as a number of units and a count of 4096-byte blocks per unit. The ability to dynamically alter the cache for logical devices allows you to easily tailor the cache for differing job mixes.

The relationship between number of cache units and the size of each cache unit can dramatically affect the throughput of the cache. For extremely sparse, random I/O, a greater number of units reduces the wait time for a cache unit to become available and increases the probability that data will remain in the cache for a reasonable amount of time. Larger cache units provide higher I/O rates for sequential data access and lower system overhead.

There are some applications for which you might want to bypass the logical device cache when performing I/O on particular files. For example, applications that the perform a large amount of I/O and that access more data than will fit in a cache can significantly decrease system performance because of thrashing in the logical device cache. To bypass the logical device cache, set the `O_LDRAW` flag and the `O_RAW` flag with the `open` system call, as described on the `open`(2) man page.

Each cache unit configured requires a header, which is maintained in main memory. At boot time, you can change the number of logical device cache headers allocated by editing the `NLDCH` parameter in the configuration specification language (CSL) parameter file. If `NLDCH` is not specified in the CSL parameter file, the value specified in `config.h` is used.

The `ldcache`(8) command lets you set logical device cache configuration and display cache statistics. Logical device cache configuration use is restricted to the super user, but users can display cache information.

2.11.1.1 Setting Cache Configuration

To set cache configuration, use the following command line:

```
ldcache -l dev -n units [-s size] [-t type]
```

The options perform the following functions:

`-l` *dev*	Specifies a file system name, or the name or number of a logical device. If *dev* is a device name, it must begin with `/`.
`-n` *units*	Specifies the number of cache *units* to be assigned. If 0, all cache for the device is released.
`-s` *size*	Specifies the *size* of each cache unit in 4096 byte blocks. This option is meaningful only when the `-n` option is nonzero. *size* is typically chosen to be a multiple of the track size of the disk on which the file system resides.
`-t` *type*	Specifies the memory *type* for cache; *type* can be `SSD`, or `MEM` (main memory). The default is `SSD`. This option is meaningful only when specified with the `-n` option.

2.11.1.2 Displaying Cache Statistics

To display cache statistics, you can use either of the following command lines:

```
ldcache -a
   ldcache -l dev -r rate
```

The `-a` option displays devices that have any read or write operations, even though no cache is attached. The `-l` option functions as it does in the configuration command line. The `-r` option specifies the refresh rate for detailed display; the default refresh rate is 1 second.

If you specify no options or arguments, ldcache displays information about all devices with cache in the format of the following example:

```
T  units    size       hits     misses     hit rate      name

S  200       252       66196      361     99.457608     /dev/dsk/drop
S  800        84     1186345      897     99.924447     /dev/dsk/tmp
M  300        42     2250878     4248     99.811629     /dev/dsk/root
S  300        42      618508     1789     99.711590     /dev/dsk/usr
S  100        42      289529     4267     98.547632     /dev/dsk/slash_a
S  100        42      117462     6167     95.011688     /dev/dsk/slash_b
S  100        42      163790     6207     96.348759     /dev/dsk/slash_c
```

In the T (memory type) column, S stands for SSD and M stands for main memory. If you use the -l option to specify a specific device, ldcache displays information about that device at the refresh rate specified (with the -r option) or at the default refresh rate of 1 second.

For example, to receive information on /dev/dsk/root, you would specify the following command line and get the output shown:

```
cray$ ldcache -l /dev/dsk/root

/dev/dsk/root                Tue May 24 09:26:51 1988

                           Read Data           Write Data

Blocks transferred:        23549               4864
Average request size:      2 blks              1 blks
Lst transfer rate:         1.067367 Mbs        0.044813 Mbs
Max transfer rate:         10.751468 Mbs       1.870442 Mbs
Cache hits:                12025               4864
Cache misses:              35                  0
Cache hit rate:            99.709784           100.000000
```

You can use the following commands when viewing the display produced by the -l option of ldcache:

Command	Description
n	Goes to next device with cache attached
+	Increases refresh interval by 1 second
-	Decreases refresh interval by 1 second

c	Clears counters to 0

2.11.1.3 Aging and Threshold Parameters of ldcache

For large applications that perform frequent write requests, you may want to consider using the following options of the ldcache(8) command:

Option	Description
-x *max*, *min*	Specifies, in seconds, aging parameters for units in the logical device cache. When the age of any dirty cache unit exceeds the *max* value, the kernel automatically flushes all dirty units older than the *min* value.
-h *high*, *low*	Specifies threshold values for the dirty units in the logical device cache. The *high* value specifies the maximum number of dirty units that can be in cache at one time. If the number of dirty units equals the *high* value, requests to dirty more cache units are put to sleep until the number of dirty units falls below this threshold value. When the *low* threshold value is exceeded, ldcache starts flushing the oldest dirty units until the *low* threshold is no longer exceeded.

These options implement a *trickle sync* mechanism, which lessens periods of intense disk activity caused by an ldsync(8) command.

The -h parameter of ldcache provides a relatively stable read cache within a larger read/write cache. The -h *max* parameter limits the number of dirty units that can be in the cache. The difference between this value and the size of the cache is the size of the read cache. These extra units are, in effect, reserved for read requests, which are typically more likely to be reused.

To pick effective -h parameter values for ldcache, you need to determine the relative amounts of the different types of I/O requests that are made to each ldcache file system. For example, if most of the requests are read requests, the -h parameter is unnecessary. If most of the requests are write requests and the amount of data written over a relatively short period of time is larger than the cache, set the -h parameter so that there is enough read cache left over to handle the read I/O requests of the applications.

When you use the trickle sync option, you may want to disable the LDSYNCTM parameter. To do this, manually set ldsynctm in the /etc/inittab file or

change LDSYNCTM in the /usr/src/cmd/init/conf.c file to a value greater than 1000000 and rebuild /etc/init.

2.11.2 System Buffer Cache

The amount of system buffer cache configured affects the performance of a system. A cache that is too small degrades system performance; too large a cache wastes memory that could be of use elsewhere. For I/O-intensive jobs that do not use raw I/O, a larger system buffer cache can be used to increase throughput. Experimentation can help determine the optimal number of buffers. System buffers are allocated at boot time in 512-word blocks and use up part of main memory.

You can change the system buffer allocation by using the following menu of the installation and configuration menu system:

```
Configure System
   ->Kernel configuration
       ->Table size parameters
```

At boot time, you can change the number of system buffers allocated by editing the NBUF parameter in the CSL parameter file.

The effectiveness of your system buffer cache and I/O in general can be monitored by using the sar(8) command.

2.11.3 Using SSD As a File System

SSD can be used as a logical partition. The SSD can be configured as a logical device and mounted individually, or grouped with other logical slices and mounted as a logical device. System performance can be improved by using the SSD as a logical device where access time is critical to system performance (for example, for file systems such as /tmp, /bin, or /lib). File systems that are used heavily can be mounted on the SSD to increase throughput and reduce I/O wait time.

As a file system, the SSD can be configured in the same manner as disk devices; that is, it is configured as one or more slices each having a starting block number and number of blocks.

One or more of these slices may be used as logical devices upon which file systems can be built. In addition, SSD slices can be combined with disk slices to form logical devices.

Note: SSD data can be lost across system power failures. This must be taken into account when deciding whether or not a file system should span both disks and SSD. It is recommended that file systems reside completely on disk and that logical device cache blocks be assigned to the SSD (see Section 2.11.1, page 51.

2.11.3.1 SSD Memory Access

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

This release supports the Cray SV1ex GigaRing based system with

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

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

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

Back door access is defined as the movement of data directly between SSD-I and disk devices. Back door transfers can occur over either a conventional point-to-point GigaRing or a SuperRing consisting of two to four mainframe GigaRing adapters. Back door transfer rates are a function of the number of

mainframe nodes, disk controllers, and disk devices, connected to the GigaRing or SuperRing channel. Whereas, the number of rings that can be configured is dependent on the number of processor modules in the system.

2.11.3.2 Back Door I/O Rules

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

1. The minimum configuration is a point-to-point ring consisting of one mainframe GigaRing connection and one ION, where the ION is an FCN, IPN, HPN, or disk or network MPN (not tape).
2. Alternatively, a SuperRing configuration, consisting of two to four mainframe GigaRing connections and one or more IONs on a single ring, can in some cases provide improved I/O bandwidth. See Section 2.11.3.3, page 57 for more information.
3. Although a ring that is set up to allow back door I/O can also be used for I/O transfers between disk and main memory, no networking or tape IONs can exist on the same ring. Any attempt to open a tape device on a ring configured with back door capability will fail and generate an error message on the system console.
4. Back door I/O access to ldcache in the SSD is limited to file systems that exist entirely on disk partitions that reside on properly configured back door-capable rings, including both primary and alternate I/O paths.
5. Back door I/O is required to use the SSD for secondary data segment (SDS) space (to support SDS space swapping).

2.11.3.3 SuperRing Configuration Rules

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

1. A SuperRing is defined to be a single GigaRing consisting of two to four mainframe GigaRing channel adapters (on either processor modules or I/O modules) and one or more IONs on a single ring. Although a SuperRing can be configured to support front door (regular) I/O, a SuperRing is primarily intended to support back door (direct disk to SSD) I/O transfers.
2. There is no maximum supported configuration on a single SuperRing. However, if the total number of mainframe GigaRing adapters and IONs on a ring is greater than 6, performance might be degraded significantly.

3. A SuperRing configuration of three mainframe GigaRing connections with two FCNs provides the best balanced overall capability for back door I/O. The FCNs can move data at ~300 Mbyte/s and the mainframe nodes operate at 200 Mbyte/s.
4. Multiple SuperRings can be configured on a single mainframe; however, a SuperRing can not be a shared ring between multiple mainframe nodes.
5. On systems that use an 8x8 backplane, all the mainframe GigaRing adapters that make up a single SuperRing must be on the same side of the mainframe. (CPU slots 0-3 are on one side, 4-7 are on the other.)
6. On systems that use a 4x4 backplane, only one SuperRing consisting of either two or three mainframe GigaRing adapters is possible unless the system boot ring is configured as a SuperRing.

2.11.4 Secondary Data Segments (SDS)

Secondary data segments (SDS) is a feature that allows a part of the SSD to be used as extended memory. This area must be defined in the parameter file.

A user can specify that a file resides on the SDS by using the `assign`(1) command. Users then make requests to either expand or contract their SDS field length. SDS is automatically released when the owning process terminates.

The system maintains a base and limit address for the SDS area of each process. All I/O requests to SDS are relative to address `0`. The simple mapping scheme allows use of a short-circuited path to process I/O requests to SDS, providing transfer rates up to 10 times higher than that of SSD file systems. The system calls `ssbreak`(2), `ssread`, and `sswrite` (see `ssread`(2)) are discussed in *UNICOS System Calls Reference Manual*.

The status of processes using SDS space can be determined with the `sdss`(1) command.

The UNICOS operating system also supports direct data transfers between SDS and disk files through the back door or side door channel.

Allocation of a file on SDS is accomplished by opening a disk file with the `O_SSD` flag set (see `open`(2)). All read/write addresses are then treated as relative to SDS.

Fortran I/O library support allows particular files to be assigned to SDS in a program-transparent manner. See `assign`(1) and `env`(1) in the *UNICOS User Commands Reference Manual*.

For a discussion of SDS management in the batch environment, refer to the *NQE Administration*.

2.11.5 File System Placement

The `/usr` partitions should be on a different disk drive than the `root` (`/`) partition(s) to reduce disk I/O contention. In addition, system core dumps should be copied to another partition so the root partition is not filled.

The `/usr/spool` and `/usr/adm` directories should be on separate file systems from the `/usr` file system so that Network Queuing System (NQS) and accounting are not interrupted if `/usr` becomes filled.

User directories should not be located on either the `root` (`/`) or `/usr` partition to prevent users from filling these partitions.

Startup and Shutdown Procedures [3]

This chapter describes the startup and shutdown of a UNICOS system. It discusses the following administration areas:

- System initialization
- System shutdown
- Run-level configuration
- System multiuser startup

Warning: This chapter contains warnings and information critical to the proper use of a Cray ML-Safe system configuration.

3.1 System Initialization

The following sections describe how to initialize your Cray system. The following tasks are covered:

- Deadstarting the system
- Initializing the UNICOS operating system
- Setting the system date and time
- Setting the system time zone

3.1.1 Deadstarting the System

The procedures for deadstarting a UNICOS system vary according to the type of I/O hardware and system hardware. Different procedures exist for GigaRing based systems (which use the system workstation (SWS)) and for systems based on an I/O subsystem model E (IOS-E) (which use an operator workstation (OWS)).

To start a GigaRing based system, use the SWS `bootsys`(8) command. For information about this command and the SWS, see the *SWS-ION Administration and Operations Guide* and the *SWS-ION Reference Manual*.

To start an IOS-E based system, issue the `load` command at the `BOOT>` prompt on the OWS, and then issue the `/bin/boot` command to start the UNICOS operating system.

3.1.2 Initializing the UNICOS Operating System

The final phase in starting your system is to initialize the UNICOS operating system. This procedure requires that you check and reset the UNICOS time, check the UNICOS file systems, and enable the network connections. The following procedure details a typical initialization. The UNICOS initialization procedure is customized for each site; messages and questions not explained in this chapter may appear during normal UNICOS initialization at your site.

If your machine is an IOS-E based system, use the `zip` window on the OWS. If your machine is GigaRing based, use the `mfcon`(8) window on the SWS.

Perform the following steps:

1. To start multiuser mode (when the system is in single-user mode), enter the following command from the console window (OWS or SWS):

   ```
   /etc/init 2
   ```

   Note: Most commands typed on the UNICOS interactive console must be entered in lowercase.

2. The date verification prompt appears as follows:

   `Is the date` *dow MM dd hh:mm:ss tz yyyy* `correct? (y or n)`

   The *dow* variable is the day of the week, *MM* is the month, *dd* is the day of the month, *hh* is the hour in 24-hour time, *mm* is the minute, *ss* is the second, *tz* is the time zone acronym, and *yyyy* is the year.

   Example:

   ```
   Is the date Thu Mar 17 15:57:13 CST 1995 correct? (y or n)
   ```

   Respond by typing `y` for yes, or `n` for no.

   If the date or time is wrong, see Section 3.1.3, page 63.

   If the time zone is wrong, see Section 3.1.4, page 63.

3. On a Cray IOS-E based system or on a GigaRing based system, if a dump of the system had been done previously, the system prompts as follows:

   ```
   Enter reason for sysdump:
   ```

   Respond by entering the reason for the dump, followed by `RETURN`.

The system continues with the following question:

```
Do you want to copy the dump to another file system? (y or n)
```

If you respond with `y`, the dump is written to the file `/core/core.`*MMdd.hhmm*, where *MM*, *dd*, *hh*, and *mm* represent the current month, day, hour, and minute, respectively. Otherwise, the dump is written to `/core.sys`. (The location of this file is determined by the options given to `coredd` in `/etc/bcheckrc`.)

Note: If the `/core.sys` file is not copied or moved to another file, the dump is overwritten when a new system dump is written to `/core.sys`.

3.1.3 Setting the System Date and Time

The UNICOS system date and time can be set at boot time (or during operation with the `root` login ID). This section describes the procedure for setting the date and time at boot time.

To change the date and time at boot time, respond with `n` when asked the following question:

`Is the date` *dow MM dd hh:mm:ss tz yyyy* `correct? (y or n)`

Enter the correct date in the following format:

MMddhhmm[*yy*[*ss*]]

The *MM* variable is the month, *dd* is the day of the month, *hh* is the hour in 24-hour time, *mm* is the minute, *yy* is the last 2 digits of the year (optional entry), and *ss* is the second (optional entry).

Example:

```
0502161595
```

This example changes the date and time to May 2, 1995, 4:15 P.M.

Verify the date and time that the system prints and continue with the startup procedure.

3.1.4 Setting the System Time Zone

The UNICOS kernel clock keeps track of the date and time relative to Greenwich mean time (GMT). The software must convert between GMT and local time whenever the time is entered or displayed. Local time refers to the time zone

where the machine is located. The UNICOS release software is configured for use in the Central time zone (CST) of the United States. You can change the time zone used by the conversion routines if your system is not located in the Central time zone.

Note: Binary-only sites should set their time-zone information in the TZ environment variable, as described in the following sections. Ignore the references to source code and simply follow the formats shown for setting the TZ variable (`TZ=...`).

3.1.4.1 Time-zone Information

To change the time zone in the UNICOS operating system, you must have the following information about your time zone:

1. The number of hours away from GMT
2. The direction from GMT (East or West)
3. The name of the time zone
4. Whether your area has daylight savings time
5. Whether the UNICOS algorithm for daylight savings time is correct for your site's location

The daylight savings time conversions used in the UNICOS operating system are valid for most parts of the United States, but may need to be changed for other countries. The conversion algorithm is located in the routine `/usr/src/lib/libc/gen/ctime.c` (see `ctime`(3)).

The library routines in `/usr/src/lib/libc/gen/ctime.c` determine the daylight savings time rules by using a table containing limits for the dates on which the time change takes place. The algorithm assumes that the time change takes place on a Sunday at 0200 local time.

Table 2 lists the time zones for the United States.

Table 2. United States Time Zones

Zone name	Hours from GMT	Acronym
Eastern	5	EST
Central	6	CST

Zone name	Hours from GMT	Acronym
Mountain	7	MST
Pacific	8	PST

All the time zones in the United States are west of GMT. The release software is shipped for Central time, which is 6 hours west of GMT, with daylight savings time enabled and the time zone names of CST and CDT. If your time zone is not in Table 2, page 64, find out the correct information for your area by calling Software Product Support (SPS).

To change the time zone, edit the `/etc/inittab` file and change the `tz` line to be correct for your site. The `tz` line in the release is as follows:

```
tz::timezone:TZ=CST6CDT
```

For example, to change to Eastern time, the `tz` line would be:

```
tz::timezone:TZ=EST5EDT
```

For sites located in areas that do not have daylight savings time, specify only one time-zone name, as follows:

```
tz::timezone:TZ=EST5
```

For sites east of GMT, use a minus sign before the number of hours difference between local time and GMT, as follows:

```
tz::timezone:TZ=SOT-1
```

For the time-zone change to take effect, you must reboot the UNICOS operating system.

3.1.4.2 Time-zone Example 1

The following is an example of setting the time zone for a site located in Geneva, Switzerland. There is no official time-zone acronym for this area, so the acronym MET (Middle European Time) will represent the time zone that is one hour east of GMT. Daylight savings time exists in this time zone; however, it begins and ends on days other than those in the United States.

This example uses the following conditions:

Geneva is 1 hour east of GMT. For this example, the time zone is named Middle European Time (MET). Geneva has daylight savings time, so two names must be specified: one name for standard time, and one name for daylight savings

time. These two names can be the same. The name MET will be used for both standard and daylight savings time.

This example assumes the following data:

Number of hours away from GMT:	1
Direction from GMT (East or West):	East
Time zone name (standard):	MET
Time zone name (daylight savings):	MET
Beginning day of daylight savings time:	Last Sunday in March
Ending day of daylight savings time:	Last Sunday in September

You change the time-zone line in the file `/etc/inittab`. You use the data given above to modify the `tz` line released with the UNICOS operating system. You give the name of the time zone when using standard time (in the following example: `MET`), followed by a plus or minus sign to reflect the sites' direction east or west of GMT (in the following example: ' - '), followed by the number of hours from GMT (in the following example: `1`), followed by the name of the time zone when using daylight savings time (in the following example: `MET`).

In the following example, the `M3.5.0` is the day that daylight savings begins: March (`3`), the last (`5`) Sunday (`0`). The `M9.5.0` is the time daylight savings ends: September (`9`), the last (`5`) Sunday (`0`).

The released and modified `tz` lines follow:

Released: `tz::timezone:TZ=CST6CDT`

Modified: `tz::timezone:TZ=MET-1MET,M3.5.0,M9.5.0`

3.1.4.3 Time-zone Example 2

The rules for the United Kingdom are different. Daylight savings time runs from the last Sunday in March to the Sunday after the fourth Saturday in October. For 1995, you would specify:

```
tz::timezone:TZ=GMT0BST,M3.5.0,M10.5.0
```

3.2 System Shutdown

You must perform several steps when shutting down the UNICOS operating system. These steps must be performed in the indicated order. However, you can stop the shutdown procedure when you reach single-user mode if you are

doing maintenance procedures that require the UNICOS operating system to be at the single-user run level.

Warning: You should never remove power from a peripheral device that is in service, and never power down the system until the UNICOS operating system has been halted.

The `shutdown`(8) script performs most of the shutdown functions. The `shutdown` script is described in detail in the following section. A subsequent section describes a sample shutdown procedure and gives an example of a typical shutdown session.

Warning: If your site is running a Cray ML-Safe system configuration, see Section 3.2.3, page 70, for more information.

3.2.1 The `shutdown` Command

The `shutdown`(8) shell script provides an orderly method of shutting down the system. Whenever the system must be shut down, such as for a reboot, run the `shutdown` command. You may find it necessary, or convenient, to change the script to accommodate your local system configuration.

The `shutdown` script released with the UNICOS operating system provides three user exits (`shutdown.pre`, `shutdown.mid`, and `shutdown.pst`) that allow you to modify the shutdown process.

You can specify the grace period allowed between the sending of a warning message and the actual shutdown. This grace period is the number of seconds of delay. For example, specifying a grace period of 300 results in a 5-minute delay. If no period is specified, the default period is 60 seconds.

The `shutdown` script released with the UNICOS operating system performs the following functions:

- Determines the user's current directory and does not proceed if the current directory is not the root directory (`/`), `/etc`, `/ce`, or `/sysops`.
- 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.
- Sends a message, using `wall`(8), warning the users who are currently logged in to the system that the system is being shut down.
- Shuts down the NQE subsystem to allow batch jobs to be checkpointed before they are terminated.

- Sends a SIGSHUTDN signal to all currently running processes.
- Stops the DM daemon, Tape daemon, and error logging.
- Stops SYS1 and SYS2 type daemons, using the /etc/sdaemon command.
- Sends a SIGHUP signal to all currently running processes.
- Sends a SIGKILL signal to all currently running processes.
- Shuts down system accounting, using the following command (see acctsh(8)):

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

  The shutacct(8) command records the action of shutting down system accounting in the /etc/wtmp file.
- Releases all logical device cache (ldcache).
- Runs 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.
- Shuts down all configured network interfaces (defined in the /etc/config/interfaces file), using the ifconfig(8) command.
- Runs 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).
- Unmounts all file systems. If any local file systems cannot be unmounted, the shutdown script will issue a warning message.
- Brings the system to single-user mode, using the init(8) command with an s argument, as follows:

  ```
  /etc/init s
  ```

When shutdown completes, the UNICOS operating system is in single-user mode. The administrator can safely perform system-related work.

3.2.2 System Shutdown Configuration

To allow the shutdown process to be tailored to the needs of the site, the shutdown(8) script provides three user exits: shutdown.pre, shutdown.mid, and shutdown.pst. The following sections describe these user exits.

3.2.2.1 The `shutdown.pre` User Exit

The `shutdown.pre` script is the first user exit of the `shutdown` script.

Warning: Modification (and / or creation) of this script must comply with the guidelines set forth in the single-user mode descriptions in Chapter 8, page 145. Additionally, the Cray ML-Safe configuration is available to normal users at the time that this script is executed. Therefore, additional care must be taken to ensure that the operations performed by this script follow the same rules and restrictions that are enforced for a security administrator, as described in Chapter 8, page 145.

If an executable named `/etc/shutdown.pre` exists, it will be executed during shutdown. At this point, nothing has been done in 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.

3.2.2.2 The `shutdown.mid` User Exit

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, the disk cache (ldcache) 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.

3.2.2.3 The shutdown.pst User Exit

The shutdown.pst script 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 continue 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 the space, shutdown.pst could potentially fill up the file system(s) containing the log files.

3.2.3 System Shutdown Procedures

To shut down the system, perform the following steps:

1. On a UNICOS system with PRIV_SU enabled, log in as root on the system console. On a system with PRIV_SU disabled (for example, a Cray ML-Safe system configuration), you must log in to the security administrator login and have an active secadm category to shut down the system. You will also need to activate the secadmin category. For more information, see Section 8.2.3.2, page 154, and Section 8.2.2, page 150.

2. Use wall(8) to give users a 10-minute warning. The wall command can read either a file or standard input for the message it sends. For more information about using the wall command to communicate with users, see Section 5.3.1, page 112.

   In the following example, the administrator types a shutdown message (terminated by CONTROL-d) into standard input; wall broadcasts this message.

```
% /etc/wall
UNICOS is coming down in 10 minutes.
CONTROL-d
```

In addition to sending a shutdown message by using `wall`, it is recommended that you put a short message in the `/etc/issue` file (see `issue`(5)) to inform any users who log in after the warning has been sent that the system will be coming down shortly.

3. Enter the following command to begin the shutdown procedure:

   ```
   shutdown
   ```

   All users logged in to the system are instructed by a broadcast message to log off the system.

   A typical session with the `shutdown` program is as follows:

```
# shutdown 300
SHUTDOWN PROGRAM
Thu Sep   1  18:51:58 EST 1992
Executing user exit: /etc/shutdown.pre
User exit /etc/shutdown.pre completed, continuing shutdown
Do you want to send your own message (y or n):y
Type your message followed by <ctrl>d...
System coming down for maintenance!
Please log off.
CONTROL-d
System coming down for maintenance!
Please log off.


(Waits for 5 minutes)


SYSTEM BEING BROUGHT DOWN NOW ! ! !
```

Warning: Cray ML-Safe processing occurs only when in multiuser mode (level 2).

When the `shutdown` program completes, it displays the following message:

```
INIT: SINGLE USER MODE
```

4. Issue the `sync`(1) command as follows to flush the file system cache to disk:

   ```
   sync
   ```

   Pause approximately 30 seconds before continuing with the following step.

5. Issue the `ldsync`(8) command as follows to flush data from all logical device caches to disk:

   ```
   ldsync
   ```

   At this point, you are in single-user mode and can perform any system administration work that is necessary.

6. If you want to halt the system, perform one of the following steps, depending on your hardware:

   - On GigaRing based systems, use the SWS `haltsys`(8) command. For information about this command, see the *SWS-ION Administration and Operations Guide*.
   - On Cray IOS-E based systems, type `ehalt`(8) in a local OWS window, or use the `xhalt` function in the `opi` interface.

3.3 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.

There are two main modes of operation for the UNICOS operating system: single-user (level s) and multiuser (level 2). Single-user mode is always indicated by run level s or S. Multiuser mode is typically run level 2; however, the system can be configured to run in multiuser mode at any level from 0 to 6.

Warning: Cray ML-Safe processing occurs only when in multiuser mode (level 2).

The characteristics of each run level are determined by information supplied in the /etc/inittab file. The /etc/inittab file contains entries that specify actions associated with run levels. Refer to the inittab(5) and init(8) man pages for more information about the format of the /etc/inittab file.

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, normally you are asked to verify the date and 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.

3.3.1 Changing Run Level

You can change your run level by issuing the following command. *level* is the run level you want to initiate:

/etc/init *level*

The specific actions that occur when a run level is initiated are controlled by the /etc/inittab file. The following sections discuss the strategies for using run levels for different purposes.

3.3.2 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 needs to 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, there should be one or more run levels devoted to initializing the system for this dedicated use. In all cases except for single-user mode (which requires little or no initialization), initialization is carried out by the rc script (see brc(8)).

3.3.2.1 Single-user Mode

Many system maintenance, modification, testing, configuration, and repair procedures are carried out 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.

Warning: When you enter single-user mode, you exit the Cray ML-Safe secure processing mode. Use of the Cray ML-Safe configuration in single-user mode by an authorized administrator does not violate the requirements of the Cray ML-Safe configuration.

When the UNICOS operating 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 allows you to make necessary changes to the system without doing any other processing. The # character is the system prompt when the UNICOS operating system is in single-user mode.

In most cases, the system is brought into single-user mode either following a system boot or by `shutdown`. In neither case should there be any user processes 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.

It is important that all file systems be unmounted. This provides an extra measure of protection against inadvertent damage done to a mounted file system by single-user mode development work or testing. Traditionally, the person doing the system work or testing while in single-user mode mounts only the partitions he or she requires. To help with this aspect of system work, you can provide a script in `/etc` that mounts the file systems containing system commands not usually found on the root partition (the `/usr` file system) and the home directories of the system staff.

3.3.2.2 Multiuser Mode

Traditionally, run level 2 is the system's primary run level for multiuser mode.

Warning: If your site is running a Cray ML-Safe system configuration, see Chapter 8, page 145, for information on the correct configuration.

Among the initializations generally carried out for multiuser mode are the following:

- Recording system startup time in `/etc/wtmp`.

- Mounting all the file systems required for normal system operation. This includes the regular system file systems (`/usr` and `/tmp`), the file system or systems containing the home directories of the system's users, and other file systems containing 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, when enabled in `/etc/recoptions`)
  - `syslogd`
- Running the `netstart`(8) 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.

You can perform some system administration tasks in multiuser mode. For example, you can perform file restore procedures (if necessary) and take periodic status checks of the system. These status checks can include the following:

- A check of free blocks (`df`(1)) remaining on all mounted file systems to ensure that a file system does not run out of space
- A check on mail (`mail`(1)) to `root` or whatever login receives requests for file restores
- A check on the number of system users (`who`(1))
- A check of all running processes (`ps -eaf`) to determine whether some process is using an abnormally large amount of CPU time

You may want to prepare additional multiuser modes that have the following characteristics:

- They do not mount certain file systems that should be kept secure from the system's users.
- They do not provide certain services or run certain daemons.
- They do not permit users to log in to the system by using certain pieces of software (for example, TCP/IP).

3.3.2.3 Dedicated System

It is sometimes necessary to provide dedicated system time so that a particularly large or time-critical job can run unencumbered by other user processes. Additionally, there 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.

Access to a UNICOS system can be restricted by having the `rc` script (see `brc`(8)) alter the `ue_permbits` field in the user database (UDB) records to allow system access only to selected users.

3.3.3 Files That Control Run-level Activity

Certain files control various aspects of a system's run levels. The following sections describe those files in detail.

3.3.3.1 The `/etc/inittab` File

The `/etc/inittab` file controls the system's available run levels. The `inittab`(5) man page explains the file's format and options in detail. This file should have the following attributes:

- The initial run level (specified by an entry with the action `initdefault`) should be single-user mode (specified by the letter `s` in the `rstate` field).
- Following the `initdefault` entry, there should be an entry with the action `timezone` to set the `TZ` environment variable to the appropriate value for the time zone in which the system is located.
- Following the `timezone` entry, there should be calls to shell scripts that actually initialize the system's state for the run level being entered.

  The `bcheckrc` (see `brc`(8)) program is called by an entry with the action `bootwait` to carry out boot-time-only actions, and the `rc` (see `brc`(8)) program is called by an entry with the action `wait` to carry out actions for

switching from one run level to another (including switching from the initial single-user mode to multiuser mode).

- There should be an entry with the action `wait` that links the special file `/dev/systty` to `/dev/syscon`.
- There must be an entry for all run levels, with an action of `respawn`, which executes the `getty`(8) command.
- 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 if the run level is intended for use by a single dedicated user; restricting access to the system is accomplished by the `rc` script (see Section 3.3.3.3), not by limiting logins to specific devices, as is often done on traditional UNIX systems.

3.3.3.2 The `/etc/bcheckrc` Script

By convention, the `bcheckrc` script (see `brc`(8)) is executed only at system boot time, before any file systems are mounted. Its main function is to set the current date and time, prompting the person booting the system for the current date, if necessary, and using the `date`(1) command to set the date.

3.3.3.3 The `/etc/rc` Script

The `/etc/rc` script is traditionally executed when the `init` process goes from run level s or 1 to a run level greater than 1. By default, the UNICOS operating system has a standard `/etc/inittab` file that is designed to activate the `/etc/rc` script when the following command is executed:

```
/etc/init 2
```

3.4 System Multiuser Startup

The `/etc/rc` script supplied by Cray is installed in the `/etc` directory. Its source is in file `/usr/src/skl/etc`. The file includes optional execution of local scripts at special points in the `rc` script. The `/etc/rc` script is designed so that it does not need to be modified for most sites.

Note: Previously, sites have heavily modified `/etc/rc` to suit their specific needs. Because the `/etc/rc` script is now unconditionally installed, you should not modify it. Please notify Cray of any modifications you found necessary so that your requirements can be considered for future releases.

When it is invoked by /etc/init, the /etc/rc script performs several operations that are required to bring the UNICOS operating system into multiuser mode. The following sections provide more detailed explanations of the events and files referenced by the /etc/rc startup script. Refer to the brc(8) man page for additional information on the rc script.

3.4.1 Load the /etc/config/rcoptions File

If there is a file named rcoptions in the /etc/config directory, the /etc/rc script uses the standard shell command dot (.) to *source* the file into its environment variable name space. Existing variables are overridden.

Warning: If your site is running a Cray ML-Safe system configuration, see Chapter 8, page 145, for information on the correct configuration.

If you are using the Installation and Configuration Menu System, the rcoptions (shown in Table 3) can be defined on the Configure System->Startup (/etc/rc) configuration menu. The parameters can have one of three values:

- YES
- NO
- ASK

If the parameter is set to YES or NO, the associated action is taken or skipped unconditionally; that is, the operator is not prompted for a response. If the parameter is set to ASK, the operator is interrogated at boot time; each parameter description begins with the associated operator prompt.

Table 3. rcoptions Decide String Parameters

Parameter	Description
RC_ACCT	(Do you want to start accounting?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the accounting subsystem is activated. The default value for this parameter is YES.

Parameter	Description
RC_CONTERR	(Do you want to continue with system startup?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then the startup will continue in the face of errors from any commands or utilities during system startup. A selection value of NO instructs /etc/rc to terminate system startup and immediately return to single-user mode upon encountering an error during system startup. A selection value of YES instructs /etc/rc to enter multiuser mode regardless of any errors encountered during startup. The default value for this parameter is ASK.
RC_DCE	(Do you want to start DCE?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then DCE is started. The default value for this parameter is ASK.
RC_DFS	(Do you want to start DFS?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then DFS is started. The default value for this parameter is ASK.
RC_FSCK	(Do you want to execute fsck?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then mfsck will run on each file system listed in /etc/fstab. Setting this to NO will still cause mfsck to run fsck, but it will do so in quick mode (fsck -q). The default value for this parameter is YES.

Parameter	Description
RC_FSCK_Y	(Do you want to automatically respond yes to the fsck prompts for ... ?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then all the fsck invocations during the boot process will have the -y flag present. The -y flag's presence causes fsck to assume a YES response to all of its prompts. The default value for this parameter is NO.
RC_MKTMP	(Do you want to mkfs /tmp?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, mkfs(8) is used to remake the /tmp file system device. If this parameter is set to NO, fsck(8) is used to check the /tmp device. The default value for this parameter is ASK.
RC_MKUTMP	(Do you want to mkfs /usr/tmp?). If there is a /usr/tmp device configured, this parameter is checked; otherwise, it is ignored. If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, mkfs(8) is used to remake the /usr/tmp file system device.If this parameter is set to NO, fsck(8) is used to check the /usr/tmp device. If the /usr/tmp file system device is remade, recovery of the vi or ex editing file is impossible.
RC_NET	(start the network?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the /etc/netstart script is executed. The default value for this parameter is YES.

Parameter	Description
RC_NFS	(Do you want to start NFS?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then NFS is started. The default value for this parameter is ASK.
RC_SADC	(Do you want to start the system activity daemon?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the system activity data collection process is activated. The default value for this parameter is YES.
RC_SMT	(Do you want to start the FDDI SMT daemon?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the FDDI SMT daemon is started. The default value for this parameter is ASK.
RC_SSHD	(Do you want to start the SSH daemon?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the SSH daemon is started. The default value for this parameter is ASK.
RC_TAPE	(Do you want to start the tape daemon?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the /etc/tpinint command is executed. The default value for this parameter is ASK.

Parameter	Description
RC_TCP	(Do you want to start TCP?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the networks are started. The default value for this parameter is ASK.
RC_USRMNT	(Do you want to mount the user file systems?) If the /etc/fstab file exists, this parameter is checked; otherwise, it is ignored. If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the file systems in /etc/fstab are mounted. The /etc/fstab file has the following format: *file_system_device_name mount_point_directory* (If you are using the configuration tool, select fstab, load your /etc/fstab file, and modify the File system configuration menu.) The default value for this parameter is YES.
RC_YP	(Do you want to start NIS (formerly YP)?) If this parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, then NIS is started. The default value for this parameter is ASK.

If you are using the Installation and Configuration Menu System, the security rcoptions parameters (shown in Table 4, page 83) can be defined on the Configure System->Multilevel Security (MLS) Configuration->MLS System Options menu.

Table 4. `rcoptions` Non-decide String Parameters

Parameter	Description
`RC_CRONLOGDIR`	The path to the `cron` log directory. The default value for this parameter is `/usr/lib/cron`.
`RC_LOG`	The file name in which the startup entries and various information is sent. The default value for this parameter is `/etc/rc.log`.
`RC_NFSLOG`	The file name of the NFS log. The default value for this parameter is `/etc/nfs.log`.
`RC_SECLOW`	The lower security level (used by `mkfs`(8) to construct `/tmp` and `/usr/tmp`). The default value for this parameter is 0.
`RC_SECHIGH`	The upper security level (used by `mkfs`(8) to construct `/tmp` and `/usr/tmp`). The default value for this parameter is 0.
`RC_SECMASK`	The security compartments mask. The default value for this parameter is `none`.

If you are using the Installation and Configuration Menu System, the following variables (shown in Table 5, page 83) can be defined in the `Configure System->Special system device definitions` menu.

Table 5. `rcoptions` File System String Parameters

Parameter	Description
`DUMPDEV`	The dump device used by the `dumpsys`(8) utility. It should be assigned the value of `dmpdev` in the configuration file on the SWS. The default value for this parameter is `" "`.

Parameter	Description
DUMPDIR	The directory created on the file system (DUMPFS) into which the dump directory is created. The value of DUMPDIR is passed to /etc/coredd, which creates a unique directory for the dump (*mmddhhmm*) and copies the dump and current versions of /unicos and /etc/crash to that directory. The directory structure looks like the following: /DUMPMPT/DUMPDIR/*mmddhhmm* The default value for this parameter is "".
DUMPMPT	The mount point onto which the file system (DUMPFS) is temporarily mounted for /etc/coredd to copy the dump onto when the system is brought up in multiuser mode. If this directory does not exist, it is created by /etc/coredd. The default value for this parameter is /mnt.
DUMPFS	The file system to which a system dump is copied when the system comes up in multiuser mode. The value of DUMPFS is passed to /etc/brc, which calls /etc/coredd. The file system is temporarily mounted on the selected mount point (DUMPMPT), and the dump is copied onto that file system. The file system is unmounted and then mounted in its usual place (specified by fstab) as the system comes up. The default value for this parameter is core.
MNTTMPOPTS	The temporary (/tmp) mount options. This specifies any miscellaneous options for the mount(8) command (for example, quotas or number of inodes) used to initialize /tmp. Note that any values specified for the -o or -t options will be overridden. The -o option is forced to rw in /etc/rc, and the -t option is forced to NC1FS. The default value for this parameter is "".

Parameter	Description
MNTUTMPOPTS	The user temporary (/usr/tmp) mount options. This specifies any miscellaneous options for the mount(8) command (for example, quotas or number of inodes) used to initialize /usr/tmp. Note that any values specified for the -o or -t options will be overridden. The -o option is forced to rw in /etc/rc, and the -t option is forced to NC1FS. The default value for this parameter is "".
PIPEDEV	The device name of the pipe file system. It is typically the same as ROOTDEV. It should be assigned the value of pipedev in the configuration file on the SWS. The default value for this parameter is root.
ROOTDEV	The device name of the / file system, or the root file system device. It should be assigned the value of rootdev in the configuration file on the SWS. The default value for this parameter is root.
SRCDEV	The file system that contains the UNICOS source. It is used in single-user mode to mount /usr/src and used by secure labeling to install trusted subject programs. The default value for this parameter is src.
SWAPDEV	The swap logical device name. It should be assigned the value of swapdev in the configuration file on the SWS. The default value for this parameter is swap.
TMPDEV	The device name of the /tmp file system. The /etc/rc script must mount the /tmp file system to facilitate the system boot. Optionally, /etc/rc makes the tmp system device or uses fsck(8) to check it and then mounts it. The default value for this parameter is tmp.
TMPOPTS	The temporary (/tmp) mkfs(8) options. This specifies any miscellaneous options for the mkfs(8) command (for example, quotas) used to initialize /tmp. The default value for this parameter is "".

Parameter	Description
USRDEV	The logical device of the /usr file system device. The /etc/rc script must mount the /usr file system to enable access to administrative directories. The default value for this parameter is usr.
USRTMPDEV	The device name of the /usr/tmp file system device, if it is configured. This parameter cana be set to blank ("") in /etc/config/rcoptions. A blank value indicates that /usr/tmp is part of the /usr file system device; in this case, the parameter is ignored. If this value is not blank, /etc/rc can (optionally) make and mount the file system. The default value for this parameter is usr_tmp.
USRTMPOPTS	The user temporary (/usr/tmp) mount options. This specifies any miscellaneous options for the mkfs(8) command (for example, quotas) used to initialize /usr/tmp. The default value for this parameter is "".

3.4.2 Set up the /etc/rc Log File

The /etc/rc log file is defined by the RC_LOG parameter; output from the /etc/rc script is captured in this file. The log file is reused (overwritten) for each boot. If the RC_LOG parameter is set to a blank value, all output goes to /dev/console. The RC_LOG parameter is exported so that scripts called by /etc/rc can also access the log file.

3.4.3 Execute /etc/rc.pre

If the local executable script /etc/rc.pre exists, it is executed at this time. That is, the /etc/rc.pre script is executed after the /etc/rc log file is established and before any file system mounts are done.

This file is not supplied by Cray.

Warning: This script does not meet the requirements of a Cray ML-Safe system configuration.

3.4.4 Make and Mount /tmp

If the RC_MKTMP parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the /tmp file system device is created. It is always checked (using fsck(8)) and then mounted at this time.

3.4.5 Mount the /usr File System

The /etc/rc script mounts the /usr file system. The file system is not checked with fsck(8), because it is usually in the /etc/fstab table, and, therefore, is checked before /etc/rc is executed (see bcheckrc on the brc(8) man page and fstab(5)).

3.4.6 Make and Mount /usr/tmp

If you have a /usr/tmp file system device, this action is taken. If the RC_MKUTMP parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the file system is remade. It is always checked (using fsck(8)) and then mounted.

3.4.7 Preserve Interrupted vi/ex Sessions

If the /usr/tmp file system was not remade, expreserve (see vi(1)) is executed to preserve any interrupted vi or ex(1) editing sessions.

3.4.8 Mount User File Systems

If the /etc/fstab file exists, /etc/rc uses the list it contains to mount the user file systems. Depending on the number of user file systems configured, it may take a significant amount of time to mount them; therefore, a speedometer is displayed at this point in the /etc/rc script.

3.4.9 Mount /proc

The /proc file system mount point is set, and the /proc (special) file system is mounted. Because various UNICOS subsystems require the presence of the /proc file system, this step is mandatory.

3.4.10 Activate Logical Device Cache

If the `/etc/config/ldchlist` file exists, the `/etc/rc` script uses the list it contains to assign cache to the designated file systems. See `ldcache`(8) for more information.

3.4.11 Execute `/etc/rc.mid`

If the local executable script called `/etc/rc.mid` exists, it is executed at this time. That is, the `/etc/rc.mid` script is executed after the user file systems are mounted, but before any administrative cleanup is done or any daemons are active.

This file is not supplied by Cray.

Warning: This script does not meet the requirements of a Cray ML-Safe system configuration.

3.4.12 Perform Administrative Cleanup

The `/etc/rc` script performs the following cleanup operations:

1. If `coredd` left a mail file, the file is mailed to `root` (see `coredd`(8)).
2. The `/usr/adm/acct/nite/lock*` files are removed.
3. Maintenance is performed on the super-user log (`sulog`).
4. Maintenance is performed on the `cron` log (see `cron`(8)).
5. Maintenance is performed on the file `/etc/wtmp`.

3.4.13 Start the Security Log Daemon

The UNICOS MLS feature `/etc/rc` script starts the security log daemon.

3.4.14 Start Accounting

If the RC_ACCT parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the accounting startup script is activated.

3.4.15 Start System Activity Data Collection

If the RC_SADC parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the activity data collection startup daemon is activated.

3.4.16 Activate Category SYS1 System Daemons

The sdaemon(8) command is used to start all category SYS1 daemons at this time. SYS1 daemons are those that must be started before network startup.

For more information, see the description of the netstart script in the *UNICOS Configuration Administrator's Guide.*

3.4.17 Activate netstart

If the RC_NET parameter is set to YES, or if it is set to ASK and the operator replies YES to the question, the network startup script is activated. The netstart(8) script initializes all networking activities.

3.4.18 Activate Category SYS2 System Daemons

The sdaemon(8) command is used to start all category SYS2 daemons at this time. SYS2 daemons are those that must be started after network startup.

For more information, see the description of the netstart script in the *UNICOS Configuration Administrator's Guide.*

3.4.19 Create Network Access List

With the UNICOS MLS feature, the secure network access list (NAL) is created in the Installation and Configuration Menu System in the Configure System->Multilevel Security (MLS)->Configuration->MLS Network Security Options->Network Protocol Security Configuration->MLS Network Access List (NAL) Sets menu.

The NAL is defined in the /etc/config/spnet.conf file, along with the workstation access list (WAL) and CIPSO map-domain sets. When the UNICOS operating system is put into multiuser state, the spnet.conf file is read by the /etc/spnet command to load the NAL, WAL and CIPSO map tables into the kernel.

3.4.20 Set MLS Wildcard Files and Directories

With the UNICOS MLS feature, the`/etc/spwcard` program (see `spwcard`(8)) is executed. This program assigns certain directories and files, such as `/dev/null`, a wildcard label, which allows them to contain files at varying security levels.

Note: If your site has already configured the system to support the Cray ML-Safe configuration, using the `spwcard` command does not relabel any file or directory (for example, relabel a multilevel directory (MLD) with a wildcard label).

3.4.21 Execute `/etc/rc.pst`

If the local executable script called `/etc/rc.pst` exists, it is executed at this time. That is, the `/etc/rc.pst` script is executed after all daemons are started, and just before the multiuser startup process is finished.

This file is not supplied by Cray.

Warning: Modification (and/or creation) of this script must comply with the guidelines set forth in the single-user mode descriptions in Chapter 8, page 145.

Additionally, the Cray ML-Safe system configuration is available to normal users at the time that this script is executed. Therefore, additional care must be taken to ensure that the operations performed by this script follow the same rules and restrictions that are enforced for a security administrator, as described in Chapter 8, page 145.

3.4.22 Complete the Multiuser Startup

As its final task, `/etc/rc` writes an entry into `/etc/boot.log`, if that file exists.

File System Maintenance [4]

This chapter includes the following topics for all Cray systems:

- Mounting and unmounting file systems
- File system utilities
- File system backup and restoration
- File system checking and repair with `fsck`(8)

4.1 Mounting and Unmounting File Systems

A disk device is a sequential array of data until it is mounted. When the device is mounted, the UNICOS kernel interprets the data as a UNICOS file system and makes the file system available as part of the system's complete directory structure.

File systems are mounted with the `mount`(8) command. The start-up script `/etc/rc` (see `brc`(8)) traditionally mounts the various file systems that are available during system operation. It may prove more convenient to have `/etc/rc` execute another script or set of scripts (located in the `/etc` directory) to mount the various file systems, allowing system users to use the same scripts to mount the file systems during single-user mode.

The `/etc/fstab` file (see `fstab`(5)) contains descriptions of file systems and swapping partitions. The `mount` command searches this file, if present, to determine the parameters it should use.

File systems are unmounted from the system by using the `/etc/umount` command (see `mount`(8)). When unmounting clustered file systems, you must specify the directory on which the file system is mounted; `umount` does not work if the file system is a cluster and you specify the cluster descriptor file name. You may use the node name to unmount a regular file system.

The `umount`(8) command flushes the file system cache to the disk before actually unmounting the file system.

4.2 File System Utilities

The following utilities are associated with the maintenance of file systems and files (more information about these commands can be found in UNICOS man pages).

Command	Description
`bmap`(8)	Reports the current use of a given block in a file system. Used to determine whether a block that needs to be flawed is currently in use by a file and, if so, the name of the file.
`ddstat`(8)	Displays configuration information about disk type character and block special devices.
`df`(1)	Reports the number of free blocks available for mounted file systems.
`diskusg`(8)	Summarizes the disk usage on a file system by file ownership.
`dmap`(8)	Reports the slices of all physical devices that compose a given logical device or the slices of all logical devices that reside on a given physical device.
`du`(1)	Provides a summary of the disk use on a file system by directory structure.
`fck`(1)	Displays data block allocation for a specific file.
`fsck`(8)	Checks and corrects the consistency of a file system before it is mounted. File system checking should always be a part of the system startup procedures. Use of the `fsck` command is described in detail on Section 4.4.1, page 101.
`fsed`(8)	Debugs file systems.
`fsmap`(8)	Reports all free block areas in a specific file system; useful for determining fragmentation.
`fstest`(8)	Tests file systems and disk devices.
`labelit`(8)	Reads or writes file system labels and security levels.
`ldcache`(8)	Assigns and displays logical device cache.
`mkfs`(8)	Creates a file system on a logical device.

`mknod`(8)	Builds a directory entry and inode for a special file.
`mkspice`(8)	Creates physical disk device inodes that describe the spare sector map, factory flaw map, and diagnostic and customer engineering slices.
`sdconf`(8)	Controls the state of a disk drive.
`sdstat`(8)	Displays information about disk device I/O.
`setf`(1)	Initializes a new or existing file. The file is created if it does not already exist, and a specified number of bytes or blocks are allocated to it.
`setfs`(8)	Modify attributes of a file system after creation.

4.3 File System Backup and Restoration

This section describes some of the procedures you can use to back up files and file systems on Cray systems running UNICOS.

The following sections discuss the three major applications of the file system backup and restoration tools:

- Local backup of file systems (usually to tape), using the `dump`(8) and `restore`(8) commands
- Remote backup of file systems through the network to another server by using the `rdump`(8) and `rrestore`(8) commands

4.3.1 Local Backup

The UNICOS `dump`(8) and `restore`(8) utilities provide the capability to backup and reload file systems. This is usually done using tapes. This section assumes that your Cray computer system has a tape subsystem as the target for the file system dump.

Warning: Use of the `dump`(8) and `restore`(8) utilities on a Trusted UNICOS system requires a multilevel archive medium.

4.3.1.1 Using the dump Command

The `dump` command copies to magnetic tape all files changed after a specified date in a specified file system. Refer to the `dump`(8) man page for a complete list

of the options available for use with dump. Some of the most useful options are described in this section.

The -t *dump_type* option and argument specify the dump level; *dump_type* can be a number from 0 through 9. If the -t option is omitted, dump defaults to a level-9 dump. A level-0 dump is a complete dump; all files in a file system are copied to tape. For a given dump level *x*, only those files modified since the last dump of level- *y* (*y* <= *x*) are dumped.

For example, if you did a level-2 dump on Monday, followed by a level-4 dump on Tuesday, a subsequent level-3 dump on Wednesday would contain all files modified or added since Monday.

Although this arrangement provides significant flexibility in scheduling dumps, a relatively simple scheme is recommended:

- Once a week, perform a level-0 dump of each file system. Use at least two sets of tapes so that you can recover files even if a file system is destroyed while you are dumping it. Because dump opens and reads a raw file system (instead of using the operating system to open and read each file), it is recommended that at least this complete dump be performed on a quiet system, with no users other than the administrator or operator logged in. dump can read an unmounted file system; if you prefer, you can unmount the file system to be dumped before you begin.
- On each day that you do not perform the level-0 dump, perform a level-9 dump of each file system. Use a different set of tapes each day for two weeks to safeguard your data. Each level-9 dump contains the files modified since the last weekly level-0 dump. Thus, to reload a file system, you need only two sets of tapes: the weekly dump, and the latest daily dump.

To simplify the task of performing dumps, you can set up a shell script for your operator as follows:

```
if [ "$1" = "daily" ] ; then
  level=9
elif [ "$1" = "weekly" ] ; then
  level=0
else
  echo "Usage: backup daily|weekly"
  exit 1
fi
for fs in /dev/dsk/root /dev/dsk/usr /dev/dsk/slash_u ; do
  /etc/dump -t level -u $fs
done
```

Using this script, the operator needs to enter only `backup daily` for a daily backup, or `backup weekly` for a weekly backup.

The `-u` option causes `dump` to write the date and time of the beginning of the dump in the `/etc/dumpdates` file, which contains the file system name, the level of dump, and the time and date that the dump started. (Refer to `dump`(5) for the format of the `dumpdates` file.)

The following options can be used with `dump`(8) to specify different tape attributes:

`-l` Specifies labeling of the tapes. The following values can be used with the `-l` option:

`nl`	Nonlabeled tapes
`sl`	IBM standard labeled tapes
`al`	ANSI labeled tapes

`-v` Allows you to enter a list of volume serial numbers (VSNs). If this option is omitted, you are asked to type in a VSN list; the VSNs in the list are separated by colons (:). Each VSN is a string consisting of 1 to 6 characters. In the following example, `dump` would use the volumes `r1`, `r2`, and `r3`:

```
/etc/dump -t 0 -u -v r1:r2:r3 /dev/dsk/root
```

`-D` Allows you to request a specific tape device for the dump, as in the following example:

```
/etc/dump -t 2 -u -D tape00 /dev/dsk/usr
```

Refer to the `tpmnt`(1) command for a more complete description of these tape-specific options. The `dump` command actually performs an `rsv`(1) and `tpmnt`(1) from these specifications.

You may want to perform your own `rsv` and `tpmnt` commands before the dump, as in the following example:

```
rsv TAPE 1
tpmnt -l sl -P /tmp/name -v r1:r2:r3
/etc/dump -t0 -u -f /tmp/name /dev/dsk/root
rls -a
```

You may also use a disk file, rather than tape, for some special purpose, by specifying the `-f` option, as in the following example:

```
/etc/dump -t 9 -f /tmp/dumpfile /dev/dsk/usr
```

Refer to the dump(8) man page for a complete list of the options available for use with the dump command.

Warning: Use of the dump(8) and restore(8) utilities on a Trusted UNICOS system requires a multilevel archive medium.

4.3.1.2 Using the restore Command

You can reload a file system from the dump tapes by using the restore(8) command. The restore command accepts various options. Refer to the restore(8) man page for a complete list of the options available for use with restore. Some of the most useful options are described in this section.

To reload a file system from dump tapes, first use the mkfs(8) command to initialize the unmounted file system. Mount the file system and change directories (using cd(1)) to the mount point. Next, use restore with the -r option to reload the full (level-0) dump; use restore with the -r option again to reload the incremental (level-9) dump.

The restore command requires many synchronous write operations, which can be time-consuming. You can disable synchronous write operations and increase the efficiency of the restore command by using the ldcache(8) facility.

The following examples show how you would perform a full restore operation if you had a file system that had been destroyed. First, you would initialize the unmounted file system, /newfs; then you would mount the file system, change directories to the mount point, and reload the full dump, as follows:

```
/etc/mkfs /dev/dsk/newfs
/etc/mount /dev/dsk/newfs /newfs
cd /newfs
/etc/restore -r -V fs1:fs2:fs3
```

The -V option specifies a volume serial number (VSN) list of dump tapes to be used for the restore operation. In this example, the tapes with VSNs of fs1, fs2, and fs3 are used.

After this completes, you would restore the last incremental dump, as follows:

```
/etc/restore -r -V fsd1
```

Now, you would remove the restoresymtab file (this file is created to pass along information between the restore of the complete dump and incremental dump) by using the following command:

```
rm restoresymtab
```

The new file system would have all of the files in it up to the last incremental dump.

If you want to reload a particular set of files from a dump tape, use the following invocation of `restore`:

`/etc/restore -x` *filenames*

The `-x` option causes `restore` to extract named files from the tapes. The file names are relative to the mount point of the file system.

For example, if you had a file system `/dev/dsk/usr_mail` mounted on `/usr/mail`, you would dump it by using the following command:

```
/etc/dump -t 0 -u /dev/dsk/usr_mail
```

To reload `/usr/mail/fred` and `/usr/mail/root` from this dump, you would first change directories to `/usr/mail`, and then use `restore` with the `-x` option to reload those particular files, as follows:

```
cd /usr/mail
/etc/restore -x fred root
```

The `restore` command can also be used in interactive mode. After reading in the directory information, `restore` provides a shell-like interface that allows you to move around the directory tree, selecting files for extraction. You can use the commands `ls`(1), `cd`(1), and `pwd`(1) as they are used in the shell and add or delete files as you wish. This is a convenient way to examine the contents of a dump tape and restore one or more single files or directories.

The `restore` command also has a `-t` option that writes out the table of contents of the dump tape to standard output.

As with `dump`, you can perform your own `rsv`(1) and `tpmnt`(1) first, using the `-f` option, as in the following example:

```
rsv CART
tpmnt -l sl -v r1:r2:r3 -g CART -P /tmp/tape
/etc/restore -if /tmp/tape
```

Select files to extract and quit the interactive mode

```
rls -a
```

The `-F` option of the `restore` command specifies the tape file ID of the dump tape to be restored. The default ID is the volume serial number (VSN) of the first tape of the dump file.

You must follow some special procedures if your /usr or / (root) file system is destroyed, because you need certain files and directories to use the restore command and the online tape daemon. The default installed tape daemon uses the /usr/spool/tape directory, and the message daemon uses the /usr/spool/msg and /usr/adm/msg directories. If the tape daemon or message daemon have not been installed by default, other directories may be used. Directories used by the daemons must exist so that the tape and message daemons can add files to them before you restore the /usr file system.

When restoring / (the root file system), you also need the tape daemon and message daemon. If a lack of disk space prevents you from keeping a spare root file system, you must keep the following binary files on the operator's workstation on Cray PVP systems.

```
/etc/msgdstop
/etc/tpapm
/etc/tpbmx
/etc/tpclr
/etc/tpdev
/etc/tpconfig
/etc/tpdstop
/etc/tpfrls
/etc/tpgstat
/etc/tplabel
/etc/tpmls
/etc/tpmql
/etc/tpset
/etc/tpu
/usr/lib/tp/avrproc
/usr/lib/tp/clsfile
/usr/lib/tp/fesreq
/usr/lib/tp/flush
/usr/lib/tp/openfile
/usr/lib/tp/opentdt
/usr/lib/tp/proceot
/usr/lib/tp/proceov
/usr/lib/tp/readerr
/usr/lib/tp/readvol
/usr/lib/tp/slnet
/usr/lib/tp/stknet
/usr/lib/tp/tpdaemon
/usr/lib/tp/tppos
/usr/lib/tp/writeerr
/usr/lib/tp/writevol
```

```
/usr/lib/msg/infd
/usr/lib/msg/msgd
/usr/lib/msg/msgdaemon
/usr/lib/msg/oper
/usr/lib/msg/rep

/bin/rls
/bin/rsv
/bin/tpcatalog
/bin/tplist
/bin/tpmnt
/bin/tprst
/bin/tpscr
/bin/tpstat
```

The following character special file for tapes is also needed for restoring directories and files, and it should be kept on the operator's workstation on Cray PVP systems.

```
/dev/bmxdem
```

Any other files you need for deadstarting the system should also be kept on the boot media.

You will need the following files to run `restore` when the system is running:

```
/usr/lib/msg/msgdaemon
/usr/lib/tp/tpdaemon
```

Copy these files into your new root file system, and then start the tape and message daemons. Now you should be ready to proceed with the restoration of your root dump tapes.

If you are running an autoloader, you must have a full system, which includes all the basic files in the major directories of your system, in order to run the `restore` command successfully. Therefore, the procedure outlined in this section will not provide you with everything you need.

4.3.2 Remote Backup

This section describes how to perform file system dumps and restorations on machines that do not have Cray online tapes but are part of a TCP/IP network.

Warning: This feature is not supported on a Cray ML-Safe configuration of the UNICOS operating system.

The rdump(8) and rrestore(8) commands are used to perform file system backups to a tape device on a remote host. These commands provide an I/O interface between the dump and restore commands on the Cray mainframe and the device on the remote host. The commands create a remote server process running the /etc/rmt command on the client machine. This processes accesses the tape device.

You must log in as root to perform the backup and restore procedures. Because the file transfer is performed across the network, you must ensure that the user root on the remote host has the Cray mainframe listed as a target machine in the .rhosts file.

For example, if you (as root) want to dump a file system from the Cray mainframe named sn1203 to the remote server hall, be sure that a .rhosts file exists in the root directory on hall and that it contains an entry for sn1203. For more information about .rhosts files, see the rhosts(5) man page.

The following examples illustrate the use of rdump and rrestore.

Example 1: The following command performs a level 0 dump of file system fs1 to the tape device rst0 on the host host1:

```
rdump -f host1:/dev/rst0 - -t 0 /dev/dsk/fs1
```

Example 2: The dump performed in example 1 can be restored to a file system filesys2 on the Cray system with the following command:

```
rrestore -f host1:/dev/rst0 - -x
```

Refer to the rdump(8), rrestore(8), dump(8), and restore(8) man pages for complete descriptions of these commands and their options. The fsck(8) command is an indispensable tool that maintains the consistency of UNICOS file systems by interactively repairing most file system damage resulting from an operating system crash. fsck reports its progress through a series of phases, checking a file system for consistency and repairing any inconsistencies it discovers. If fsck determines that a file system has no inconsistencies, or that it has had its inconsistencies repaired, you can safely mount the file system. You should also use fsck to ensure that file systems are not damaged before going into multiuser mode or doing backups.

4.4 File System Checking and Repair with fsck

The following section provides an overview of file system operation, and how using the fsck(8) command can help ensure date integrity. The subsequent

sections describe the behavior of `fsck` and the phases that `fsck` goes through while checking a file system.

4.4.1 Overview of File System Operation

The file system directories consist of pointers to inodes. In turn, these inodes point to blocks of pointers to data and directories. Unfortunately, the operating system cannot perform extensive validation of file system integrity, and when this elaborate construction loses its consistency, there can be a serious loss of data. With careful maintenance, however, you can ensure that the file system works safely and efficiently.

Damage to the file system occurs when a portion of its structure is lost before it can be written to disk. This damage is typically caused by a hardware, operational, or operating system failure. While a file system is in use, it consists of a combination of data on disk and, for efficiency, data in kernel memory.

The memory-resident data is written at regular intervals by the `init`(8) process with the `sync`(2) system call. The cache can be written to disk at any time by using the `sync`(1) command. On Cray PVP systems that use logical device caching, the logical device cache can be written to disk at any time by using the `ldsync`(8) command. (`ldsync` should always be issued after the `sync` command is issued.)

The `inittab`(5) file allows you to control the rate of the `sync` and `ldsync` operations with the `sleeptime` and `ldsynctm` entries. More frequent execution of the `ldsync` command reduces the risk of file system corruption in the event of a system crash, but the increased system overhead may impact system performance. The default rate of `ldsync` is 120 seconds, and the default rate of `sync` is 30 seconds.

If the operating system is stopped before all the data is returned to disk, the structure of the file system may be damaged. Usually, this damage is corrected by `fsck` at restart time. If the file system is used in an inconsistent state, however, the damage quickly spreads throughout the system and destroys it.

The rules for correct use of the file system are as follows:

- Before using a file system, you must ensure that it is consistent
- When you stop the system, each file system must be consistent

The `fsck`(8) command checks and corrects the consistency of a file system before it is mounted. File system checking should always be part of the system startup procedures.

4.4.2 Using fsck

When using the fsck(8) command, usually you will respond yes to all of the prompts. However, in the event of a system crash, the damage may be extensive enough to warrant recovery from a back-up tape. If the file system is consistent, fsck prints a summary of statistics about the file system.

Note: Many corrective actions may result in some data loss.

You can use the -y option on the fsck command to avoid the questions asked by fsck concerning inconsistencies it found. This option automatically attempts repairs as though you had answered yes to the questions. Use this with caution, however, because the corrections may involve some data loss.

The following examples show the results of the use of fsck, first, without the -y option, and then with the -y option.

```
$ /etc/fsck tmp2fs
tmp2fs: device tmp2fs opened as partition 0
tmp2fs: superblock fname , fpack
tmp2fs: Phase 1 - Check Blocks and Sizes
tmp2fs: Phase 2 - Visit Directories
tmp2fs: Phase 3 - Checking Directories
tmp2fs: Phase 4 - Checking Non-Directories and Link Counts
tmp2fs:
tmp2fs: i-node 0.0000002 has problems
tmp2fs: i-node summary
tmp2fs:         owner 0, mode 100644, link 1
tmp2fs:         size 13312, mtime Wed Dec 31 18:00:00 1989
tmp2fs: paths to this i-node ...
tmp2fs:          - ./afile
tmp2fs:         out-of-range allocations
tmp2fs:         (warning) file size field in i-node is incorrect
tmp2fs: clear ('y' or 'n')? y
tmp2fs: Phase 5 - Verify Dynamic Information
tmp2fs: block 0/71 missing from free block list
tmp2fs: block 0/72 missing from free block list
tmp2fs: block 0/73 missing from free block list
tmp2fs: partition 0, free i-node count is 62, should be 63
tmp2fs: rebuild dynamic information ('y' or 'n')? y
tmp2fs: Phase 6 - Rebuilding Dynamic Information
tmp2fs: file system summary
tmp2fs:     partition 0 on device tmp2fs
tmp2fs:         64 total i-nodes (63 free i-nodes)
```

```
tmp2fs:         30 total tracks (25 free tracks, 8 free blocks)
tmp2fs: ***** FILE SYSTEM WAS MODIFIED *****

$ /etc/fsck -y tmp2fs
tmp2fs: device tmp2fs opened as partition 0
tmp2fs: superblock fname , fpack
tmp2fs: Phase 1 - Check Blocks and Sizes
tmp2fs: Phase 2 - Visit Directories
tmp2fs: Phase 3 - Checking Directories
tmp2fs: Phase 4 - Checking Non-Directories and Link Counts
tmp2fs:
tmp2fs: i-node 0.0000002 has problems
tmp2fs: i-node summary
tmp2fs:         owner 0, mode 100644, link 1
tmp2fs:         size 13312, mtime Wed Dec 31 18:00:00 1989
tmp2fs: paths to this i-node ...
tmp2fs:          - ./afile
tmp2fs:         out-of-range allocations
tmp2fs:         (warning) file size field in i-node is incorrect
tmp2fs: clear? yes
tmp2fs: Phase 5 - Verify Dynamic Information
tmp2fs: block 0/71 missing from free block list
tmp2fs: block 0/72 missing from free block list
tmp2fs: block 0/73 missing from free block list
tmp2fs: partition 0, free i-node count is 62, should be 63
tmp2fs: rebuild dynamic information? yes
tmp2fs: Phase 6 - Rebuilding Dynamic Information
tmp2fs: file system summary
tmp2fs:     partition 0 on device tmp2fs
tmp2fs:         64 total i-nodes (63 free i-nodes)
tmp2fs:         30 total tracks (25 free tracks, 8 free blocks)
tmp2fs: ***** FILE SYSTEM WAS MODIFIED *****
```

All mountable file systems should be listed in the `fstab`(5) file, which the `mfsck`(8) command uses, and you should check these file systems each time the system is rebooted.

The `fsck` command cannot be executed on a mounted file system, because this would repair only the physical disk, leaving all the system buffers incorrect. The only exception to this is the `root` file system, which is always mounted and must be repaired while mounted.

To repair the `root` file system, enter the following command:

```
fsck /dev/dsk/root
```

`fsck` may respond by asking questions. You can use `mfsck` to run file system checks in parallel, which can speed system startup. If `fsck` prompts for a response to any problem, an analyst experienced in repairing UNICOS file systems should assist you. You may reply `n` to any `fsck` prompt, leaving the indicated inconsistency unresolved.

Under no circumstances should you mount or boot from any file system that still has unresolved inconsistencies detected by `fsck`.

When you are using `fsck` to repair damage following a crash, it is useful to first use the `-n` option with `fsck` to survey the damage (the `-n` option assumes an automatic `no` response to all questions). Having seen the extent of the damage and determined that there are no extraordinary inconsistencies, you may use the `-y` option, in conjunction with other `fsck` options, to avoid having to type `y` in response to each question.

The `fsck` command provides orphan checking, which allows the possible recovery of files that have no links to the file system directory structure. (Such a file may occur when a directory entry for the file is not written to the disk prior to a system crash.)

Note that the set-user-ID and set-group-ID bits are cleared on all orphans recovered by `fsck`.

4.4.3 `fsck` Phases

The `fsck`(8) command goes through nine phases, described in the following sections. Unless otherwise specified, the phases are the same for all native Cray file systems.

4.4.3.1 Initialization Phase

During the initialization phase, `fsck` verifies that all opened devices are partitions of the same file system. Most problems cause the program to stop. The following checks are performed:

- All superblocks and dynamic blocks can be located
- Inode blocks, bad blocks, super blocks, and dynamic blocks can be allocated without error

- Track size and root inode number are consistent across all partitions in the file system
- Total inode count is consistent with the number of blocks allocated for inodes
- Track size and total size are consistent with the value returned by a `stat`(2) system call

4.4.3.2 Phase 1

During phase 1, `fsck` examines each active inode. Errors detected during this phase will be announced during phase 3 or 4 when a file name is associated with the inode.

The following checks are performed:

- The mode field is valid
- The sectors belonging to the file may be allocated without conflict and within valid areas of the file system
- The last byte of the file is contained in an allocated sector

4.4.3.3 Phase 2

If the `-f` option is not specified, `fsck` enters phase 2 and examines the contents of all directory sectors, noting garbled sectors and inodes that are not valid. Links from the directory to inodes are saved. The `fsck` command travels through the directories by order of the address of the first disk block, rather than in directory tree search order. By doing so, the execution speed is faster, but a phase 3 subroutine is required to gather a file name for a problem report.

For each entry in a directory, `fsck` verifies that a nonzero inode field refers to an accessible inode.

The directory tree structure is validated. Each directory should be accessible along only one path, and it should contain valid `.` (dot) and `..` (dot dot) entries.

4.4.3.4 Phase 2X

If the `-f` option is specified, `fsck` skips phases 2 through 4 and prints the errors discovered during phase 1, but offers no opportunity to clear the inode.

4.4.3.5 Phase 3

The `fsck` command reports errors in directory inodes. The names of unlinked directories are displayed to the operator to be selected for relinking and directories with garbled sectors are displayed to the operator to be selected for clearing.

4.4.3.6 Phase 4

All nondirectory inodes with problems are reported in phase 4.

4.4.3.7 Phase 5

During phase 5, `fsck` examines the free track, free sector, and free inode information in the dynamic block. Any errors detected are announced. The operator is offered an opportunity to rebuild the dynamic block.

4.4.3.8 Phase 6

During phase 6, `fsck` rebuilds all information in the dynamic block, except the magic word.

4.4.3.9 Termination Phase

A summary of the state of the file system is printed.

Basic Administration [5]

This chapter describes the following tools and methods, many of which are commonly used in the day-to-day operation of a UNICOS system:

- Using the `cron` and `at` utilities
- The temporary directory (`TMPDIR`)
- Communicating with users
- Monitoring system security
- Job and process recovery
- Kernel user exit (`uesyscall`)

Warning: This chapter contains warnings and information critical to the proper use of a Cray ML-Safe configuration of a UNICOS system.

5.1 Using the `cron` and `at` Utilities

The `cron`(8) and `at`(1) utilities are invaluable tools for automating many administrative tasks. You can use them to run administrative tasks at regular intervals and during off-peak hours, when they will not interfere with the interactive work of most users. Neither utility is appropriate for every administrative task; by using both, however, administrators can avoid many time-consuming and repetitive tasks.

Note: For information on using the `cron` and `at` utilities on a Cray ML-Safe system configuration, see Section 8.4.4, page 199.

5.1.1 Administrative Use of `cron`

The `cron`(8) process executes commands at specified dates and times. As a system administrator, you can use the `cron` process to run tasks on a periodic basis. The `cron` utility is especially convenient for the following administrative tasks:

- Turning off the programs in certain directories during prime time (by using the `chmod`(1) utility to remove execute permission for the program).
- Running programs during hours other than prime time for the following procedures:

- File system administration
- Accounting
- System security procedures

You can specify the commands to be executed by using the `crontab`(1) utility, which takes as its argument a `crontab` file that describes the commands and the times at which they should be run. The `cron` process consults the files located in the directory `/usr/spool/cron/crontabs` to determine which tasks are to be performed and at what times they are to be performed.

Each user maintains only one individual `crontab` file, which is the user's ID, and which requires that the `/usr/spool/cron/crontabs` directory contain separate files for each user. The name of the `crontab` file is used as a user ID to get user and group permissions.

A `crontab` file consists of lines containing six fields each. The fields are separated by spaces or tabs. The first five fields are integers that specify the following:

- Minute (`0` to `59`)
- Hour (`0` to `23`)
- Day of the month (`1` to `31`)
- Month of the year (`1` to `12`)
- Day of the week (`0` to `6`, with `0` = Sunday)

Any of the first five fields can be an asterisk (*), indicating that any value is appropriate. The last field is the command line to be executed at the appropriate time. For example, the following line in a `crontab` file would execute a local program called `/usr/bin/task` once a week, every Sunday morning at 6:30 A.M.:

```
30 6 * * 0 /usr/bin/task
```

If the system is not running at the time a command is to be executed by `cron`, the command does not execute. Consequently, `cron` is most appropriate for periodic tasks that do not interfere with normal system operation if not executed, and for tasks that must be run at a specific, regular time.

The `cron`(8) process is usually started from the file `/etc/rc` during system startup. See `crontab`(1) and `ksh`(1) for more detailed information.

The cron daemon makes an attempt to report fatal errors that cause termination by printing an error message on the system console. Also, a user of the at or crontab utility receives a warning message if the cron daemon is not active at the time the at or crontab command is issued.

The cron daemon can limit dynamically the number of concurrently running jobs. It can also maintain up to 26 separate queues, and control the number of jobs executed in each queue. The file /usr/lib/cron/queuedefs is used to maintain definitions for all queues. If this file does not exist, the default values are used. See queuedefs(5) for more detailed information.

Changes to queue definitions take effect before the next job is executed by the cron daemon.

The cron daemon logs all command invocations, terminations, and status information in the file /usr/lib/cron/log. Records that begin with the character > pertain to command invocations. Two invocation records are written for each command execution: the first displays the command being executed; the second contains the login name of the user who executed the command and the process ID, job queue, and time stamp for the command. Command termination records begin with the character < and are similar to the second invocation record, except that a nonzero termination status or exit status is also printed. Records that begin with the character ! indicate status information.

The cron utility uses named pipes to communicate between the user-level commands and the daemon process.

5.1.2 Administrative Use of at

The at(1) utility submits commands or shell scripts for execution at a specific time. Its format is as follows:

at *time* [*date*]

It takes as its argument the time (and optionally, the date) at which to execute a list of commands that it reads from the standard input. Other options and arguments are available (see the at(1) man page).

The at utility is intended primarily for single, nonrepetitive execution of a command or script, and is thus especially appropriate for scheduling large jobs to run during off-peak hours. However, the at utility can set up periodic execution of a task if a script being run by at uses at itself to reschedule its own execution. For example, if a script, whose file name is /usr/bin/task, contains a line such as the following, it reschedules itself to be run at 3:30 the

next morning (and the morning after that, and so on, because the script still contains the rescheduling line):

```
echo "sh /usr/bin/task" | at 0330 tomorrow
```

The advantage to using `at` instead of `crontab`(1) for periodic command execution is that you are assured that the command will run; if the system is down at the time `at` would normally run the command, it runs as soon as the system is brought up again. The drawback to this automatic execution is that the command is not guaranteed to run at the specific time you request. That is, if a command is submitted through `at` to be executed at 3:30, and the system is down for dedicated time until 7:30, the command will run at 7:30 when the system is running, which may interfere with users' work if it is a large, CPU-intensive command.

The prototype file allows you to customize `at` command files by controlling what information is written into the `at` job file, `/usr/spool/cron/atjobs`. If a file named `/usr/lib/cron/.proto.`*q* exists (*q* is a queue name), this file is copied into the job file. Otherwise, the `/usr/lib/cron/.proto` file is used.

The following substitutions are made during creation of an `at` job file:

Variable	Description
`$a`	User's current account name
`$m`	User's current file creation mask (see `umask`(2))
`$l`	User's current file size limit (see `ulimit`(2))
`$d`	Name of the current directory
`$t`	Time (in seconds since 1/1/70) when the job is scheduled to execute
`$<`	Read standard input until EOF is reached

The following is an example of a prototype file:

```
newacct $a
cd $d
ulimit $l
umask $m
$<
```

At minimum, a prototype file containing `$<` must exist to successfully run the `at` utility. The `at` utility exits with an error if no prototype file exists.

The `at` utility can queue jobs in one of 25 different queues, with the `cron` daemon controlling the number of executions for each queue. (You can use this

queuing mechanism to limit the use of the crontab and at utilities.) Running the at utility with the -qx option as the first argument queues the command in queue x. The default queue is a. A special queue, b, is defined as a batch queue; jobs in this queue run whenever the defined maximum level is not exceeded (as specified in the queuedefs file). Queues d through z, by default, run at the same priority as b. Queue c (available with the standard AT&T at utility to run cron executions) is not available with the UNICOS at utility; the crontab(1) utility should be used to submit crontab jobs. Jobs in all other queues run at the time specified on the command line.

5.1.3 Restricting Use of crontab and at Utilities

Users can potentially abuse system resources when using the crontab(1) and at(1) utilities. However, both the crontab and at utilities provide methods for restricting user access. The /usr/lib/cron/cron.allow and /usr/lib/cron/at.allow files contain the login names of users (one per line) allowed access to the crontab and at utilities, while the /usr/lib/cron/cron.deny and /usr/lib/cron/at.deny files contain login names of users denied access to the utilities.

When a user submits a crontab file, crontab checks cron.allow for a list of users permitted to have a crontab file. If no cron.allow file exists, the file cron.deny is scanned for users who are denied crontab files. If neither file exists, only root is allowed to have a crontab file. The same process is used for determining access to the at utility. The null cron.allow file would mean no user is allowed a crontab file, while a null cron.deny file would mean that no user is denied a crontab file.

For additional information on the at.allow, at.deny, cron.deny, and cron.allow files, see the *UNICOS Configuration Administrator's Guide.*

5.2 The Temporary Directory (TMPDIR)

The TMPDIR directory contains temporary user subdirectories and files. TMPDIR is created at the beginning of an interactive session or batch job. All the files and directories in TMPDIR are deleted at the completion of the session or job. UNICOS commands and libraries create temporary files in TMPDIR instead of /tmp or /usr/tmp.

UNICOS temporary directories are owned by the user and have group and other permissions turned off. This prevents other users from seeing or deleting files in a temporary directory they do not own.

5.3 Communicating with Users

During the operation of a UNICOS system, it is frequently necessary for administrators to use the system to communicate information to its users. This section discusses a number of UNICOS commands and tools that enable you to communicate with users:

- The `wall`(8) command
- The `/etc/motd` file
- The `/etc/issue` file
- The `/usr/news` directory
- The `write`(1) utility
- The `mail`(1) utility

5.3.1 The `wall`(8) Command

The `wall`(8) command broadcasts items of immediate concern to all users currently logged in to the system. Run the command by typing the following:

```
/etc/wall
```

The `wall` command responds by telling you to type your message and to press `CONTROL-d` when you are finished. To ensure that all users who are currently logged in see a message sent by `wall`, run the command while you have `root` privileges; otherwise, the message goes only to users who allow messages to be written to their terminals (see `mesg`(1)). Additionally, users who are not currently logged in will never see the message; `wall` is thus not a suitable method for communicating a message to all users who have accounts on the system.

The `wall` command is typically used to send the following messages:

- Warnings that the system will soon be brought down for scheduled downtime. Users who log in after the message is sent, however, miss the message and should be notified by the `/etc/issue` file (see `login`(1)).
- Warnings that the system must be brought down immediately to address a system emergency.
- Warnings that a particular file system has run out of disk space and that users should make an immediate effort to delete any unneeded files (see the description of the `-g` option on the `wall`(8) man page).

5.3.2 The /etc/motd File

The /etc/motd (message-of-the-day) file is displayed to users after they are logged in to the system. The /etc/motd file is an ordinary text file, and the administrator may place messages in it by using any UNICOS text editor.

Messages that should be placed in /etc/motd are those that are less immediate than those requiring the use of wall(8), but they are important enough that users should be forced to see them. The administrator should remove messages from /etc/motd as soon as they are no longer needed. Suitable items for inclusion in this file include the following:

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

5.3.3 The /etc/issue File

The /etc/issue file is displayed while a user is logging in, before the user has successfully logged in to the system. It is an ordinary text file, and you may place messages in it by using any UNICOS text editor.

Messages placed in /etc/issue should be brief and so important that users may need the information to decide whether or not to log in to the system. Possible messages include the following:

- Warnings that the system will be brought down soon (so that users who do not see a wall(8) message are not surprised when the system is brought down shortly after they log in)
- Warnings that the system is being used for dedicated time and that not all users will be able to log in

5.3.4 The /usr/news Directory

When users log in to the system they are alerted to the existence of any new files placed in the /usr/news directory. When a user then runs the news(1) utility, it displays any news files that have been created or modified since the last time the user ran news. The files placed in /usr/news are ordinary text files created with any UNICOS text editor, and they are usually assigned names that give a general

idea as to their contents. For instance, a news file containing information about a modification to a system library might be given the name `new.library`.

Because users are not notified of the existence of a new news file until the next time they log in, and because there is no guarantee that any given user will see the file (a user may choose to ignore the item by not running the `news` utility), `/usr/news` is appropriate for items that are not time-sensitive or items that are of interest to only some of the system's users. These categories include the following:

- Notices regarding recent system changes, such as a newly installed version of a command or library
- Explanations of imminent system reconfigurations or changes
- Explanations of recent system problems and their possible effects on users

It is a good idea to remove any old files in `/usr/news` periodically, not only to save disk space, but also to prevent new users on the system from having to read through a long list of out-of-date news items. The `/usr/news` file may be cleaned out regularly by `cron`(8).

5.3.5 The `write`(1) Utility

The `write`(1) utility initiates immediate person-to-person communication with a logged-in user by opening that user's tty or pty for writing and copying each line of text you type to his or her screen. To write to a user with a login name of `dolores`, for example, you would issue the following command:

```
write dolores
```

If the user `dolores` happened to be logged in on more than one tty or pty, you could specify the connection:

```
write dolores ttyp001
```

If, in this example, the user `dolores` is currently logged in, a message appears on her screen indicating that you are writing to her. Typically, the user `dolores` replies by writing back to your account; each line of text she types appears on your screen.

Given the immediate nature of its communication, the `write` utility allows you to perform the following functions:

- Converse with a user
- Obtain information about what a user is doing

- Warn a specific user to stop what he or she is doing
- Instruct a specific user to clean up his or her directories

Because each typed line appears on the other user's terminal without regard for what that person may be typing at the moment, it is easy for the other user's messages to your terminal to appear to interfere with your typing. This problem is customarily solved by having the two users take turns typing, ending a message with an `o` on a line by itself (standing for "over," much as in a two-way radio conversation). To end such a session, either user then ends a message with an `oo` on a line by itself (for "over and out"). Thus, a typical "conversation" carried out by `write` might look like this (your input appears in **bold**):

```
# write dolores
Message from dolores (ttyp001) - Mon May 11 08:20:15 - ...
Yes
o
Please clean up your account, we're out of space.
o
All right, I will.
o
Thank you.
oo
<EOT>
```

Because many users either do not know of this etiquette when using `write`, or do not follow it, they think that `write` is difficult to use. In practice, it is used rather sparingly, mainly when more convenient forms of communication (such as simply calling the user on the telephone) are impossible. Taking steps to educate your user community in the proper use of the `write` utility will prove valuable when `write` is the appropriate communication method.

Note: On a UNICOS system or Cray ML-Safe configuration, for `write` to execute properly, the user's active security labels must be equal.

5.3.6 The `mail`(1) Utility

The `mail`(1) utility provides a way to leave messages for specific users, whether or not they are currently logged in to the system. The `mail` utility is used as follows:

```
mail ralph
```

Type in message

```
CONTROL-d
```

You may specify more than one account name, in which case copies of the message go to each user named. The next time users to whom you (or anyone else) have sent mail messages log in to the system, the system alerts them to the fact that they have mail messages waiting. The `mail` utility is thus particularly well suited for messages such as the following:

- Instructions to clean up directories
- Asking or responding to questions
- General communication

In theory, there is no guarantee that the recipient of a mail message will actually see the message, because the recipient may choose not to run the `mail` utility to read the message; however, in practice, most users read their mail when they log in.

> **Note:** On a Cray ML-Safe configuration, the recipient of a mail message might not be authorized to read mail at the classification with which it was sent.

For more information see `mail`(1) and `mailx`(1).

5.4 Monitoring System Security

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

5.4.1 Super-user Privileges

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

5.4.1.1 Password Security for Super User

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

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

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

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

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

5.4.1.2 Physical Security

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

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

5.4.1.3 Setuid Programs

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

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

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

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

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

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

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

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

5.4.1.4 root PATH

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

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

The root PATH is set in two separate places:

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

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

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

For example, suppose a knowledgeable user creates a program that mimics a commonly used system utility, such as ls(1). In addition to performing the expected system function (listing the files in the current directory), the new ls utility makes a copy of a program such as ksh(1) and turns on the setuid bit on the copy. An unsuspecting super user with the current directory in PATH, having changed directories to a user's directory and inadvertently run the bogus ls, then creates a setuid 0 shell, which gives anyone executing it complete control over the system.

5.4.2 User Security

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

5.4.2.1 The umask Utility

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

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

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

5.4.2.2 Default PATH Variable

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

The users expect to be able to execute programs in the current directory without having to precede the program name with ./ to explicitly indicate the current directory. However, many UNICOS systems traditionally place the current directory first in the PATH, which can make the users vulnerable to a security breach, as described in Section 5.4.2.4, page 121. The current directory should thus be the last entry in the default PATH, after the normal system directories.

5.4.2.3 User Groups

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

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

Having many groups, each containing a small number of users, is safer than having fewer groups, each with large numbers of users with access to each other's files. Members of most logical groups (for example, members of a software development project) 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 with 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 may also choose to create a separate group for each individual user at the time you create the account, and then add users to additional logical groups as necessary.

5.4.2.4 File-owner Fraud

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

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

5.4.2.5 Login Attempts

Unauthorized users might attempt to gain access to the system by making repeated attempts to login. To help prevent such attempts, you can configure the

number of bad login attempts that will be allowed before the `login` terminates. By default, the system will allow an unlimited number of bad login attempts. To put a limit on such attempts, edit the `/etc/config/confval` file (see `login`(1)).

Note: For information on limiting login attempts on a UNICOS system or a Cray ML-Safe system configuration, see Chapter 8, page 145.

5.4.3 Partition Security

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

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

5.5 Job and Process Recovery

This section describes the recoverability considerations and restrictions of the UNICOS operating system.

5.5.1 Restrictions to Job and Process Recovery

This section lists restrictions to recovering jobs and processes submitted either interactively or as batch jobs via the Network Queuing Environment (NQE) and Network Queuing System (NQS). The sections that follow list restrictions common to batch and interactive recovery, and restrictions unique to batch recovery.

5.5.1.1 Restrictions Common to Batch and Interactive

The following list describes restrictions common to batch and interactive job and process recovery:

- All the files that a process was using when it was checkpointed must be present when the process is restarted. This includes all open files, the present working directory, and any shared-text binaries (such as shells) in use by

the process. If any of these is not available when the process is restarted, the `restart`(2) system call fails and returns an `EFILERM` error (errno 51). In the restart file, each of these files is identified by the file system minor number and inode number. If either number changes, for example, if a file system is restored after a process is checkpointed, the `restart`(2)) system call fails with an `EFILERM` error.

The requirement for shared-text binaries to be present can cause restart failures if a system is booted on an alternate root file system, for example, when a system is upgraded from one UNICOS update release to another. If the old root file system is not available when checkpointed jobs are restarted, the restarts will fail with an `EFILERM` error, because the shells are not available. When converting from one root file system to another, the old root should be mounted on some alternate mount point (`/mnt` for example) so that checkpointed processes can be recovered.

- If the `RESTART_FORCE` option is not specified on the `restart`(1) invocation, any file that was in use at checkpoint time must not have been modified since that restart file was created in a nonsequential fashion. That is, the `restart` will fail if any bytes in the file between offset 0 and the file size have been modified since the checkpoint occurred. This rule allows the job or process to be checkpointed and to continue execution, sequentially extending output files, without invalidating the restart file.

  If a restart file includes files that were being written to in a nonsequential mode, it will probably abort or produce incorrect results unless the programmer of the user application has programmed properly for this type of operation.

- The access permissions of the files and directories in use at checkpoint time must not have been changed from their original values, or changed to deny the required access at restart time.

- User and group ownership of the files must be unchanged.

- The sum of the sizes of all unlinked files in use by the target process set must be less than the system limit specified as `MAX_UNLINKED_BYTES` in `config.h`. Note that the intent is to checkpoint and restart processes and jobs using unlinked (sometimes called *zerolink*) files, as long as the files in question are small. This covers such cases as unlinked files created by `/bin/sh` in order to handle here documents, and small temporary files typically used by compilers. Applications that use very large temporary files (for example, `assign -t`) should be changed to use files in the linked temporary directory.

- If a user attempts to copy a restart file or to move a restart file to a different file system, it is no longer marked as a restart file and is not restartable.
- An open pipe connection that originates, or terminates, outside the checkpoint target set prevents the successful completion of the checkpoint.
- Any form of TCP/IP socket usage prevents the successful completion of the checkpoint. Note that the innocent use of certain TCP/IP system calls, such as `gethostname`(2), can cause the TCP special device `/dev/net` to be opened, and this causes a checkpoint request to fail.
- At checkpoint time, there must be sufficient disk space to contain the restart file. Note that the restart file contains at least two sectors worth of header information, any pipe data in transit between processes, the contents of any unlinked files in use, and the user control structures and program image for each process in the checkpoint target set.

  On systems with SSDs, any secondary data segment (SDS) regions in use are appended to the data region of the associated process.
- There must be sufficient system resources, such as inode buffers, file table entries, and job table slots.
- There must be sufficient process slots available so that the successful execution of `restart` will not exceed either the system or user maximum number of processes allowable.
- None of the process, process group, and job IDs needed for the successful execution of `restart` may be in use at restart time.
- Processes using shared memory segments (Cray T90 series systems only) cannot be successfully restarted.
- For a UNICOS system or Cray ML-Safe system configuration, the user must be the owner and have MAC write access or be an authorized user or privileged process.

5.5.1.2 Recovery Restrictions Unique to Batch

The following list describes job and process recovery restrictions unique to batch jobs submitted through NQS:

- If the user specifies at submission time, by way of the `qsub`(1) option `-nc`, that the job is not to be checkpointed, it is never checkpointed, even in response to a `qchkpnt`(1) command.

- The super user cannot checkpoint an NQS job directly (using the `chkpnt`(1) utility), without letting NQS know what is happening, and expect it to recover. The `qmgr`(8) command must be used to checkpoint an NQS job so that the main NQS daemon is aware of what is happening.
- Because NQS does not use the `RESTART_FORCE` flag on the `restart` invocation, any file that was in use at checkpoint time must not have been modified in a nonsequential fashion since the restart file was created. That is, the `restart` will fail if any bytes in the file between offset 0 and the file size have been modified since the checkpoint occurred. This rule allows the job to be checkpointed and to continue execution, sequentially extending output files, without invalidating the restart file.

5.5.2 Checkpoint and Restart Errors

If something goes wrong during a `chkpnt`(2) or `restart`(2) system call, an error code is returned in the global variable `errno`. For lists of such error codes, see the `chkpnt`(2) and `restart`(2) man pages.

The following section discusses how to use the `crash` program to examine the restart-information buffer to help determine why a checkpoint or recovery was not successful.

5.5.2.1 Examining the Restart-information Buffer

If you are having difficulty determining why an application will not checkpoint or restart, attempt to recreate the scenario on a relatively quiet system, and use the kernel debugger program `crash`(8) to examine the restart-information buffer.

The system is built with several restart-information buffers. A restart-information buffer is obtained and used for each checkpoint and restart operation.

The `resinfo` subcommand of the `crash` command may be used to obtain a usage summary for the restart-information buffers in a system. Using a quiet system allows you to examine an individual restart-information buffer without having it overwritten. If you have determined which restart-information buffer you want to see in detail, invoke `crash` and issue the following command to receive a detailed account of everything that is still in the restart-information buffer. The `-` (dash) option causes the long form of the listing to be output.

`resinfo -` *buffer number*

If you are not sure which buffer to examine, the `resinfo -` command displays all the restart-information buffers.

5.5.3 Recovery and Signals

The UNICOS operating system supports the automatic checkpoint and restart of batch jobs run by NQS across multiple shutdown and restart events. No modifications are needed for batch job requests run by NQS in order to take advantage of the UNICOS recovery facility, except in special circumstances.

There are two signals involved in the implementation of job and process recovery. The SIGSHUTDN signal warns of impending system shutdown, and the SIGRECOVERY signal is delivered to a recovered process. The following sections discuss these two signals.

5.5.3.1 SIGSHUTDN

To support special batch job requests that cannot be automatically checkpointed and restarted, and to allow limited recovery of interactive processes, a user process can register to catch the SIGSHUTDN signal. This indicates that the system is in the process of an orderly system shutdown.

Upon receipt of a SIGSHUTDN signal, the catching process can take steps to record its own state for later recovery, or improve its chances for recovery by closing unrecoverable files, and so on. Interactive processes can checkpoint and kill themselves by using the chkpnt(2) system call, thereby creating a restart file for later recovery.

The NQS qmgr(8) command for bringing down NQS in an orderly manner is as follows:

shutdown *grace-time*

The *grace-time* operand specifies the length of time between notifying a job with the SIGSHUTDN signal, and checkpointing or killing the job. Note that a shutdown or shutdown 0 command does not send the SIGSHUTDN signal to the jobs; rather the jobs are immediately checkpointed or killed.

Any shutdown scripts that are executed when an orderly system shutdown is imminent should first invoke the NQS qmgr command to shut down NQS, checkpointing all recoverable batch jobs, and then send the SIGSHUTDN signal to all interactive processes in the system by using the killall(8) command.

5.5.3.2 SIGRECOVERY

When a process is recovered from a restart file, either by itself or as a member of an NQS job, a SIGRECOVERY signal is posted to that process. By default, the SIGRECOVERY signal is ignored, so a process must register a signal handler

for SIGRECOVERY if there may be action necessary (for example, to restore previously closed unrecoverable files).

5.6 Kernel User Exit (uesyscall)

The uesyscall system call is a user exit into the kernel that allows you to write a site-specific system call. This function gives you access to kernel structures not otherwise available and allows a site to implement functionality in the UNICOS operating system that requires kernel support.

Warning: The kernel user exit (uesyscall) does not meet the requirements of a Cray ML-Safe configuration of the UNICOS operating system.

The structure of uesyscall is as follows:

```
int uesyscall (int subsyscall, void *paramaddress, int len);
```

subsyscall	Sub-system call number defined by the site in uex.h, the system call include file. The subsystem call allows multiple system calls from one common entry point.
paramaddress	Address of the parameter list passed to the system call. The site-defined parameter list allows different parameters to be passed to each subsystem call.
len	Length (in words) of the parameter list. The length argument allows different parameters to be passed to each subsystem call.

An entry in the system call table has been added to the uts/c1/os/sysent.c file to support the uesyscall system call. The uts/c1/md/krn_uex_syscall.c file contains the system call source. The associated uts/include/sys/uex.h include file contains user exit definitions. The source for the system call and include file are distributed with source and binary releases.

The krn_uex_syscall.c source file contains a stub routine that simply returns. The main routine parses the input parameters and calls specified subsystem calls, which allows you to write multiple site system calls.

If you want the system call to be accessible to any user, it is recommended that you write a library interface to the system call. Creating a library interface allows

extra sanity checks and validation, and provides a cleaner, more understandable user interface.

Caution: Use caution when creating a site-specific system call to avoid introducing the ability to corrupt data and to panic the system. In addition, the UNICOS kernel is now multithreaded. If a site adds code to update any tables in the kernel, you may need to place multithreading locks around the kernel structure being updated.

User Database (UDB) [6]

The user database (UDB) contains an entry for each user allowed to log into and run jobs on your system. The UDB, which replaces the traditional `/etc/passwd` authorization file, allows faster access to an individual user's information than with the user database previously supplied; it also allows safe use of multiple sources when user information is being changed.

Warning: If your site is using SecurID, there must be consistent definitions in the UDB and the SecurID database. See the preface of this manual for more information on SecurID documentation.

Processes needing information about a user do not need to know the structure or design of the UDB; library interface functions present user information in forms compatible with traditional use. Also, Cray continues to support the traditional `/etc/passwd` and `/etc/group` files.

The `/etc/passwd` file is automatically generated during UDB updates and requires no manual modification. The `/etc/group` file must be manually updated to include the proper group IDs and group names, but the group membership lists are automatically maintained when the UDB is updated. The `/etc/acid` file must be manually updated to include the proper account IDs and account names; this file has no membership lists.

The UDB is the core component of the user limits feature, which is discussed in Section 6.2.1, page 131. This section describes the UDB maintenance utilities.

6.1 Login Accounts and the UDB

The UDB consists of the files `/etc/udb` and `/etc/udb.public`, plus extension files `udb.index`, `udb.priva`, and `udb.pubva` in the directory `/etc/udb_2`. These extension files support additional fields that could not be added to the existing files. The UDB files `/etc/udb.public`, `/etc/udb_2/udb.index`, and `/etc/udb_2/udb.pubva` have public read permission and reflect all current public information in the database. Sensitive information, such as encrypted passwords and security fields, are present only in the files `/etc/udb` and `/etc/udb_2/udb.priva`.

The UDB has one entry, or record, per user or resource group. Each entry has many fields, which are divided among the UDB files. The access library `libudb`(3) collects all information belonging to a specific record and presents it

in a combined form to the calling process. For a description of the exact content of UDB entries, refer to the udbgen(8) man page.

Warning: For information on setting up accounts on a Cray ML-Safe configuration of the UNICOS operating system, see Chapter 8, page 145.

6.1.1 Providing Login Accounts

A primary responsibility of any system administrator is to provide users with system login accounts. These logins should allow users to access the system as efficiently as possible, with the smallest possible impact on the system and on other users.

A login needs the following two elements:

- An entry in the UDB
- A home directory for the login

Use the udbgen(8) command to create and maintain the UDB. The udbsee(1) utility converts the database format to an ASCII file that may be used as input to udbgen.

Although the UDB exists in an encoded form, the udbsee utility converts the UDB into a simple text format file. The udbsee utility also provides a number of selection and formatting options that may be used to extract data for reports and other administrative uses. Users can always use udbsee to view their own non-secure control parameters, and, in the form of a text file, the UDB can be transported easily from system to system and manipulated by other UNICOS commands.

Note that *dynamic* user information is updated internally in the UNICOS system but is not written into the UDB except when certain events occur. Therefore, the UDB files may not show the exact state of accumulated information. However, alterations to *fixed* user information are immediately reflected in the UDB because the database editor writes changes as soon as they have been verified by the administrator. The udbpl(8) command produces reports containing information from the kernel tables and the UDB.

6.1.2 Removing Login Accounts

To remove a login account, use the udbgen(8) `delete` subcommand to remove the relevant entry from the UDB. However, because the user's files and work in progress may be valuable to other users working with a particular project, it is

more useful initially to disable the user's ability to log in to his or her former account. To do this, use the udbgen command to disable the login:

```
udbgen -c 'update:john:permbits+:system-restricted:'
```

After the login has been disabled long enough for authorized users to retrieve any valuable files, you should delete the login's home directory and files.

You may also delete the entry in the UDB at this time; it is more useful, however, simply to keep the entry in the UDB as a record of the login's existence. This may prove useful in cases in which the deleted user owns files in directories other than the home directory and also for accounting purposes.

6.2 User Control Capabilities

UDB entries can include not only general information users need to log in and establish an initial environment, but also specific information for controlling user limits. This section describes user limits and privileges and their use.

6.2.1 User Limits

User limits are generally applied to processes or jobs and establish the maximum amount of a resource that can be requested. Limit information may be separated for job (J) and process (P) usage. Most limits are actually two values; one for batch and the other for interactive. This means that you can provide restricted resources for interactive use, for example, without limiting a user's batch resources to the same degree. The following limits are available:

Resource	Control
Number of processes	J
CPU time	J, P
Memory	J, P
Secondary data segments (SDS) (Cray PVP systems only)	J, P
Tapes	J
File allocation	J, P
PE limit	J
MPP time	J, P
MPP barrier	J

Core file limit	P
Open file limit	P
Shared memory segments (Cray T90 series systems only)	J
Shared memory size (Cray T90 series systems only)	J

Limits can be disabled by assigning a 0 value, except for CPU time, memory, file allocation (where a zero value means unlimited), and the open file limit (which can be set only within a range of values). When a new UDB record is created with a minimum set of parameters, for example, as with the following command, default values are assigned to all limit fields.

```
udbgen -c 'create:buck:uid:230:gids:10:'
```

The default values are as follows:

- Allow unlimited CPU, memory, file, and core file limits
- Deny access to all other resources (such as tapes and SDS)
- Set the open file limit to 255
- Set the process limit to the configured kernel process limit

The defaults are a property of the UDB; the administrator can change them to site-specific defaults as desired.

6.2.2 Privileges

Privileges are enforced in several ways, depending on what they are and how the privilege is handled in the system. Various mechanisms exist to identify privilege violations that are affected by this control. Some privileges, such as interactive authorization, should be made clear to the user.

There are a number of controls for establishing privileges. The following list shows those controls necessary ("Type" can be categorized as an L or an A. The L represents a single value privilege, such as true, false, or an integer, and an A represents an array of privileges):

Privilege	Type
Account IDs	A
Compartments	A
Default compartments	A

Minimum compartments	A
Default security level	L
Default integrity class	L
Default integrity category	A
Group IDs	A
Maximum security level	L
Minimum security level	L
Maximum integrity class	L
Categories	A
Permissions	A
MLS permissions	A
Site-specific	A

Note that there are two permission fields in the UDB: *permbits* for user permissions and *permits* for MLS permissions.

Some arrays may also be bit lists, as used for security controls. Some privileges, such as access to tapes and SDS, are implied by nonzero values in the relevant limit control field and do not require an explicit privilege in this category.

Space for 32 site-specific privilege bits is provided. There is no use of these bits in the released system.

6.2.3 Quota Fields

The UDB includes three fields for controlling quotas:

Field	Description
CPU quotas	Tenths of seconds allotted
CPU quotas used	Accumulated CPU time in tenths of a second
Login failures	Quotas for UNICOS MLS feature

6.2.4 Other UDB Information

There is a collection of items that do not fit neatly into the limit, quota, or privilege categories previously described. They mostly involve the areas of data migration and the fair-share scheduler, as follows:

Control	Category
Authorized shares	Share
Decaying accumulated costs	Share
Last decay time	Share
Nice increment	System
Online thresholds	Data migration
Password aging	System
Media selection	Data migration

An example of fair-share scheduling using UDB entries can be found in *UNICOS Resource Administration*.

6.3 The /etc/passwd and /etc/group Files

The /etc/passwd and /etc/group files exist and can be used with programs. They are automatically updated when the UDB is updated. You do not need to update these files, except to add group names and group IDs to the /etc/group file when a new group is added.

6.3.1 The /etc/passwd File

The /etc/passwd file (see passwd(5)) is maintained by the udbgen(8) command and is provided for compatibility with previous systems (and will never be removed because it is an integral part of the UNIX system). This file is not capable of performing UNICOS user validation.

An entry in /etc/passwd has the following format:

login:*passwd*:*uid*:*gid*:*comment*:*home*:*shell*

login The identifier for the login. The user uses this as the name under which to log in to the system. It is often helpful to the administrator (and to the system's users) if there is some evident logic behind the assignment of login names to users; typical examples include the user's initials, last name (plus a first initial), or employee identification number. Alternatively, because the who(1) utility lists login names of those logged in to the system, login names can be arbitrary strings of letters or numbers if the users of those logins require anonymity when using the system.

passwd The encrypted string representing the login's password. This string is always an asterisk (*) in a system using the UDB.

uid The numeric user identification number given to the user when he or she logs in to the login account.

gid The numeric group identification number given to the user when he or she logs in to the login account. The *gid* should be a valid group listed in the `/etc/group` file (see `group`(5) for details on the file format).

comment Traditionally used to store the name of the user for whom the login has been created.

home The path name of the login's home directory, to which the user's current directory is set when the user logs in. The home directory must exist, and its user and group ownership should be set to the user's uid and gid numbers. The directory's permissions should be set so that the owner has read, write, and execute (search) permission for the directory; the group and other permissions may be set according to local security considerations.

shell The program executed as the user's login shell program. This field may be left blank, in which case the shell program is the standard shell `/bin/ksh` (see `ksh`(1)).

6.3.2 The `/etc/group` File

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

An entry in `/etc/group` has the following format:

group:*passwd*:*gid*:*logins*

group The group name. The name can be up to 8 characters long, the first of which must be alphabetic; the remaining characters are alphanumeric.

passwd The encrypted password for the group. This field is always set to an asterisk (*) (this field is unused) in a system using the UDB.

gid The numeric group ID. The *gid* is an integer with 0 through 99 reserved for system use and 100 through 65535 available for user group identification.

logins A list of login names as defined in the UDB. The names are separated by commas.

If the list of login names is longer than approximately 400 characters, additional lines are created in `/etc/group` to hold the remainder of the membership list. All the lines for a single group are adjacent in the file, and the *group*, *passwd*, and *gid* fields in each line for each group member are identical. This accommodates groups with an arbitrarily large membership, while keeping the line length in `/etc/group` within reasonable bounds and imposing minimal impact on existing usage.

You can edit this file in order to add a new group name. To do so, add a line similar to the following:

group_name`:*:137:`

Otherwise, a group name cannot be used with the `udbgen gid` field directive, only group ID numbers. When a new group ID number is introduced through the `udbgen gid` field directive, a default name, `G-`*nnnnn*, is created (where *nnnnn* is the gid number). You may later change this name in `/etc/group` to something more meaningful.

6.4 The nu Command

You can use the `nu`(8) command to create, modify, delete, and eliminate login accounts. If you prefer to use a graphical interface to manage user logins, you can use the `xadmin`(8) command, which has all the functionality of `nu`. For additional information about the `xadmin` command, see the `xadmin`(8) man page or the online tutorial in `xadmin`.

The `nu` command handles file locking and syntax checking, and it permits the use of configurable defaults, making the process of maintaining large numbers of logins easier.

When adding new logins, the `nu` program prompts for the login ID, password, name, and other information for each new user. The program facility then creates the login, creates its directories, initializes the directory contents, and makes an entry in a log file.

When modifying logins, the `nu` program asks repeatedly for login names and instructions for the changes that are to be made to those logins. When the changes are completed, it sorts the updated login records and merges them simultaneously into the UDB.

When logins are deleted, an entry exists in the UDB for the deleted logins. This prevents those specific UIDs from being reused, and it permits accounting data to be meaningful after the accounts are deleted. The program repeatedly asks for the names of logins to be deleted and verifies the deleting of files within those logins.

The nu program can also eliminate logins. In this case, nu deletes almost all information pertaining to a specified login ID.

A complete description of the use of nu and the setup of the configuration file are presented in the *UNICOS Administrator Commands Reference Manual*.

Crash and Dump Analysis [7]

This chapter includes information on system crash analysis and recovery, including diagnosis and debugging of system problems.

Warning: This chapter contains warnings and information critical to the use of a Cray ML-Safe configuration of the UNICOS operating system.

This chapter does not answer all questions or solve all problems. If you need assistance in debugging a system crash, or if you have a recurring UNICOS kernel problem, contact Global Product Support. It is important to have available all dumps and a copy of the UNICOS operating system that caused the crash.

7.1 Introduction

This chapter describes system crash analysis and recovery, including the dump process and the operation of the UNICOS kernel debugger `crash`(8). It also discusses ways to diagnose and debug system problems.

When recovering from system crashes, you should have the following information available:

- UNICOS kernel source code, if available
- *UNICOS Administrator Commands Reference Manual*
- *UNICOS User Commands Reference Manual*

7.2 Using the `crash` Program

The `crash`(8) command is an interactive program that helps locate the problem causing the error by examining the UNICOS system image, which can be either the image of the running system or the image saved following a system crash. `crash` provides an interactive interface that formats and displays system control information from the system memory image.

After you enter the `crash` command, you can examine the following elements of the UNICOS system memory image:

- Buffer and buffer headers
- Callout table

- File table
- I-node table
- Map tables (`coremap` only)
- Buffers
- Mount table
- Process table
- Pseudo-tty table and tty table
- Other system tables
- Stack dump, traceback, and frame package formatting
- System (dump) statistics
- User structures

To use `crash` effectively, you need a dump, knowledge of the computer system instruction set, and familiarity with the UNICOS system kernel. Alternatively, you can run `crash` on an active system by using `/dev/mem` instead of a dump.

Many commands are available in the `crash` utility; those described in this manual are `proc`, `stat`, `trace`, `stack`, and `ut`. Two especially valuable features of `crash` are pipes and redirection, which allow `crash` to interface easily with other UNICOS commands.

Once inside the `crash` utility, you can enter `?` or `help` to produce an abbreviated list of commands. Refer to the `crash`(8) man page for the syntax and description of each command.

Caution: System dumps and the data obtained and written when executing the `crash` command should be labeled at `syshigh` on UNICOS MLS and Cray ML-Safe system configurations to avoid accidental disclosure of protected information.

7.3 Analyzing System Problems

This section is a guide to identifying problems that lead to a crash, not a step-by-step guide for debugging crashes. Experience in debugging and knowledge of the UNICOS kernel program are the keys to success in debugging a crash.

System crashes can be divided into the two following categories:

- Panic
- Running system

7.3.1 Panic

The UNICOS kernel program panics when it encounters an unrecoverable hardware or software problem. When UNICOS panics, a panic message is displayed on the operator's workstation. The `SYSDUMP1?` prompt is displayed and the operator is given the opportunity to perform a system dump.

7.3.1.1 Debugging Panics

To debug a panicked system, follow these steps:

1. Use the `stat` command (see `crash`(8)), which tells you why the system panicked, the time of the crash, how long the system was up, and the system name. Debugging concentrates on the CPU that detected the panic condition. This information is displayed by `stat`.
2. Executing the `stat` command reports the process slot of the panicked process. For example, the output of the `stat` command may include the following:

   ```
   Panic process: p[76]
   ```

   You can use this process number as the argument to the `stack` command, as in the following example:

   ```
   stack 76
   ```

   Executing `stack` in this manner produces the kernel stack traceback for the panicked process.
3. Use the `ut` command (see `crash`(8)) to examine the system trace buffer. The trace buffer provides a history of the most recent action within the UNICOS system. Try to correlate the trace buffer entries to the data on the stack.
4. Check the load on the system by using the `proc` command (see `crash`(8)) to display all of the current processes. Pay particular attention to the number of swapped processes or large numbers of similarly named processes.

The stack traceback or the trace buffer should have pointed you to the failing area within UNICOS. If the stack data appears to be correct, check the per-process data in the process table and user area for the crashed CPU.

Beyond this point, the number of causes of a panic are too numerous to list. Your best strategy is to use `crash`(8) to examine the details of the data structures in the failing area of the system and to compare the data values for both a running system and the UNICOS kernel source code, if available.

Caution: Security administrators should examine system dumps for security audit data that may have been generated, but not written to the official security log. The administrator can examine the dump via `crash`(8) using `slog` and `rslog`, and by using the `reduce`(8) command.

7.3.1.2 Buffer Flushing

The UNICOS operating system provides a feature that minimizes file system corruption by flushing buffered data to its target I/O devices before the system is halted. You implement this feature as a option configurable at compile-time. By default, it is enabled at configuration. To disable the feature, define the `FLUSHONPANIC` variable in the `config.h` file as `0` and compile the kernel. You may do this manually or through the install tool.

When this feature is enabled, three types of buffered data are flushed:

- Kernel data structures that must be updated on disk, such as mount, inode, and quota table entries
- UNICOS system buffer cache
- Logical device cache (`ldcache`)

This feature significantly reduces the amount of user and system data lost when a UNICOS system crashes as a result of a call to the `panic`() routine. The reduction of lost data minimizes file system corruption.

7.3.2 Running System

This section discusses systems that have not crashed, but are not running normally. You may still be able to log into the system, or there may be some users who are still logged in and are able to use the system effectively.

Sometimes it is impossible to fix a system, even when it is still running. If this is the case, log off as many users as possible, and issue the `sync`(1) and `ldsync`(8) commands. When you halt the system, get a dump so that the problem can be

studied and solved. This is an extreme measure, so it should not be done without first exhausting all measures for fixing the system while it is still active.

Perform the following steps when the system is not running normally:

1. Examine the system console for error messages.
2. If you are able to log in to the system, use the ps(1) command with the -elf options, as follows, to get a list of all processes on the system:

   ```
   ps -elf
   ```

   Check for the following conditions:

   - Swapped processes. If you notice a large number of processes swapped out (the PC_LOAD bit in per-process flags is off), try to determine what is causing the swapping. Several large processes can cause the system to begin swapping, which degrades the response time for interactive processes.
   - Running processes. A large number of running processes should not degrade interactive response time, but they can degrade turnaround time for utilities that are CPU-intensive, such as compilers, assemblers, and pattern-matching tools.
   - Zombie processes. If you notice more than a few zombie processes, look at the PGRP label of the processes in question to determine what happened to the parent process. If the parent is init, use crash to check the WCHAN address (ds address) and try to determine why init is not waiting for its child processes.

7.4 The fdmp Command

The fdmp(8) command formats system dump images. It also provides the ability to format data from the IOS, which is not available from the crash(8) command. In addition, fdmp allows 132-column output suitable for line printers.

UNICOS Multilevel Security (MLS) Feature [8]

This chapter describes the UNICOS multilevel security (MLS) feature for system and security administrators. The UNICOS security environment is established at UNICOS build time.

The information in this chapter is intended for system and security administrators. It is assumed that you have read and understood the information presented in the *UNICOS Multilevel Security (MLS) Feature User's Guide*.

In general, a secure system must protect information from unauthorized disclosure and modification. A secure system must also be designed to guarantee that established security mechanisms are correctly implemented and consistently maintained. The UNICOS MLS feature provides mechanisms to protect both system integrity and sensitive information.

Warning: In previous releases of UNICOS operating system documentation, the term *Trusted UNICOS* was used to refer to the configuration that most closely approximated the B1 evaluated configuration of UNICOS release 8.0.2. Starting in the UNICOS 10.0 release, this configuration is referred to as the *Cray ML-Safe configuration* of the UNICOS operating system. Although the Cray ML-Safe configuration of the UNICOS operating system is not an evaluated product, this configuration fully supports all functionality described in the B1 evaluation criteria.

Design specifications for the MLS feature were derived from the *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC). The UNICOS MLS feature implementation strategy is defined by the following basic DoD control objectives:

- Security policy
- Accountability
- Assurance

These Department of Defense (DoD) objectives are implemented by UNICOS MLS feature mechanisms that can be divided into the following security policies:

- Privilege and system management (also referred to as *trusted facility management*)
- Mandatory access control (MAC)

- Discretionary access control (DAC)
- Identification and authentication (I&A)
- Object reuse
- Installation and configuration
- Auditing

The following sections describe the mechanisms and associated procedures for the UNICOS MLS feature.

8.1 Overview of UNICOS Security Mechanisms

UNICOS security mechanisms are either part of the traditional UNICOS operating system (for example, file permission bits) or are enhancements to traditional UNICOS structures or functions (for example, mandatory access controls). These mechanisms and how they correlate to the *Trusted Computer System Evaluation Criteria (TCSEC)* are shown in Figure 5.

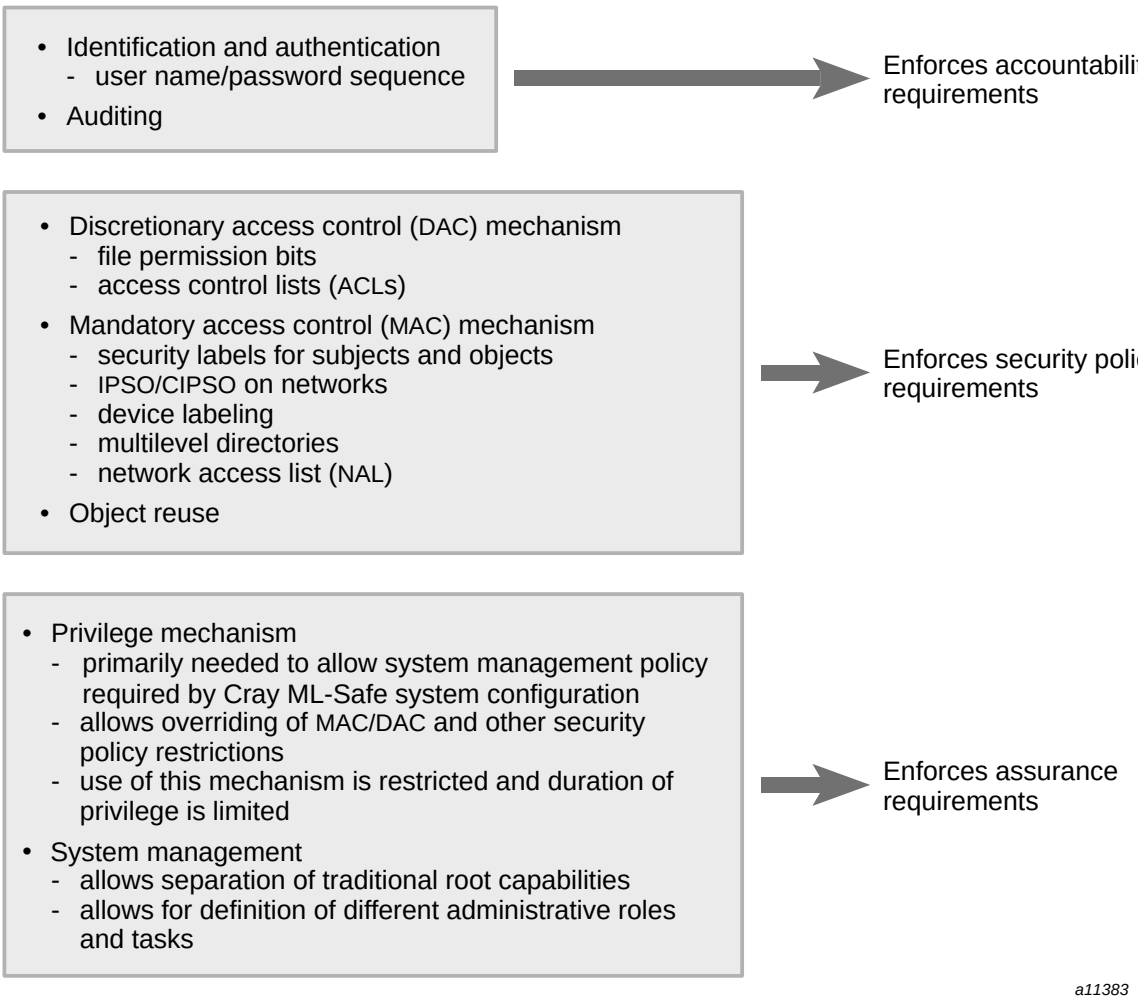


Figure 5. UNICOS Security Mechanisms

Figure 6 presents a high-level overview of how the following security mechanisms interact in the sequence of UNICOS tasks:

- Identification and authentication (I&A)
- The security label portion of mandatory access controls (MAC) and discretionary access controls (DAC)
- The PAL-based privilege mechanism

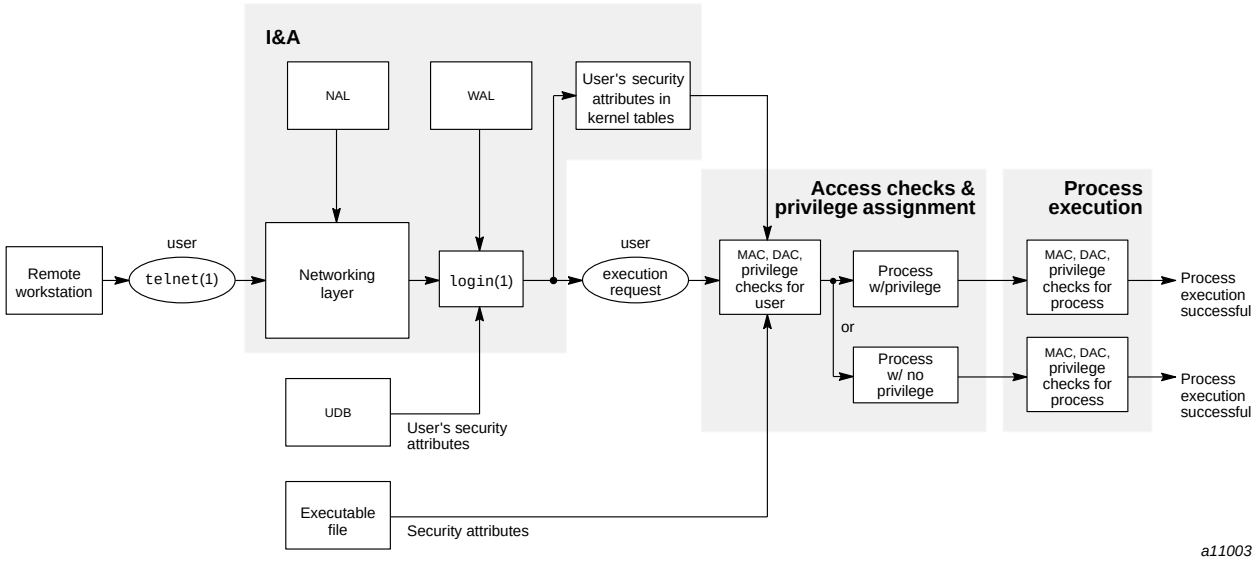


Figure 6. Interaction of UNICOS Security Mechanisms

Security auditing is not shown in Figure 6 because auditing overlays the whole system. That is, auditing records can be generated by various actions, which are not easily shown in the figure. For an introduction to auditing, see Section 8.8, page 258.

Security object reuse is not shown in Figure 6 because this mechanism is inherent in the UNICOS operating system and is not easily shown in the figure. For an introduction to object reuse, see Section 8.6, page 227.

I&A mechanisms are explained in more detail in Section 8.5, page 202. DAC mechanisms are described in more detail in Section 8.3, page 169. MAC mechanisms are described in more detail in Section 8.4, page 171. System management mechanisms are described in more detail in Section 8.2, page 148.

8.2 System Management

Many different administrative tasks are performed by multiple administrators with differing levels of expertise and authority. Operator tasks often differ from those of a security or system administrator. On a traditional UNIX system, the use of `root` (USER ID 0) allows for relatively easy administration of a computer system. `root` can override virtually all system restrictions in order to perform

a task, possibly with no way to trace the action back to a specific user. This potential lack of administrative accountability is undesireable.

System management (or trusted facility management (TFMgmt), as it is referred to in the TCSEC) allows administrative work to be accomplished on a UNICOS system. The principal requirements of system management, as defined in *A Guide to Understanding Trusted Facility Management*, publication NCSC-TG-015, are as follows:

- The separation of operator and administrator functions.
- The logical (or physical) separation of the database information corresponding to those functions.
- The implementation of least privilege such that functions have only the minimum necessary privileges to the databases.

On a UNICOS system, system management is supported by the following mechanisms:

- A super-user mechanism
- The privilege assignment list (PAL)-based privilege mechanism

The super-user mechanism (enabled by the `PRIV_SU` configuration parameter) allows the `root` user to override virtually all system restrictions. This mechanism provides the traditional method of system administration support on a UNICOS system.

The PAL-based privilege mechanism uses privilege assignment lists (PALs) to support the *principle of least privilege*, which is the ability to grant each subject the most restrictive set of privileges for only as long as needed to perform a set of authorized tasks.

With the PAL-based privilege mechanism, PALs are associated with files that administrative users typically execute. When a user executes a file, privilege attributes from the PAL are assigned to the resulting process. The assigned privilege attributes can vary, depending on the active category of the user.

A process with no active categories (that is, running on behalf of a nonadministrative user) can also be assigned privilege attributes from the PAL of the file. This provides the traditional set-user-ID-root functionality by allowing nonadministrative users to perform limited administrative functions in a controlled environment.

The PAL-based privilege mechanism is always available and cannot be configured through a configuration option. However, this mechanism is

effective only after PALs have been assigned to files by using the `privcmd`(8), `setpal`(7X), and / or `setprivs`(8) commands.

Warning: In UNICOS 9.2 and later releases, all sites are required to assign PALs. The supported privilege configurations are as follows:

- PALs augmented by `PRIV_SU`
- PALs only

The following sections provide more information about the super-user mechanism, the PAL-based privilege mechanism, and information about UNICOS categories used by the PAL-based privilege mechanism.

8.2.1 The Super-user Mechanism (`PRIV_SU`)

The UNICOS operating system supports a fully functional super-user mechanism. This mechanism works as the super user does on earlier UNICOS systems that do not use the MLS feature, and it allows a process with an effective user ID of 0 to override UNICOS restrictions.

The `PRIV_SU` configuration parameter enables the use of the super-user mechanism. To enable or disable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Super-user privilege policy?` selection in the UNICOS Installation and Configuration Menu System. The default setting for this mechanism is `ON`.

8.2.2 UNICOS Categories

UNICOS categories are used on systems that use the PAL-based privilege mechanism to identify administrative roles. Each role is represented by a category. A category has a name and a corresponding bit value within a 32-bit mask. The following categories are defined in the `tfm.h` file:

Category	Description
`secadm`	Defines the security administrator role
`sysadm`	Defines the system administrator role
`sysops`	Defines the system operator role
`unicos`	Not currently used / reserved for use by Cray
`sysfil`	Not currently used / reserved for use by Cray
`archive`	Not currently used / reserved for use by Cray

`datamgr`	Defines the data migration daemon
`netadm`	Not currently used / reserved for use by Cray
`diagadm`	Defines the diagnostic administrator role
`daemon`	Not currently used / reserved for use by Cray
`system`	Defines the single-user mode administrator for systems using the PAL-based privilege mechanism.
`smail`	Not currently used / reserved for use by Cray

Note: The category names described in the previous list are reserved for use by Cray.

The user responsible for creating or modifying administrative accounts should assign each administrative user an appropriate set of authorized categories in the user database (UDB). The authorized categories define the set of categories available to the administrative user.

An administrative user can activate an authorized category (or categories) by using the `setucat`(1) command. The user's active category identifies his or her current administrative role.

The UNICOS Installation and Configuration Menu System does not support the ability for a site to define local administrative categories. If your site requires categories other than those defined by Cray, a manual procedure for defining local administrative categories is available to sites with access to UNICOS source code.

8.2.3 The PAL-based Privilege Mechanism

The PAL-based privilege mechanism uses categories to define administrative roles and abilities. The following general policy is used by the PAL-based privilege mechanism to define the types of abilities allowed to the administrator roles:

- Users with an active `system` (for single-user mode) or `secadm` category are allowed to override all command restrictions, including security label restrictions.
- Users with an active `sysadm` category are allowed to perform typical system administrator functions, but are usually constrained by security label restrictions.

- Users with an active `sysops` category are limited to performing typical system operator functions.

The administrative abilities that are specific to a command on a system using the PAL-based privilege mechanism can be found in the man page documentation for that command. Administrative abilities allowed by a command can be modified by updating the privilege assignment list (PAL) associated with that command.

Warning: The PAL-only configuration of the UNICOS operating system is highly restrictive, designed to meet the needs of the evaluated configuration. This configuration is only recommended where the evaluated configuration is absolutely required.

The system management policy supported by the PAL-based privilege mechanism does not automatically grant special abilities to users based on their active category. Rather, special abilities are granted according to the effective privileges and privilege text of a process.

A process is assigned privileges and privilege text when the user executes an executable file that has been assigned a PAL. A PAL provides mapping from a user's active category to the privileges and privilege text of the process. In general, the UNICOS kernel grants special abilities based only on the effective privileges of a process, while UNICOS commands grant special abilities based only on the privilege text of a process.

An administrator can use the `setpal`(8) and `setprivs`(8) to set the PAL attributes of a file. Also, the `privcmd`(8)command can be used to assign the MLS attributes (including PALs) to files.

For an overview of how PALs and categories work to enforce the assignment of privileges, see Section 8.2.3.1.

8.2.3.1 Overview of Process Privilege Attributes

The following sections provide an overview of how process privilege attributes that are obtained from the privilege assignment list (PAL) are used.

8.2.3.1.1 The Process

When a user executes a file, the resulting process is assigned privilege attributes based on the user's active category and privileges and the PAL of the file. The privilege attributes of the process consist of permitted privileges, effective privileges, and privilege text. *Effective privileges* are checked by the UNICOS kernel and allow a process to override specific restrictions. *Permitted privileges*

are the privileges that a process can make effective. *Privilege text* is checked by UNICOS commands and allows a process to perform special actions that are controlled entirely within the command.

8.2.3.1.2 Privileges

The UNICOS operating system has a unique privilege for each ability that is traditionally associated with the user `root`. For example, the `PRIV_DAC_OVERRIDE` privilege overrides the permissions mode and access control list (ACL) protections on any object.

The UNICOS system also has defined privileges to override the UNICOS security restrictions. For example, the `PRIV_MAC_WRITE` privilege overrides the mandatory access controls for writing to an object.

When a process issues a system call to request a restricted action, the UNICOS kernel verifies that a process has the specific effective privilege(s) needed to perform the action. For example, a process may issue the `open`(2) system call to open a file, but is denied discretionary access control (DAC) access by the permissions mode and ACL of the file. For this example, the UNICOS kernel would verify if the process had the `PRIV_DAC_OVERRIDE` privilege. If it does, then the DAC access is granted; if not, the kernel rejects the request to open the file.

8.2.3.1.3 Privilege Text

Although most processes rely on checks done by the UNICOS kernel to control privileged operations, there can be times when the kernel cannot completely enforce the desired policy. This is when privilege text is used.

An example of using privilege text can be shown with the `passwd`(1) command. On a traditional super-user system, the `passwd` command allows a user to change any user's password if the user executing the command has the real user ID of `root`. If the user is not `root`, then `passwd` allows the changing of only the current user's password.

On a system using the PAL-based privilege mechanism, the `passwd` command verifies if it is running with the `chgany` privilege text. If it is, the `passwd` command allows the changing of any user's password. If not, the `passwd` allows the changing of only the current user's password. If the super-user mechanism is also being enforced, the `passwd` command also allows real user ID `root` to change any user's password.

Enforcement of, and abilities granted by, a specific privilege text is unique for every command. The `chgany` privilege text can be used in other commands to control restricted actions that are unrelated to changing passwords. Conversely, if another command can be used to change passwords, it can use any privilege text value to enforce the same policy as the `passwd` command.

8.2.3.2 UNICOS Security Privileges

The UNICOS privileges are a granular representation of traditional super-user abilities that are enforced within the UNICOS kernel. That is, instead of allowing a process with effective user ID 0 to override all UNICOS kernel restrictions, a set of specific privileges have been defined that allow a process to override specific sets of UNICOS kernel restrictions.

A process with the following privileges effective is allowed to perform the action described:

Privilege	Description
`PRIV_ACCT`	The following is allowed: • Allowed to use the `acct`(2) system call, which is used to enable or disable process accounting. • Allowed to use the `dacct`(2) system call, which is used to enable or disable process or daemon accounting. • Allowed to use the `devacct`(2) system call, which is used to control device accounting. • Allowed to use the `wracct`(2) system call, which is used to write accounting records.
`PRIV_ADMIN`	Allowed to perform restricted network-related administrative functions. Also allowed to perform various restricted system administrative functions where other privileges do not apply.
`PRIV_AUDIT_CONTROL`	The following is allowed: • Allowed to open the audit log device file (`/dev/slog`) by using the `open`(2) system call. This privilege is used by the auditing daemon to manage audit data.

- Allowed to disable kernel auditing for itself by using the setusrv(2) system call. This privilege is used by Cray ML-Safe processes to manage their own auditing.
- Allowed to obtain its current auditing state by using the getusrv(2) system call. If a process does not have this privilege effective when calling getusrv, the current auditing state that is returned is indeterminate. This functionality prevents a process from determining whether its actions are being audited.

PRIV_AUDIT_WRITE

Allowed to use the slgentry(1X) system call, which is used to write data to the audit trail.

PRIV_CHOWN

When {_POSIX_CHOWN_RESTRICTED} is enabled, a process that uses the chown(2) system call is allowed to change the owner of a file and specify a group to which it does not belong.

PRIV_DAC_OVERRIDE

Allowed to override the permission bit and access control list (ACL) protections on named objects. This privilege is applicable to system calls that accept path name parameters.

PRIV_FOWNER

Allowed to act as the owner of a file. This privilege is applicable to system calls that are used to set file attributes.

PRIV_FSETID

Overrides the following restrictions:

- The effective user ID of the calling process must match the file owner when setting the set-user-ID (S_SUID) or set-group-ID (S_SGID) mode bits on that file.
- The effective group ID or one of the supplementary group IDs of the calling process must match the group ID of the file when setting the set-group-ID (S_SGID) mode bits on that file.
- The set-user-ID (S_SUID) and set-group-ID (S_SGID) mode bits on a file are cleared upon successful return from the chown(2) system call.

- When FSETID_RESTRICT is enabled, a process cannot create or manipulate set-user-ID or set-group-ID files as allowed on traditional UNICOS systems.

PRIV_IO

Allowed to perform restricted tape and disk I/O related functions. This privilege is applicable to system calls that make use of I/O drivers.

PRIV_KILL

Allowed to send a signal to a process that it does not own.

PRIV_LINK_DIR

Allowed to create or delete a hard link to a directory.

PRIV_MAC_DOWNGRADE

The following is allowed:

- Allowed to downgrade the active security label of an object.
- Allowed to relabel a socket as multilevel or "fuzzy."

PRIV_MAC_READ

The following is allowed:

- Allowed to override security label protections on an object when attempting to gain read, execute, or search permission to that object.
- After a process obtains read access to an object (for example, through the open(2) system call), this privilege ensures that the process may continue to read the contents of that object, even if the security label of that object is modified.

PRIV_MAC_RELABEL_SUBJECT

Allowed to set its authorized and/or active MLS attributes to any value.

PRIV_MAC_UPGRADE

The following is allowed:

- Allowed to upgrade the active security label of an object.
- Allowed to relabel a socket as multilevel or "fuzzy."

PRIV_MAC_WRITE

The following is allowed:

- Allowed to override security label protections on an object when attempting to gain write permission to that object.
- After a process obtains write access to an object (for example, through the open(2) system call), this privilege ensures that the process may continue to write the contents of that object, even if the security label of that object is modified.

PRIV_PAL_KEEP

Overrides the restriction that, when the contents of a file is modified, all associated file privileges and PAL category records are cleared.

PRIV_POWNER

Allowed to act as the owner of a process. This privilege is applicable to system calls that are used to set or retrieve attributes of other processes.

PRIV_PROC_ACCESS

Overrides the restriction that a /proc file process that is not the calling process cannot be accessed if the /proc file process has the PC_NOCORE flag is set.

PRIV_RESOURCE

Allowed to use system calls that set or retrieve session resource attributes (for example, limits, nice values, and so on).

PRIV_RESTART

Allowed to create or set the attributes of a restart file.

PRIV_SETFPRIV

Allowed to set file privileges and PALs.

PRIV_SETGID

Allowed to change its real group ID.

PRIV_SETUID

Allowed to change its real or saved user IDs.

PRIV_SOCKET

Allowed to access a privileged socket.

PRIV_TIME

Allowed to set a time adjustment value for the system clock.

8.2.3.3 Process Privileges

A process is allowed to override system restrictions that are enforced by the UNICOS kernel only if it possesses the appropriate granular privilege or privileges that are required to override those restrictions. Every process has two sets of privileges: permitted and effective. These privileges are defined as follows:

Privilege set	Description
Permitted privileges	Privileges that are authorized for use by a process. These privileges do not allow a process to override system restrictions. The permitted privileges of a process are a superset of its effective privileges.
Effective privileges	The privileges with which a process is currently functioning. These privileges are checked by the kernel to determine if the process can override system restrictions. The effective privileges of a process are a subset of its permitted privileges.

A process can add or remove any of its permitted privileges to or from its effective privileges. The permitted privileges serve as a base set of privileges that can be made effective. A process cannot add privileges to its permitted privileges, but it can remove privileges.

8.2.3.4 Privilege Assignment List (PAL)

A process is assigned permitted and effective privileges when a user executes an executable binary file that has been assigned a privilege assignment list (PAL). A PAL is comprised of file privileges and PAL category records, which are explained in the following sections.

8.2.3.4.1 File Privileges

Every executable binary file has three sets of privileges: allowed, forced, and set-effective. When a process executes a executable binary file, the permitted and effective privileges of a process are modified, based on the privileges of the file. If

a file has not been explicitly assigned file privileges, the default value for each set of privileges is the null set. File privileges are defined as follows:

Privilege set	Description
Allowed privileges	The maximum set of privileges that are inherited from the process that executed the file. The intersection of the allowed privileges with the permitted privileges of the process becomes the permitted privileges of the new image process.
Forced privileges	These privileges are unconditionally added to the permitted privileges of the new process image.
Set-effective privileges	These privileges are made effective for the new process image. Only the privileges that are also permitted for the process can be made effective.

An administrator can modify the privilege sets of a file by using the `setprivs`(8) command.

8.2.3.4.2 PAL Category Records

After the permitted and effective privileges of a process have been initialized using the allowed, forced, and set-effective file privileges, the permitted and effective privileges of a process are further constrained based on the user's active category. This refinement is performed by using PAL category records.

A PAL category record is comprised of three components: an active category, privileges, and privilege text. Each record is in the following form:

```
Active_category:Privileges:Privilege_Text
```

When a user's active category matches that of a PAL category record, the permitted and effective privileges of a process are further constrained by the privileges specified in that record. A `PRIV_NULL` privilege in the `Privileges` field indicates that no privileges are assigned to the process.

The privilege text value in the record is also assigned to the process. A `TEXT_NULL` privilege text in the `Privilege_Text` field indicates that no privilege text is assigned to the process.

Every executable binary file has at least one PAL category record. If a file has not been explicitly assigned a PAL category record, the default record is as follows:

```
other:PRIV_NULL:TEXT_NULL
```

If a user's active category does not match any PAL category records, then the `other` PAL category record is used to further constrain privileges and to assign privilege text. If the user has no active category, the `other` PAL category record is used. The privileges and privilege text of the `other` PAL category record can be modified, but the `other` PAL category record cannot be removed.

Warning: Cray has defined PALs for executable files in the set of Cray ML-Safe components. If a site modifies a UNICOS Cray-ML Safe component PAL for a file to define a local administrative policy, all PAL category records must be defined in terms of the privileges and privilege texts defined by Cray for that file. The site should modify only the active categories specified in the PAL category records and include only the privilege text values as specified on the man page for that file.

Assigning PALs to files that are not part of the set of Cray ML-Safe components, or modifying UNICOS Cray ML-Safe component PALs for files such that a PAL specifies a set of privileges not defined for that file by Cray, should not be done on a Cray ML-Safe system configuration.

An administrative user can update the PAL category records of a file by using the `setpal`(8) command.

8.2.3.4.3 Privilege Text

Privilege text is assigned to a process through a PAL category record. This text is a character sequence of up to 8 characters in length. Privilege text is associated only with PAL category records that have a specific active category. That is, it is not typically associated with the `other` PAL category record.

Privilege text is used to identify a type of administrative user. For example, a command may grant special abilities to a specific type of administrative user. However, if the command just checked for a specific active category, there would be little flexibility for a site to customize the administrative policy.

Instead, commands can check for a specific privilege text to determine if the user should be granted special abilities. This allows a site to customize the administrative policy by updating the PAL category records with various active categories, but specifying the privilege text that the command checks.

A command uses the `cmptext`(2) system call to check privilege text values. You can use the `privtext`(1) command to display the privilege text that is assigned if a user executes a specific command. See the *UNICOS User Commands Reference Manual* for examples on using this command.

8.2.3.5 Propagation of Privileges

Figure 7 shows how the process privilege state, the file privileges, and the privilege assignment list (PAL) form and propagate privileges across an exec(2) system call.

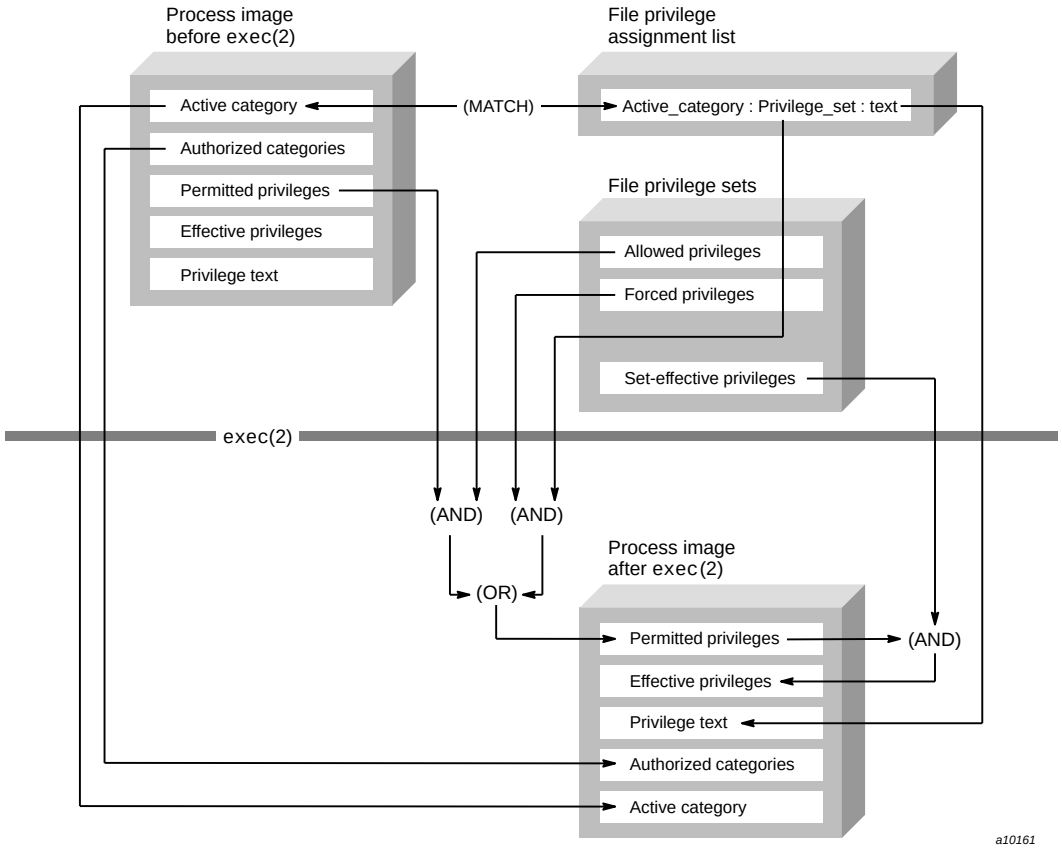


Figure 7. Propagation of Privileges

8.2.3.6 Super-user PALs

To make administrative applications function on a system using privilege assignment lists (PAL), changes to application source code are often required. Simply assigning a privilege assignment list to an application does not guarantee that the application will run properly.

Administrative applications often contain internal checks for real and/or effective user ID 0 and can grant special abilities to users with those attributes. Set-user-ID `root` applications often toggle their effective user ID between user ID 0 and the user's real user ID to control the availability of traditional root abilities.

On a system using PALs, dependence on user ID 0 is not desirable because administrative users on such a system do not necessarily run with user ID 0. Applications that check for user ID 0 may not grant administrators the abilities that they require or, in the worst case, the application can simply fail to run.

Modifying application source code to make it work on a system using PALs may not be an option for some Cray sites. Sites may not have the technical understanding of application source code that is necessary to safely modify it or do not always have access to application source code.

The goal of super-user PALs is to provide a relatively easy way for customers to make applications work on a system using PALs without requiring modifications to application source code.

To make an administrative application run on a system using PALs, the application executable binary file must be assigned the `priv_root` flag. This flag can be assigned using the `spset`(1) command.

The file should also be assigned a PAL. To guarantee that the program runs with all traditional `root` abilities, the following file privileges should be assigned:

File privileges	Privilege
Allowed	`PRIV_NULL`
Forced	`PRIV_ALL`
Set-effective	`PRIV_ALL`

The following PAL category record should also be assigned:

```
system:PRIV_ALL:TEXT_NULL
other:PRIV_NULL:TEXT_NULL
```

Additional category entries can also be specified to allow privileged execution by various types of administrators.

When a user executes an executable binary file that has been assigned a PAL and the `priv_root` flag, permitted and effective privileges and privilege text are initially assigned to the process according to the usual privilege assignment algorithm. However, once permitted and effective privileges have been assigned, they propagate across subsequent program executions and are not constrained by PALs.

Also, the real and saved user IDs of the process are forced to 0. If the file is a set-user-ID file, the effective user ID of the process is initialized to the file owner. Otherwise, the effective user ID of the process is forced to 0.

Forcing user IDs to 0 is necessary to address potential user ID 0 checks within the application. This means that when an administrator executes such a file, the process runs as if it were executed directly by the `root` user regardless of the administrator's original user IDs.

If a process sets the value of its effective user ID to 0, all of the permitted privileges of the process are made effective. If a process sets the value of its effective user ID to nonzero, the effective privileges of the process are cleared.

If a process uses the `setuid`(1X) system call to change its real, effective, and saved user IDs to nonzero values (that is, removes its ability to function as the `root` user), the permitted and effective privileges of the process are automatically cleared.

8.2.3.7 Software Not Part of the Set of Cray ML-Safe Components

Cray has defined PALs for executable files in the set of Cray ML-Safe components. UNICOS commands that are not included in the set of Cray ML-Safe components may not have PALs defined by Cray. Such commands may be required for administrative tasks, but will not function in a strict PAL-based privilege environment because PALs (or super-user PALs) have not been assigned.

The UNICOS Installation and Configuration Menu System is assigned a super-user PAL and allows escaping to a specially-privileged shell. By escaping to this shell and running an administrative command that is not included in the set of Cray ML-Safe components, it allows the command to function properly in a strict PAL-based privilege environment.

Warning: Using the specially-privileged shell in multiuser mode to run commands that are not included in the set of Cray ML-Safe components should not be done on a Cray ML-Safe system configuration. The specially-privileged shell should be used only from single-user mode.

8.2.3.8 Determining PAL Privileges

To determine the privileges to include in the PAL for a command, the command should be analyzed by investigating the code and associated documentation. The following example shows how the `cat`(1) command can be analyzed.

The cat command displays file contents. This involves reading data from one or more files and writing that information to standard output. The cat command depends only on the UNICOS kernel to grant special abilities to administrative users.

For administrative users, the cat command (through the UNICOS kernel) allows reading data from any file and writing that information to standard output (which can be redirected to any file). The privileges involved in overriding the security label, permission bit, and access control list (ACL) protections of any file are the PRIV_MAC_READ, PRIV_MAC_WRITE, and PRIV_DAC_OVERRIDE privileges.

If you do not completely understand the high-level functionality of a command, you can construct a set of required privileges by determining the privileges that are used in each system call that the command directly or indirectly invokes. The man pages for the system calls contain the privileges associated with each system call.

The cat command does not internally alter its behavior based on any user attributes (for example, user ID, active category, and so on). This means that only the TEXT_NULL privilege text is required.

The cat command should not grant special abilities through the UNICOS kernel to nonadministrative users. This means that the PRIV_NULL privilege should be associated with nonadministrative users, which is done as follows:

```
other:PRIV_NULL:TEXT_NULL
```

The default administrative policy for PAL-based privilege ensures that users with an active system or secadm category are allowed to override all mandatory access and discretionary access control protections of a file. This means the PAL category records for the system and secadm categories should contain the PRIV_MAC_READ, PRIV_MAC_WRITE, and PRIV_DAC_OVERRIDE privileges.

The default administrative policy for PAL-based privilege ensures that users with an active sysadm category can override only the discretionary access control protections on a file. This means the PAL category record for the sysadm category should contain only the PRIV_DAC_OVERRIDE privilege.

This results in the following PAL category records:

```
system:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
secadm:PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE:TEXT_NULL
sysadm:PRIV_DAC_OVERRIDE:TEXT_NULL
other:PRIV_NULL:TEXT_NULL
```

Defining the allowed, forced, and set-effective file privileges can be based on the privileges in the PAL category records. Privileges are rarely, if ever, inherited from a previous process. This means the allowed privileges contain only PRIV_NULL. The forced and set-effective privileges contain all the privileges specified in the PAL category records. This means the file privileges consist of the following:

```
allowed = PRIV_NULL
forced = PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE
set-effective = PRIV_DAC_OVERRIDE,PRIV_MAC_READ,PRIV_MAC_WRITE
```

8.2.3.9 Process Privilege Management

A process should begin execution with the minimum set of permitted privileges that are necessary to complete its task. Some or all of those privileges may also be effective upon execution using the set-effective privileges of the executable file.

A process does not have to remove effective privileges if leaving those privileges effective for the life of the process does not cause the process to perform an action that compromises the system security policy. If a process must perform an action that, because one or more privileges are effective, causes the process to perform an action that places the system security policy at risk, the process should remove the undesired effective privileges.

If a process must execute a file, the process should remove privileges that are not required for propagation from its permitted (and effective) privileges.

UNICOS library routines allow a process to set, retrieve, and manipulate its permitted and effective privileges. The names of these routines have the prefix priv_. See the *UNICOS System Libraries Reference Manual* for more information on these routines.

8.2.3.10 Privilege Text Management

A program that internally verifies user IDs and/or categories does so by retrieving these attributes with the getuid(2), geteuid(2), and getusrv(2) system calls. The retrieved information is then compared against a desired value (for example, user ID 0, secadm category, sysadm category, and so on) and the program alters its behavior accordingly.

The altered behavior may consist of granting special abilities to a user who possesses specific attributes and granting more restrictive abilities to all other users. For example, on UNICOS 7.0 MLS systems, the /bin/passwd command allows a user with real user ID 0 to modify any user's password, but restricts all

other users to modifying only their own password. The `/bin/passwd` program accomplishes the check for real user ID 0 using the `getuid` system call.

In a PAL-based privilege environment, administrative tasks are not performed by users with user ID 0. Instead, programs grant special abilities based on the privilege text. Programs use the `cmptext`(2) system call to check privilege text values. The `cmptext` system can be used to verify if a user has effective user ID 0, real user ID 0, or a specific privilege text value. This system call is used as a replacement for the `getuid` and `geteuid` system calls to determine if a user should be granted special abilities.

8.2.4 Privileged Shell

The UNICOS user environment relies on the shell for command execution, I/O redirection, and management of a user's session. Proper execution of the shell requires that an administrator set up the necessary conditions for the correct execution of commands using the shell. The administrative work may include changing the working directory, redirecting I/O, setting the label of the process, and so on.

Note: On the UNICOS system, the default shell (`/bin/sh`) is the Korn shell.

To make it possible for administrators to execute these administrative functions correctly on a UNICOS system, privilege is needed in some cases. The privileged shell is used on UNICOS systems to provide the mechanism needed to enable and manage privileges within the shell when appropriate. The privileged shell is available only if the system is using the PAL-based privilege mechanism.

This privileged shell can be used to accomplish the following tasks by the security administrator, system administrator, and system operator roles:

- The security administrator can override all MAC and DAC restrictions on files and directories, as well as override all restrictions on killing processes, setting the shell process label, and changing the shell process resource restrictions. The following shell built-in actions are privileged for the security administrator:
  - I/O redirection
  - File name expansion
  - `cd`(1)
  - `echo`(1)
  - `kill`(1)

  - print (see ksh(1))
  - pwd(1)
  - read(1)
  - setusrv(1)
  - setucmp(1)
  - setulvl(1)
  - test(1)
  - ulimit (see ksh(1))

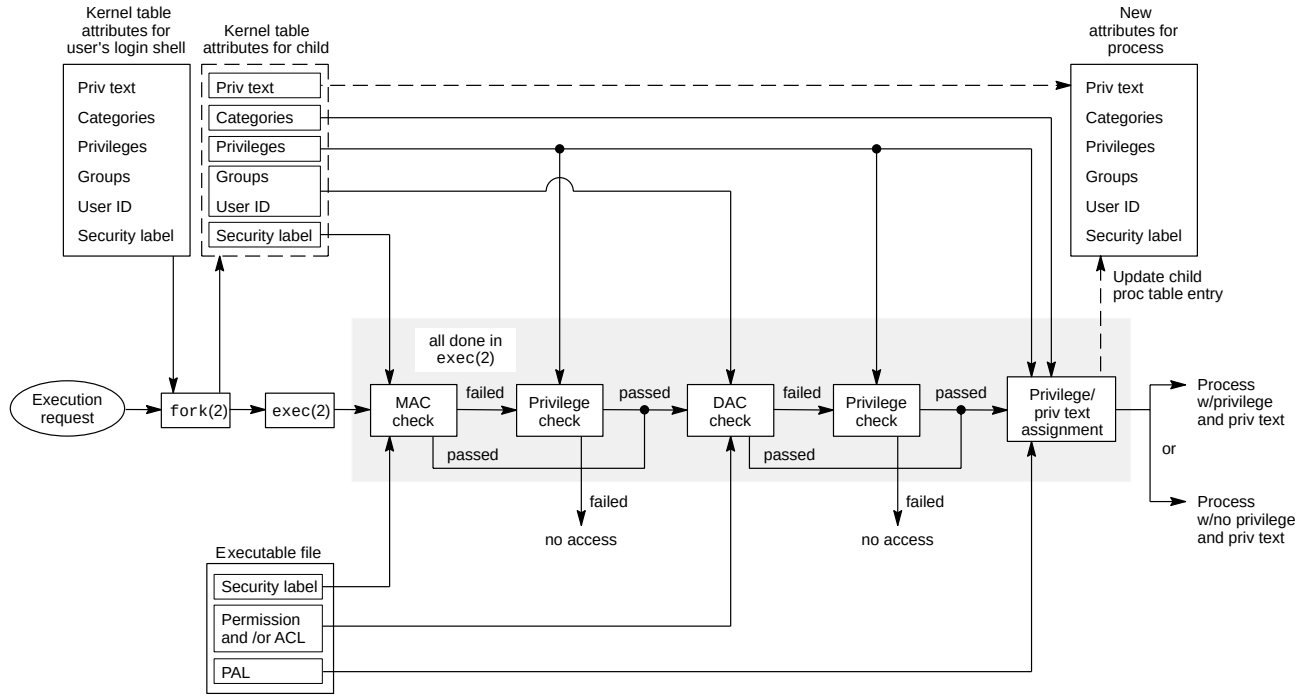
- The system administrator can override DAC restrictions on files and directories, kill processes regardless of process ownership (subject to the MAC restrictions), and override restrictions on changing the shell process resource limits. The system administrator does not have the authority to override MAC restrictions or set process labels. The following shell built-in actions are privileged for the system administrator:
  - I/O redirection
  - File name expansion
  - cd(1)
  - kill(1)
  - pwd(1)
  - test(1)
  - ulimit (see ksh(1))
- The operator can kill processes regardless of process ownership (subject to the MAC restrictions) and override restrictions on changing the shell process resource limits. The operator does not have the authority to override MAC restrictions, override DAC on files, or set process labels. The following shell built-in actions are privileged for the operator:
  - kill(1)
  - ulimit (see ksh(1))

To use the privileged shell, an administrator enters the shell by activating the appropriate administrative or operator category (by using the `setucat`(1) command) and executing a subshell.

8.2.5 Overview of Access and Privilege Checks

This section shows the relationship between mandatory access control (MAC), discretionary access control (DAC), and privilege checks by describing a nonadministrative user login and command execution sequence.

After `login`(1) successfully sets up the kernel table MLS attributes for the user' s `login` shell, the user is free to execute a command. As shown in , this results in a new child process (which is essentially a mirror image of the `login` shell).



To execute an object, the security label of the subject must dominate the security label of the object. If not, a privilege check is done. *a11005*

Figure 8. Overview of Initial MAC / DAC Checks and Assigning of Privileges

The exec(2) system call then verifies that the child process has access to the command.

MAC access is granted if either of the following conditions are true:

- The active security label of the child process dominates the security label of the command
- The child process has the appropriate privilege to override the MAC policy for file execution

If neither condition is true, then execute permission to the file is denied.

If either condition is true, then DAC checks are performed. The group IDs and effective user ID assigned to the login shell are compared against the permission bits and ACL of the executable file, as defined by the algorithm outlined in the *UNICOS Multilevel Security (MLS) Feature User's Guide.*

If DAC access is granted, the process is assigned permitted privileges, effective privileges, and privilege text based on the user's active category and the PAL of the file. The effective user and group IDs of the process could change if the file was a `setuid` or `setgid` program.

8.3 Discretionary Access Control

Discretionary access control (DAC) is implemented through a set of rules that control and limit access to an object, based on an identified individual's need to know and without intervention by a security officer for each individual assignment.

This is accomplished by the use of standard UNICOS mode permission bits and an access control list (ACL); the ACL and mode bits allow the owner of a file to control read (r), write (w), and execute (x) access to that file. The file's owner can create or modify an ACL that contains the identifiers and the r / w / x permissions for those individuals and groups that are allowed to access the file.

An ACL contains one or more ACL entries; each ACL entry defines file access information for a user. The entry contains a user field (which defines the user's login name), a group field (which defines the group name), and the permissions field (which is used to define any combination of r / w / x or define no (n) access).

The ACL entries, which define the absolute permissions, are intersected with the file's group (mask) bits to determine the type of discretionary access allowed; this is called the effective permissions.

Object access is always governed by the mandatory policy restrictions established by the security administrator.

Refer to the section on using ACLs in the *UNICOS Multilevel Security (MLS) Feature User's Guide* for more information on the following:

- How ACLs are used
- Examples of how to create and maintain ACLs

You can also refer to `spacl`(1) in the *UNICOS User Commands Reference Manual*.

8.3.1 `umask` on a MLS System

On a Cray ML-Safe system configuration, the default setting for `umask`(1) is 077. This default is defined in the `umask` command in `skl/c1/etc/profile` and `skl/c1/etc/cshrc`.

This default should not be changed on a Cray ML-Safe system configuration. Users of `umask` are affected in the following ways:

- Users that rely on new files automatically having group and/or world access must manually set the file access permission bits to enable this access.
- Users who want a `umask` other than 077 must place the `umask` command in their `$HOME/.profile` or `$HOME/.login` file.

Site administrators should ensure that any affected users are made aware of these changes.

8.3.2 Managing Set-user-ID and Set-group-ID Files

The UNICOS set-user-ID (setuid) or set-group-ID (setgid) functionality can be very useful, although it poses some security risks. A poorly designed setuid or setgid program can compromise the security of the owner or owning group of a file. When the owner of a file grants access permission to other users, the owner effectively relinquishes control over the setuid or setgid file.

Previous releases of the UNICOS system with the MLS feature enabled have restricted the management of setuid and setgid files. The UNICOS system uses the `FSETID_RESTRICT` configuration parameter to allow sites to restore the UNICOS non-MLS functionality of setuid and setgid files.

When `FSETID_RESTRICT` is enabled, only appropriately authorized users can create or link to setuid or setgid files. Only appropriately authorized users can

open setuid or setgid files for writing. Also, for chmod(2), chown(2), and for setting access control list (ACL) operations, the setuid and setgid mode bits of a file are cleared if the user is not properly authorized.

When FSETID_RESTRICT is not enabled, the UNICOS (POSIX) policy on setuid and setgid files is enforced. In addition, when setting an ACL on a file, the setgid mode bit of a file is cleared if the user does not belong to the owning group of the file and is not an appropriately authorized administrative user. This functionality makes the set-ACL behavior consistent with the behavior of chmod.

The FSETID_RESTRICT parameter is enabled by default. To enable or disable this parameter, use the Configure system->Multilevel security (MLS) configuration->System options->Restrict setuid/setgid file creation selection in the UNICOS Installation and Configuration Menu System.

Note: For the UNICOS 10.0 release, the default value of the FSETID_RESTRICT configuration parameter will be changed to OFF.

The spcheck(8) utility allows the security administrator to search the system for setuid and setgid files and to maintain surveillance of their use. See Section 8.8.9, page 341, for more information.

8.4 Mandatory Access Control

Mandatory access control (MAC) is implemented through the UNICOS security policy, which is a set of rules that control access based directly on a comparison of the subject's clearance and the object's classification. The UNICOS security policy is enforced for all attempts to access an object.

The security policy controls read and write operations in order to prohibit unauthorized disclosure of any system or user information. The security policy is defined as the set of rules and practices by which a system regulates the disclosure of information. The mandatory security policy enforced by the UNICOS MLS feature is as follows:

- A subject may read or execute an object only if the active security label of the subject dominates the security label of the object.
- A subject may write to an object only if the security label of the subject is equal to the security label of the object.

A security label consists of a security level and zero or more security compartments. A maximum of 17 hierarchical security levels and 63

nonhierarchical compartments are used to represent a nonadministrative subject's clearance and an object's classification on the UNICOS system.

The security levels can range from through 16, with having the lowest value and 16 the highest. The UNICOS system also uses system high (`syshigh`) and system low (`syslow`) labels. See Section 8.4.2.1, page 185, for more information on these labels.

An appropriately authorized administrator can use the `nu`(8) or `udgben`(8) commands to assign each user a minimum, maximum, and default security level in the user database (UDB). The security administrator also assigns each user a set of active and authorized compartments in the UDB. The active compartments define the set of compartments with which a user is currently functioning. The authorized compartments define the range of compartments that a user can add to his or her active set of compartments. For more information on how these values are used to determine a user's security attributes, see the *UNICOS Multilevel Security (MLS) Feature User's Guide*.

Objects are assigned security labels by the installation process, administrative procedures, or by inheriting the active security label of the subject who creates it. If an object is not explicitly assigned a security label by one of these methods, it has a security label of level 0 and null compartments by default.

A session's security attributes are set at session initiation. These attributes are determined by the information defined in the UDB plus information defined for network and other configuration options.

MAC checks are always performed before discretionary access control (DAC) checks. If either of the conditions in the first two bullets described previously are met, then DAC checks are made. Otherwise, the subject is denied access to the object.

The security administrator can use the `deflbl_as_minlbl` configuration field `/etc/config/confval` to allow minimum security labels to be defined for users. Enabling this field allows the user's default security label (that is the `deflvl` and `defcomp` fields in the user database) to become the user's minimum security label. Use of this field means users must log in with a security label that dominates their default security label.

The security administrator can also use the `mincomps` field in the UDB to define a user's minimum compartment set. See Section 8.5.3, page 206, for more information on these fields.

The security administrator can also define a set of permissions for each user in the UDB. These permissions grant special privileges to the user and are defined

in the security parameter file (`sys/secparm.h`). See Section 8.7.5.4, page 235, for more information on these permissions.

For more information on how these values are used to determine a user's security attributes, see the *UNICOS Multilevel Security (MLS) Feature User's Guide*. This manual explains the interactive login process for both UNICOS and Cray ML-Safe system configurations. For more information on setting up UDB security entries, see Section 8.7.6, page 236, for more information.

The following sections describe how the UNICOS MAC policy affects the following system operations:

- Directory operations
- File system operations
- Device labeling
- `cron` and `at` operations
- Cray ML-Safe mail
- `/proc` operations
- `syslogd` operations
- IPC objects

8.4.1 Directory Operations

You can use the `mkdir -L` command to create a directory with a security label that is different than your active security label (assuming the requested label is within your authorized range). If the relabeling fails, the directory's label remains at your active security label. If the `mkdir -L -p` command is used, only the last directory in the path is relabeled. All intermediate directories are created at your active label.

Any user can upgrade the label of an empty directory (for example, when using the `spset -c` or `spset -l` commands) if all of the following conditions are met:

- The user has MAC write access to the target directory
- The user is the owner of the target directory
- The target directory is empty

If you are properly authorized, you can override these restrictions and change the label of any directory; the definition of properly authorized depends on which system management mechanism your system is using. For more information, see the `mkdir`(1) man page in the *UNICOS User Commands Reference Manual*.

To create a directory, the security label of the directory must always fall within the security label range of the file system on which the directory resides.

8.4.1.1 Removing Files from Directories

To ensure compliance with the TCSEC object reuse requirements, the UNICOS system clears the name of the object being removed from the directory. Only the object name is cleared by this change. The remaining entries in the directory entry structure (for example, inode number, name signature, record length and name length) are not cleared.

Administrators should be aware that directories with removed objects that exist on UNICOS 8.0 MLS systems continue to have "removed" object names in them on later UNICOS releases until the directory entries are replaced with another object or the directory block is released.

8.4.1.2 Wildcard and Multilevel Directories (MLDs)

The UNICOS MLS feature uses two mechanisms for labeling directories that contain objects at different security labels: wildcard directories and multilevel directories (MLDs). The following sections explain the function and use of these directories. These two types of directories can be used exclusively or in combination on a UNICOS system. Wildcard directories should not be used on evaluated configurations of the UNICOS operating system. Cray recommends using MLDs for products that process multiple security labels (for example, `mail`, `lpr`, `lpd`, `cron`, NQS, and CRL). See Section 8.4.1.2.2, page 175, for more information.

8.4.1.2.1 Wildcard Directories

The UNICOS MLS feature uses security level 63 to indicate a wildcard directory. A wildcard directory can contain files at any security label within the boundaries of the file system. Access to files in the wildcard directory must satisfy the discretionary and mandatory access policy enforced by the UNICOS MLS feature.

You can apply security compartments to a wildcard directory, although these compartments are ignored by the system when doing a mandatory access control check.

Wildcard directories are established primarily for daemons that must service many requests and output queues; the assignment of wildcard directories should be restricted to those described in Section 8.7.7, page 238. Also, the use of wildcard directories avoids replication of special directories for every use.

The `/usr/tmp` and `/tmp` directories, which are accessible to many system utilities, must be assigned the wildcard security level (or converted to multilevel directories (MLDs), which is explained in the next section) to allow them to contain files with varying security levels.

The Network Queuing System (NQS) spooled output and data files directories must also be labeled as wildcard directories (or MLDs). The NQS directories are automatically labeled by NQS.

For more information on labeling these directories, see Section 8.7.7, page 238.

8.4.1.2.2 Multilevel Directories (MLDs)

The UNICOS system has relied upon the use of wildcard directories to hold files at multiple labels. These directories have been used for public directories like `/tmp` as well as for spool directories (for example, `/usr/spool/mqueue`).

The use of wildcard directories violates the TCSEC requirements, as they create a potential for write-down security policy violations. Wildcard directories have been replaced with MLDs.

Warning: Only the use of MLDs is allowed on the evaluated configuration of a UNICOS system. There are no checks or warnings enforced by an evaluated system to ensure only MLDs are used. This requirement must be enforced by administrative procedures. There is no configuration parameter in the UNICOS Installation and Configuration Menu System that enables MLDs.

MLDs provide a method of sharing a common directory name, while partitioning the actual directory contents according to security labels.

For example, a wildcard directory provides a single name space at all labels. Because of this, if `/tmp` is labeled as a wildcard directory, a process at level 0 and compartment set 077 can create `/tmp/file` at level 0 and compartment set 077, but a process at level 0 and compartment 0 would be unable to create `/tmp/file` at level 0 and compartment set 0, as that file already exists in the directory and is not writable by the second process.

A MLD has two parts: a root directory and a multilevel symbolic link to the root directory. A MLD provides a discrete name space for each label represented within the directory. As a result, if `/tmp` is a MLD, a process at level 0 and

compartment set 077, and a process at level 0 and compartment set 0 can each create a file known to each process as `/tmp/file`. The result is two different files at two different labels containing completely different data.

Warning: Because of the nature of the symbolic link expansion, MLDs do not work across NFS mount points. It is not recommended to use MLDs on NFS mounted file systems.

The following list contains the Cray products or commands that use MLDs when running on a Cray ML-Safe system configuration. The mail directories require the use of MLDs on either a UNICOS or a Cray ML-Safe system configuration:

- `jtmp` directories
- `/tmp` directory
- `/usr/tmp` directory
- `cron`(8) and `at`(1) spool directories
- `lpr`(1) and `lpd`(8) spool directories
- Mail directories (`/usr/mail` and `/usr/spool/mqueue`), plus the user's directories that are used to save mail messages.
- NQS spool directories
- CRL debug log directory as defined by `${RLLOGDIR}`

For sites that have users running at multiple labels, `usr/preserve/`*LOGIN_NAME* must be created as an MLD or a wildcard directory.

8.4.1.2.3 Creating a MLD

The following information on MLDs assumes you have read and understood the MLD section in the *UNICOS Multilevel Security (MLS) Feature User's Guide.*

There are several ways to create a MLD:

- Create a new MLD
- Customize the directory by first creating it and then creating a link to it
- Convert an existing directory structure to a MLD

To create a new MLD, use the `mlmkdir`(8) command. This command creates the directory and the symbolic link, labeling both with a mode of `rwxrwxrwx`.

This labeling scheme is convenient if you want to create a directory that is available to public use or when creating a directory for users (for example, a user's mailbox directory). A more appropriate mode should be assigned after creating the directory.

The `mlmkdir` command creates the multilevel symbolic link with the name specified on the command line and adds the `.mld` suffix to the name to create the directory. This naming convention is convenient, but not required, as shown in the following example.

The following example shows how to use the `mlmkdir` command. The `ls -l` command is used in this example to show the result of the operation; for ease of use, the output from the `ls` command has been edited to remove the link counts, owners, groups, and creation time:

```
$ setucat secadm
secadm$ pwd
/home/user
secadm$ mlmkdir /home/user/mld
secadm$ ls -l
total 32
drwxr-xr-x  .
drwxr-xr-x  ..
lrwxrwxrwx mld -> /home/user/mld.mld
drwxrwxrwx mld.mld
secadm$ cd mld
secadm$ /bin/pwd
/home/user/mld.mld/000
secadm$ cd ..
secadm$ /bin/pwd
/home/user
secadm$ setucat 0
$
```

If you do not want to use the `.mld` suffix for the directory name, or you want to create a multilevel symbolic link to an existing directory, you can use the `-m` option of the `ln`(1) command. This option works like the `-s` option of the `ln` command, but instead of creating a regular symbolic link, it creates a multilevel symbolic link.

The following example shows how to change the `.mld` suffix to a `.dir` suffix. In this example, the directory is created by using the `mkdir`(1) command and then the multilevel symbolic link is created using the `ln` command:

```
$ setucat secadm
secadm$ pwd
/home/user
secadm$ mkdir mymld.dir
secadm$ ln -m /home/user/mymld.dir mymld
secadm$ ls -l
total 48
drwxr-xr-x  .
drwxr-xr-x  ..
lrwxrwxrwx mld -> /home/user/mld.mld
drwxrwxrwx mld.mld
lrwxrwxrwx mymld -> /home/user/mymld.dir
drwxr-xr-x mymld.dir
secadm$ cd mymld
secadm$ /bin/pwd
/home/user/mymld.dir/000
secadm$ cd ..
secadm$ /bin/pwd
/home/user
secadm$ setucat 0
$
```

To remove a MLD, you can use the `mlrmdir`(8) command. This assumes that the directory is empty (that is, only the root directory and labeled subdirectories are present, with no files or directories created by users). The following example removes both directories created in the two previous examples. Use of this command removes both the multilevel symbolic link and the directory:

```
$ setucat secadm
secadm$ pwd
/home/user
secadm$ mlrmdir mld mymld
secadm$ ls -l
total 16
drwxr-xr-x  .
drwxr-xr-x  ..
secadm$ setucat 0
$
```

The `cvtmldir`(8) command provides the ability to convert between wildcard directory and MLD structures.

The `cvtmldir` command allows an administrator to convert a wildcard directory to a MLD, while preserving all files and directory tree structure, and placing each file correctly in the multilevel directory tree.

The `cvtmldir` command also allows an administrator to convert a multilevel directory to a wildcard labeled directory, but the conversion may not be perfect, as file name collisions can result.

To use `cvtmldir` on a `PRIV_SU` system, you must be the super user. To use `cvtmldir` on a system using only the privilege mechanism, you must have an active `system` category.

On a Cray ML-Safe system configuration, the only way to obtain an active `system` category is to bring the system to single-user mode. On systems that use the PAL-based privilege mechanism, but are not strict Cray ML-Safe environments, it is possible that a user be assigned the `system` category in a multiuser state.

The following sections provide examples of converting directories.

8.4.1.2.4 Converting from Wildcard Directory to MLD

Conversion from a wildcard directory to a MLD can be done by one of the two following procedures:

- As a copy from one directory tree to another
- As a conversion "in-place," using the source directory as the root of the directory tree that is created for the destination

In either case, `cvtmldir` does not change the source directory structure. This avoids loss of data in the event of an unsuccessful conversion. Once conversion completes successfully, you must remove the source directory tree. In either case, `cvtmldir` uses file system links whenever possible to reduce the amount of storage needed to perform the conversion.

8.4.1.2.5 Conversion by Copying

If the directory to be converted is not the root of a mounted file system, the simplest way to convert it is by copying from the original directory structure to a new directory structure. In this case, to clean up after the conversion, remove the old directory structure.

For this conversion method, rename the source directory and convert the renamed source directory into a directory with the original name. This reduces

the probability that the source directory changes during the conversion and there is no need to rename the destination directory.

The following example shows the conversion of /usr/spool/mqueue on a system with the PRIV_SU configuration option enabled. This example assumes that /usr/spool/mqueue is not the root of a mounted file system and the file system on which /usr/spool/mqueue resides does not support the syslow label.

```
mv /usr/spool/mqueue /usr/spool/mqueue.old
cvtmldir -m /usr/spool/mqueue.old /usr/spool/mqueue
rm -rf /usr/spool/mqueue.old
spset -l 0 /usr/spool/mqueue.mld
```

If your file systems support the syslow security label, use the spset -l syslow /usr/spool/mqueue.mld command instead of the spset command sequence shown in the previous example.

The following example shows the conversion of /usr/spool/mqueue on a system using only the privilege mechanism (again, this example assumes that /usr/spool/mqueue is not the root of a mounted file system):

```
setucat system
mv /usr/spool/mqueue /usr/spool/mqueue.old
cvtmldir -m /usr/spool/mqueue.old /usr/spool/mqueue
spset -l syslow /usr/spool/mqueue.mld
setucat 0
```

When the conversion is completed, you can remove the /usr/spool/mqueue.old directory.

In the previous example, the setucat system command activates the system category. If this category is already active, ignore this step. The setucat 0 command deactivates the system category. If you are doing other tasks that require the system category immediately after this task, ignore this step.

In all conversion examples in the following sections, the procedures are shown for a system with PRIV_SU enabled. Add the setucat commands when necessary to use the procedures on a system with the PAL-based privilege mechanism.

8.4.1.2.6 Converting In-place

If you want to convert the mount point of a file system into a MLD, you cannot convert by copying as shown in the previous section, as there is no way to make

the copy become the root of the file system. To make this type of conversion, you need to convert the directory "in-place." The in-place conversion of a wildcard directory results in the entire wildcard directory structure being located in `/tmp.mld`.

The most reliable way to successfully complete this type of conversion is to do it in single-user mode. This is necessary to unmount and remount the new directory, as `/tmp` is usually busy on a multiuser system.

The `/dev/dsk/tmp` device is used in the following example; if the device containing the `/tmp` file system has a different name on your system, use that device.

The following example shows the conversion of the `/tmp` directory from a wildcard directory to a MLD.

```
umount /dev/dsk/tmp
mv /tmp /tmp.mld
mount /dev/dsk/tmp /tmp.mld
cvtmldir -f -m /tmp.mld /tmp
spset -l syslow /tmp.mld
```

When the conversion is completed, `/tmp.mld` contains a set of directories, each of which has a name of octal digits that begins with a zero. Each of these directories is a labeled subdirectory; the name of each directory is an octal value with the first three characters representing the security level and all following characters representing the octal compartment mask (this naming convention is explained later).

Before `/tmp` can be used as a MLD, remove the entries that are not names of labeled subdirectories from `/tmp.mld`, as follows:

1. Remove all entries that have nonoctal characters in their names or entries that do not consist of at least three octal characters starting with 0, as shown in the following example:

   ```
   cd /tmp.mld
   ls -l | grep -v "0[0-7][0-7]*" | xargs rm -rf
   ```

2. Execute the `ls -l` command. Remove any entry that is not a directory.

3. Examine each remaining directory to determine whether the name of the directory matches the label. To do this, use the `spget`(1) command to obtain the level and compartments of each directory and convert the level value to octal using the following table:

Label	Octal representation
0	000
1	001
2	002
3	003
4	004
5	005
6	006
7	007
8	010
9	011
10	012
11	013
12	014
13	015
14	016
15	017
16	020
51	063 (`syslow`)
54	066 (`syshigh`)
63	077 (wildcard)

Note: The naming convention for MLDs outlined in this section may change in future UNICOS releases.

After converting the level to octal, delete the leading zero on the compartment bit mask and put the two octal values together as follows:

```
<octal level><compartment bit mask>
```

For example, if a directory has a level of 14 and a compartment set of 0705, its name as a labeled subdirectory would be `016705`. If an existing directory name does not match the name you have determined, then it is not a labeled subdirectory and should be removed.

8.4.1.2.7 Converting from MLD to Wildcard Directory

Warning: Only MLDs are supported on a Cray ML-Safe system configuration.

Converting from a MLD to a wildcard directory can result in name collisions. The `cvtmldir`(8) command renames the colliding file and reports both the old and new name to the administrator. The administrator should tell the affected users that the files are renamed.

Conversion from a MLD to a wildcard directory can be done by one of the two following procedures:

- As a copy from one directory tree to another
- As a conversion "in-place," using the source directory as the root of the directory tree that is created for the destination

The following example shows how to convert a directory that is not the root of a file system to a wildcard directory:

```
rm /usr/spool/mqueue
cvtmldir -w /usr/spool/mqueue.mld /usr/spool/mqueue
```

Successful completion of this example results in a wildcard directory tree in `/usr/spool/mqueue` and a MLD tree in `/usr/spool/mqueue.mld`. Remove the multilevel directory tree as follows:

```
rm /usr/spool/mqueue.mld
```

To convert a MLD that is the root of a file system to a wildcard directory, you must convert it in-place. You may find it useful to know what labeled subdirectories are in the MLD before starting the conversion.

Do this by changing to the directory containing the MLD structure (usually `<pathname>.mld`) and executing the `ls`(1) command to obtain a list of all directories in the root of the MLD. You may want to save the output from the `ls` command execution in a file for future reference.

The following example shows the in-place conversion of /tmp from a MLD to a wildcard directory:

```
cd /tmp.mld
ls
cd /
rm /tmp
cvtmldir -f -w /tmp.mld /tmp.mld
umount /tmp.mld
mv /tmp.mld /tmp
mount /tmp
```

Successful execution of this conversion results in a wildcard directory representation of the MLD structure, although there may be residual labeled subdirectories. These are the directories in the list obtained from ls before the conversion. Remove these directories.

8.4.1.3 Directory Permissions

Regardless of whether the UNICOS MLS feature is enabled, directories such as /bin and /etc should not have public write access permission assigned to them. If public write permission exists on such a directory, a user could conceivably replace an existing command with a modified version with the same name, thus introducing a Trojan horse.

A suitable access mode for these directories is 755, which specifies public read access. Read permission allows users to read a directory as a file, discovering all the names it contains. For similar reasons, a user's home directory and .profile (see sh(1)) or .login and .cshrc (see csh(1)) files should be owned by that user, and write access should be restricted to the file owner.

8.4.2 File System and File Operations

The following sections explain the labeling and use of file systems and files on a UNICOS system by providing the following information:

- System high and system low labels
- File system labeling
- File system access control
- File system backup
- File system security

- File labeling

8.4.2.1 System High and System Low Labels

The ability to protect system files from unauthorized access or modification was provided on pre-8.0 MLS systems by discretionary access controls (DAC) only. In order to support the UNICOS security policy and TCSEC criteria, the UNICOS system now uses the system high (`syshigh`) and system low (`syslow`) security labels.

Only a properly authorized user can override these labels to modify, read, or write to a system file. The `syshigh` and `syslow` labels do not fall within the range of labels used by the nonadministrative user population on a UNICOS system.

The `syshigh` label is assigned to system-private databases, such as the user database (UDB), audit logs, and administrator-only binaries. The `syshigh` label is not dominated by any user label. This means that system files protected by the `syshigh` label cannot be read or written to by an unauthorized user.

The `syslow` label is assigned to the majority of binaries, public databases, and public directories. The `syslow` label is dominated by all user labels, but is equal to no user label. This means that system files protected by the `syslow` label can be read, but not written to by an unauthorized user.

A security administrator has the capability to use a privileged shell in order to set his or her label to the `syshigh` or `syslow` labels in order to do any necessary administrative work. See Section 8.2.4, page 166, for more information.

The MLS feature provides a default set of security labels for system files. See `/etc/privdb` for more information.

The `SECURE_MAC` configuration parameter indicates the following to system commands and daemons:

- That UNICOS file system label ranges (including `/tmp` and `/usr/tmp`) have been updated to include the `syshigh` and `syslow` labels.
- That administrative procedures have been established to adequately manage files with `syshigh` and `syslow` labels.

This parameter is set by using the `Configure system->Multilevel security (MLS) configuration->System options->Enforce system high/low security labels?` selection in the UNICOS Installation and Configuration Menu System.

Once the SECURE_MAC parameter has been enabled and the new configuration has been activated through the installation tool, a new kernel must be built and booted in order for the parameter to become effective.

Warning: The SECURE_MAC parameter is intended for use on UNICOS systems using the PAL-based privilege mechanism.

You can enable the SECURE_MAC parameter on systems with PRIV_SU enabled. However, administrative commands that depend on set-group-ID (setgid) functionality (instead of the root user ID) to access protected devices do not have the authority to override device label protections as required. A site can convert setgid commands to use the PAL-based privilege mechanism.

A command or daemon that needs to create syshigh or syslow labels initially has to invoke sysconf(2) to determine if it should manipulate its active label throughout its execution to create and/or manage the syshigh and syslow labels.

If the SECURE_MAC configuration option is not enabled, the calling process manages file labels as was done on UNICOS 7.0 MLS systems.

The following library routines support system labeling. Man pages for these routines can be found in the *UNICOS System Libraries Reference Manual*.

Routine	Description
mls_create(3)	Allocates and creates an opaque security label for use during security label comparisons.
mls_extract(3)	Extracts a label from an opaque security structure.
mls_import(3)	Converts the text representation of the security label to the internal representation.
mls_export(3)	Converts internal security label to text representation.
mls_free(3)	Frees security label storage space.
mls_dominate(3)	Performs a security label domination test.
mls_equal(3)	Performs a security label equality test.
mls_glb(3)	Computes the greatest lower bound.
mls_lub(3)	Computes the least upper bound.

Note: Local (site) code that now performs security label comparisons should be modified to use the mls_equal and mls_dominate library routines, rather than attempting to perform their own label comparison.

8.4.2.2 File System Labeling

You can use the `mkfs -L` and `mkfs -U` commands to define the minimum and maximum security levels, respectively, and the `-C` option to define the authorized compartments of a new file system. The defined values and a security label are then written to the file system, as shown in the following example:

```
/etc/mkfs -L 0 -U 5 -C 0377 /dev/dsk/usa
```

If these options are not specified, the minimum and maximum security levels of the file system default to 0 and no valid compartments are specified. This means that files assigned any other security label cannot be written to the file system. A file outside the range of the file system's security label cannot be written to the file system.

The `RC_SECLOW`, `RC_SECHIGH`, and `RC_SECMASK` configuration parameters allow an administrator to assign a label range (minimum security level, maximum security level, and compartment mask, respectively) to the `/tmp` and `/usr/tmp` file systems when the `mkfs` command is used during a reboot process. If you want to support the use of the `syshigh` and `syslow` security labels on these file systems, the `SECURE_MAC` parameter must be enabled. These parameters can be set by using the following selections in the UNICOS Installation and Configuration Menu System:

- For `RC_SECLOW`, use the `Configure system->Multilevel security (MLS) configuration->System options->/tmp and /usr/tmp minimum security level` selection
- For `RC_SECHIGH`, use the `Configure system->Multilevel security (MLS) configuration->System options->/tmp and /usr/tmp maximum security level` selection
- For `RC_SECMASK`, use the `Configure system->Multilevel security (MLS) configuration->System options->/tmp and /usr/tmp compartment mask (octal)` selection

The `-u`, `-l`, and `-c` options of the `labelit`(8) command can write an upper security level, a lower security level, and authorized compartments, respectively, to an existing mounted or unmounted file system. The `-s` option is used when installing the UNICOS MLS feature. It sets a security label on a nonsecure file system.

When used with the `-c` option, the `-s` option can change the authorized compartment set of a file after the compartments have been initially set. This includes the ability to remove as well as add compartments. When used with the `-l` or `-u` options, the `-s` option increases the minimum security level or

decreases the maximum security level of a file system after these values have been initially set. You must be properly authorized to use the `-s` option for these purposes.

The following is an example of using the `labelit` command:

```
/etc/labelit -u 5 -l 0 -c 0377 /dev/dsk/usa
```

Labeling a file system with the `labelit -c` command can be done any time before it is mounted; the UNICOS MLS feature does not have to be enabled to do so.

8.4.2.3 Changing File Labels

You can use the `-c`, `-l`, and `-k` options of the `spset` command to set the security compartments, security level, and flags, respectively, on files.

You must be properly authorized to use these options. This means you must have the following security attributes:

- On a system with `PRIV_SU` enabled, you must be the super user
- On a system using the PAL-based privilege mechanism, you must have an active `secadm` category.

8.4.2.4 File System Access Controls

On a UNICOS system, the following condition must be met to successfully mount a file system:

- If the `Configure System->Multilevel Security (MLS) Configuration->System Options->Restrict file system labels to system range?` configuration option is turned on, then the minimum and maximum security levels (with the exception of the `syslow` and/or `syshigh` security levels) and the authorized compartments of a file system must fall within the authorized ranges of the UNICOS system; otherwise the `mount` request fails.

To mount a file system on a UNICOS system with `DEV_ENFORCE_ON` set to `ON`, the device on which the file system resides must have one of the following sets of attributes:

- It must be in the `OFF` state.
- It must be in the `ON` state with the `mldev` flag on. It must also have a minimum and maximum security level range that encompasses the range

of the file system and have an authorized set of security compartments that contains all the valid compartments of the file system.

Whenever the kernel is requested to assign an inode from the inode free list, the following checks are applied to the relevant file system:

- The file system is checked for a security label.
- The file system is checked to ensure that its maximum and minimum security levels and compartments bound the new file's security label.

If any of these security checks fail, the system call requiring the allocation of the inode fails, and the inode is not allocated.

8.4.2.5 File System Back up Operations

Enabling the UNICOS MLS feature does not alter the need for sound security practices such as file system back-up operations. All backups to tape should be stored in a secure area.

The `cpio`(1) command provide back-up operations for a file system; it processes files for the active security label of the user. On UNICOS systems using the PAL-based privilege mechanism or have the `PRIV_SU` configuration option enabled, the `cpio` command allows any user to archive data to a single-level medium. An appropriately authorized user can archive data to a multilevel medium.

Warning: Authorized users can override MAC and DAC restrictions when using the `cpio` command. Special care must be taken to ensure that data is not inadvertently downgraded.

This may occur if you do not use the `-M` and `-z` options of `cpio`. Also, when using the `-d` option, the directory is created with the label of the last object processed. If no object has been processed, the directory is created at the label of the person invoking `cpio`.

In addition, `cpio` supports the following functions:

- Allows authorized users to restore all security attributes from a multilevel medium.
- Provides the mechanism necessary to archive privilege assignment list (PAL) information in the `cpio`(5) archives.
- Minimizes the chance that security attributes in a `cpio` archive can be altered or fabricated by a non Cray ML-Safe process.

- Minimizes the risk that a multilevel `cpio` archive created by an authorized user can be read by an unauthorized user who does not have at least MAC read access to every file in the archive.

These changes to `cpio` cause the following migration issues:

- `cpio` can only restore security attributes from archives on a multilevel medium.
- For archives created with the `-z` option, redirection of `cpio` output and input must be done in a privileged shell.
- For archives created with the `-z` option, pipes connecting `cpio` output and input must be labeled as multilevel.

In addition, access control lists (ACLs) are preserved if the user is the owner of the file and has MAC write access, or is a security administrator.

The following rules must be observed when using `cpio`:

- For `cpio`, your active security label must dominate the security label of the files being copied. An appropriately authorized user can copy any file.
- To dump files, your active category must be a superset of the categories of the file.
- ACLs are preserved only if the user is the owner of the file, has an active `secadm` category, or has write access to the file.

On a UNICOS system use the `cpio -z` command to make a secure copy of a file. This option must be used when you copy in (`-i`), copy out (`-o`), or pass (`-p`) a file. In addition, the `-x` (excludes copying or preservation of ACLs), `-M` (preserving all security attributes), and `-P` (excludes copying or preservation of PAL information) options are valid only when used with the `-z` option on a system that has MLS enabled.

The `dump`(8) and `restore`(8) commands allow a security administrator to process a subset of or an entire file system with files at various security levels and compartments. These commands also dump and restore ACLs applied to files and directories.

Caution: You should take special care to ensure that dump files cannot be modified or accessed by unauthorized users.

8.4.2.6 File System Security

The system or security administrator should perform the following tasks to ensure file and file system security:

- Protect memory access; there should be no public access to `/dev/kmem` (see `mem`(4)) and `/dev/mem`. Permitting public access would give users access to information not belonging to them.
- Protect access to devices; there should be no public access to the raw disk devices used for storing user files. If public access were permitted, it would be possible for a user to read from the inode list, locate the position of any information on the disk, and read it. Therefore, all entries in the `/dev` directory that pertain to disks should be owned by `root` and assigned an access mode of 600.

  On systems that have `DEV_ENFORCE_ON` enabled, raw disk devices containing file systems are protected from public access in one of the following ways:

  – By default, they are in the `OFF` state, which means only privileged processes can access them.

  – If they are in the `ON` state, they can only be mounted as file systems if they have the `mldev` flag set. This means only privileged processes can access them.

  Caution should be used to preserve these attributes when manipulating the labels of raw disk devices. For example, when changing a disk device from `OFF` to `ON`, ensure that you change it from single-level to multilevel at the same time (or before you change the state of the device).
- Protect public access to terminals. On UNICOS systems without `PRIV_SU` enabled, the default mode and access control lists (ACLs) on terminals prohibit access to the terminal by anyone but the owner. Utilities like `write`(1) override this restriction to allow interaction between users. User can use the `mesg`(1) command to allow access to their terminal through the `write` command. `write` filters its input to avoid transmission of nonprintable characters that can be interpreted as escape sequences by the recipient's terminal.

  On UNICOS non-MLS systems or systems with `PRIV_SU` enabled, this protection is not available. Users should be advised to set the mode on their terminals to 600 if they want to prevent others from writing directly to their terminals. On these systems, setting a mode of 600 disables the `write` command on that terminal and prevents direct writes to the terminal.

- Protect access to source code; generally, UNICOS source code should not be available online at customer sites. To protect UNICOS source code and system logic, the code should be assigned the `system` category.

  On systems that support the use of the `syshigh` label, this label should be used instead of the `unicos` or `system` categories to protect source code.

- Keep your file system backups in a physically secure area.
- Turn on accounting and security logging.
- Educate users on the use of UNIX file permissions, UNICOS security levels and compartments, and the `-x` option (encryption mode) of the `vi`(1), `ed`(1), or `ex`(1) commands.

In addition, a security administrator can assign one of the two following flags to a file (but not to directories) so that read or write access to that file is logged in the security log:

- `trapr`
- `trapw`

Both flags trap read and write accesses. That is, if you assign the `trapr` flag to a file, both read and write accesses to that file are logged; if you assign `trapw` to the file, the same thing happens. The functions of the flags may be split in future releases.

Use the `-k` option of the `spset`(1) command, as shown in the following example, to set these flags; you must have an active `secadm` category to set them:

```
spset -k trapr file1
spset -k trapw file2
```

8.4.2.7 File Labeling

Regular files, named pipes, and sockets are assigned the active security level and active compartments of the creating subject. This information is recorded in both the memory and disk versions of the inode describing the object.

Block and character special files are created with a security label set to 0 and in the `OFF` state. The administrator must change this label to reflect the nature of the information available through the device at any given time. Use the `spdev`(8) command to label the device.

Only a properly authorized user can raise or lower the security level of a file. The security administrator can assign a security label to a file as long as the label being assigned is within the authorized range of the security administrator.

A regular file, block or character special file, or named pipe can be removed or unlinked only by a subject with the same security level and compartments as those of the object.

All nondirectory files created within a directory must have a security level and compartments equal to the security level and compartments of the directory.

See the *UNICOS Multilevel Security (MLS) Feature User's Guide* for more information on creating and using files on a UNICOS system.

8.4.3 Single-level and Multilevel Files and Devices

On UNICOS systems with the MLS feature, the concept of multilevel devices has been extended to files. As a result, it is possible to have single-level and multilevel files as a general concept, and single-level and multilevel devices when rules apply to special devices.

Whether a file is single-level or multilevel depends on the nature of the data to be stored in the file. If the data in the file can be protected by a single, externally-applied label, then the file should be made a single-level file. This is the case for most files on a UNICOS system.

If the data in the file contains internal labeling information that describes the classification of specific portions of the data, the file should be made a multilevel file. An example of this type of file is the raw disk device file for a file system that contains inodes that contain labels that reflect the classifications of files.

The information stored within a single-level file is all at one security label, which is the active security label on the inode. Because the kernel knows this label, it enforces the normal MAC rules regarding access to the file and allows nonprivileged processes to gain access subject to mandatory and discretionary access controls.

The active label on a single-level file reflects the nature of the data in the file. If the file is a device special file, the data in the file is the data currently accessible through the device (for example, the data on a currently-mounted tape volume). If the file is a regular or FIFO file, the data in the file is the data that was placed there by the creator and subsequent writers to that file.

In addition to the active label, a file has a label range. This label range determines what the legal values for the active label on the file. If the file is a device special

file, the label range is set by the administrator and reflects the physical security of the device, as well as any applicable site security policy issues relating to the use of the device.

If the file is a regular or FIFO file, the label range is set by the kernel and reflects the label range of the file system on which the file resides. Regardless of the type of file, the active label of the file can never be set outside the label range of the file.

On a UNICOS system, most regular and FIFO files are single-level files. An example of a single-level device special file is the tape pseudo device. This is provided by the tape daemon to a nonadministrative user when it is asked to manage tape data.

The information stored on a multilevel file is at different security labels. Each chunk of information is associated with a label embedded within the data on the file. The labels on data within a multilevel file are managed by the software that places the data in the file.

Each data object represented within a multilevel file has its own active label, so the active label of the file itself has little meaning. The label range on the file does dictates the range of differently-labeled objects that can be placed in the file. As with single-level files, the label range on a device special file is set by the administrator, while the range of a regular or FIFO file is set by the kernel, based on the file system label range.

Because the data contained in a multilevel device is at more than one label, and the labels can only be managed by the software that put them there, the normal MAC rules cannot be used to control access to multilevel files. Instead, access to multilevel files is only granted to privileged processes. Enforcement of this restriction is controlled by the DEV_ENFORCE_ON configuration parameter, which is explained later in this section.

An example of a multilevel file is a dump archive. It contains an archived file system, which can be shipped to a disk or across a network. An example of a multilevel device is a file system.

> **Note:** Devices that are used as multilevel devices, but are physically secure as the system itself, can be left in the OFF state to signify that they can handle data from privileged processes at any label allowed by the system. This is a convenient way to handle disk devices that are well-secured and will contain file systems. The rules for access to devices in the OFF state are the same as for access to multilevel devices. Files cannot be placed in the OFF state, so they must be labeled as multilevel if they are used for multilevel data.

To make a file multilevel, you must enable the multilevel security flag (mldev), as shown in the following example:

```
spdev -m filename
```

In the previous example, the -m option of the spdev(8) command sets the mldev and secdv flag (the spdev and state flags are explained in the next section) and preserves all other flags with their current values. To enable the device (that is, set the state flag), use the -s option as shown in the following example:

```
spdev -m -s filename
```

The following example shows how to use the -k option of the spset(1) command to set the mldev flag and preserve the current value of the secdv. The spset command cannot be used to set the secdv flag.

```
spset -k mlsdev,secdv filename
```

The following example shows how to set or preserve the state flag (which is described in the next section) on a device file:

```
spdev -k mldev,state,secdv filename
```

Once a file is made multilevel, you can make it single-level by using the spset command to specify every flag that is currently set, except for the mldev flag, as shown in the following example. This example makes the file single-level, but preserves the value of the secdv flag:

```
spset -k secdv filename
```

The -k option of the spset command can be used to set or preserve flags (for example, the trap flags). See the spset man page in the *UNICOS Administrator Commands Reference Manual* for more information.

8.4.3.1 Assignment and Access Rules for Labeling Information

The previous section addressed only the multilevel and single-level flag (mldev), the active label, and the label range of files. Two other flags also control access and labeling of devices only. They are the secure device flag (secdv) and the state flag (state). The following paragraphs describe the rules for assigning labels, label ranges, and flags to devices and files:

- If the secdv flag is 0, the state flag must be off (that is secdv must be on in order to enable state).
- If secdv is 0, the label is invalid.

- If the state flag is off on a device, access is restricted to privileged processes only.
- If mldev is set to 1, the file is multilevel. Only a privileged process can access it. If mldev is set to 0, the file is single-level. It is accessible to users subject to normal MAC and DAC rules.
- The minimum and maximum label ranges of nondevice inodes are always equal to the label range of the file system on which they reside.
- The active label on a single-level inode are always within the label range of the inode.
- When mounting a file system, if the device on which the file system resides is enabled (that is, state is 1), the device must be multilevel (mldev is 1), and must have a label range that encompasses the label range of the file system.

The setdevs(2) system call is allowed to set the secdv flag to 1 or 0. The setfflg(2) system call is only allowed to set it to 0.

If DEV_ENFORCE_ON is enabled, system calls that check for MAC write access to file inodes fail if the user is not authorized and the inode is in one of the following states:

- It is a device inode and the state flag is not on.
- The file is multilevel (mldev is set to 1).

To set the DEV_ENFORCE_ON parameter use the Configure system -> ->Multilevel security (MLS) configuration->System options->Enforce strict device labeling rules? selection in the UNICOS Installation and Configuration Menu System. The default setting is OFF; no change in the way devices are handled should occur when set to OFF. This parameter must be on for a Cray ML-Safe system configuration.

Warning: The DEV_ENFORCE_ON parameter is intended for use on UNICOS systems using the PAL-based privilege mechanism.

You can enable the DEV_ENFORCE_ON parameter on systems with PRIV_SU enabled. However, administrative commands that depend on set-group-ID (setgid) functionality (instead of the root user ID) to access protected devices do not have the authority to override device label protections as required. A site can convert setgid commands to use the PAL-based privilege mechanism.

Enabling this parameter means that devices must be labeled before they are made available to nonprivileged users and that access to the devices is subject to the restrictions described previously.

If DEV_ENFORCE_ON is disabled, device labeling is enforced according to the UNICOS 7.0 implementation, although the pty behavior described in Section 8.4.3.3, page 197, is available.

Physical disk devices are always left off to ensure that only privileged processes can gain access to them. Devices constructed of multiple physical devices or logical devices residing on pieces of physical devices are used as the entry points to physical devices for normal system operations.

8.4.3.2 The spdev Command

To set file and device labels, ranges, and labeling related flags, use the spdev(8) command. For complete information, see the spdev(8) man page. The spdev command uses the following options:

- The -L and -K options specify the active level and compartments, respectively, on a single-level device. These options are ignored for multilevel devices, since the kernel forces the active label on a multilevel device to the maximum label.
- With the exception of the -C (clearing all device labeling information on the device) and the -p option (printing all device labeling information on the device), all operations through the spdev command are incremental. That means an administrator can set the label range, return and set the active label, and return again to turn the device on without having to specify all the other options for each command invocation.
- Multiple file names on the command line are allowed and apply the requested operations to each in turn.
- All user-level checks for security administrator category or root are removed. All operations that can be requested by spdev are mediated by the kernel.

The UNICOS Installation and Configuration Menu System allows the setting of minimum and maximum levels, and the authorized compartments for tape devices. Use the Minimum security level for this group:, Maximum security level for this group:, and Maximum compartments for this group: selections in the Configure system->Tape configuration->configure tape resource group(s) menu in the UNICOS Installation and Configuration Menu System.

8.4.3.3 Pseudo Terminals

For pseudo terminals (ptys), the following rules apply:

- The pty label is set by the first user who opens it. The kernel labels the pty (instead of `login`(1)).
- The master and slave labels are kept in sync by the kernel.
- A pty label automatically changes when a nonprivileged process issues a `setulvl`(2) or `setucmp`(2) system call. When a privileged process issues a `setulvl` or `setucmp` system call, the pty label does not change.

The `setusrv`(2) system call sets the label and range on a pty if executed by a privileged process.

8.4.3.4 Pty Device Inodes

Pseudo terminal (pty) devices provide the terminal connection emulation for a UNICOS login session. Access to a pty must be carefully controlled on a Cray ML-Safe system configuration to prevent avenues of attack. The following two avenues of attack are known to exist on a Cray ML-Safe system configuration:

- The ability of a process with write access to a pty slave device special file to subvert the physical hardware or the terminal emulation software on the other end of a login connection by using escape sequences.
- The ability of a process with read or write access to a pty slave device special file different from the one currently in use as a controlling tty of a login session to force a device driver close on the pty, thereby closing the connection.

The first attack allows a user to cause another user to issue commands without knowing it. The second attack, under certain circumstances, can result in a login session remaining active on a pty that becomes available for another login as well. This can result in unauthenticated access to the system through the previous, still active, login session.

A Cray ML-Safe system configuration ensures proper protection of pty slave device special files in the `dev` directory of the running `root` file system. However, it cannot protect access to the pty device through device special files outside the `/dev` directory of the running `root` file system.

Warning: The following information must be observed for a Cray ML-Safe system configuration.

To ensure proper protection of pty devices, do the following:

- Administrators must not create pty slave device special files outside the `/dev` directory of a `root` file system.

- When an alternative `root` file system is mounted, mount it beneath a directory that prevents search access by nonadministrative personnel. A suggested way to do this is to have all the mount points for alternative `root` file systems in a single directory that is owned by an administrative user, has an administrative owning group, and has a mode that allows search access only to the owner and the owning group. An access control list (ACL) or a `syshigh` security label can also be used to enforce this restriction.

8.4.4 `cron`, `batch`, and `at` Operations

For the `cron`(8) command to work with multiple labels on UNICOS systems, the `/usr/spool/crontabs` and `/usr/spool/atjobs` directories should be converted to multilevel directories (MLDs). In order for `cron` to process nonzero labeled requests, the `cron` daemon directories must also be converted to MLDs.

This allows users to have `crontab` files and one-time batch jobs at more than one label and run successfully. The label at which the job runs is the active label at the time the job was submitted with the `at`(1), `batch`(1), or `crontab`(1) commands.

If the label at which the job runs is not valid (that is, the legal range in the user database (UDB) and the label at which the job runs is no longer valid, the job at the invalid label is deleted. The `mail`(1) command attempts to send this deletion notification. Mail is received at the new security label if that label dominates the older, invalid security label.

If MLDs are used, users of `at`(1),) `batch`(1), and `crontab` can use the full functionality of these commands on a UNICOS system. See Section 8.4.1.2.2, page 175, for more information on converting to or creating a MLD.

8.4.5 Multilevel Mail Operations

The `mail`(1) and `mailx`(1) commands support a multilevel mail capability. This capability is needed so that users can communicate through electronic mail without violating the UNICOS security policy.

In general, mail is delivered only to a user if the label of the sender is dominated by the maximum label of the recipient. The exception is when mail is sent with a `syshigh` label. Mail with this label is usually the output of administrative batch jobs or information sent to administrators by system daemons. It must be delivered, but it is never dominated by the maximum label of the recipient, as no user can log in with a label of `syshigh`.

Mail delivered at `syshigh` is delivered to the recipient on the local machine (unless it is addressed off of the machine by the sender. In this case, attempts are

made to deliver it to the target machine. These attempts may not succeed; this depends on your network configuration).

The `.forward` file of the recipient is not processed when mail with a `syshigh` label is delivered, but the `/usr/lib/aliases` file is processed. When configuring the `/usr/lib/aliases` file, ensure that actions taken when delivering mail run safely at `syshigh` regardless of what user is receiving the mail with a `syshigh` label. Programs that permit access to or change data not directly relating to the delivery of mail should not be used in the `/usr/lib/aliases` file, nor should user-supplied programs or programs subject to change by users be allowed.

Warning: Programs within the `/usr/lib/aliases` file that are not part of the set of Cray ML-Safe components should not be used on a Cray ML-Safe system configuration. Even when using commands in the set of Cray ML-Safe components, avoid commands and command line arguments that can compromise the security policies if executed on behalf of a nontrusted recipient at the `syshigh` label.

See the *TCP/IP Network User's Guide* for more information on the user interface changes for trusted mail.

In addition, the following administrative changes should be made to ensure the proper execution of multilevel mail:

- Your site may want to restrict the mail files from which a user can get mail-received announcements. To do this, remove the `PRIV_MAC_READ` privilege from the privilege assignment lists (PALs) for `mail` and `mailx`.
- To allow users to save mail messages at different security labels, the user's directory that contains the saved mail message file should be converted to or created as a multilevel directory (MLD).
- The directories containing the system mail box (`/usr/mail`) and the directory used to spool queued outgoing mail (`/usr/spool/mqueue`) should be converted to or created as a MLD.

See Section 8.4.1.2.2, page 175, for more information on converting to or creating a MLD.

8.4.6 The `/proc` File System Operations

When using the `/proc` file system on a UNICOS system, the following rules are observed:

- Mediation changes make it possible for a different `errno` to be returned. The difference occurs if there are multiple errors. For example, if discretionary access controls (DAC) are checked before mandatory access controls (MAC), and a process has neither DAC or MAC access, a MAC violation error is returned instead of a DAC violation error as before.
- Processes can only display files on the `/proc` file system that their label dominates.
- Cray ML-Safe processes can override MAC write restrictions.

8.4.7 `syslogd` Operations

The `FORCED_SOCKET` configuration parameter controls the use of the `/dev/log` pipe. When enabled, the `syslogd`(8) command cannot use the world-writable `/dev/log` pipe. Instead, `syslogd` is forced to use the socket interface.

To enable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Enforce socket usage for syslogd?` selection in the UNICOS Installation and Configuration Menu System.

By default, the `FORCED_SOCKET` configuration parameter is disabled on a UNICOS 10.0 system, which means that `syslogd`(8) and `syslog`(3) continue to work as they did on UNICOS 7.0 MLS systems. If your site chooses to run a Cray ML-Safe system configuration, this configuration parameter must be enabled.

8.4.8 Destructive Reads on Named Pipes

In previous releases of the UNICOS system with the MLS feature enabled, the mandatory access control (MAC) read policy has been enforced for reading all objects, including pipes. Because reading a pipe is destructive, this read operation is also considered a write operation. Therefore, pipes can be used by two cooperating processes to subvert the MAC policy. This is a covert channel.

The `SECURE_PIPE` configuration parameter can be used to close this covert channel. When enabled, MAC write access is required to read a pipe. When disabled, only MAC read access is needed to read a pipe.

To enable this parameter, use the `Configure system->Multilevel security (MLS) configuration->System options->Enforce restricted pipes?` selection in the UNICOS Installation and Configuration Menu System.

8.4.9 IPC Objects

The System V IPC mechanism uses three named object types on the UNICOS system: shared memory segments, semaphores, and message queues. These objects have associated security label and ACL information that users need to set and get by using the `spget`(1), `spset`(1), and `spclr`(1) commands. See the associated man pages for more information on how to set and get information on these object types. See the `ipcs`(1) man page for more information on IPC objects.

In addition, IPC object creation and use can be audited on a UNICOS system. See Section 8.8.7.6, page 276, and Section 8.8.7.8, page 284, for more information.

8.5 MLS Identification and Authentication (I&A)

This section provides an overview of I&A security implementation and describes login and password features used on a UNICOS system. The following sections refer to `login`(1), but the information applies to the `ftpd`(8) and `rexecd`(8) daemons also.

8.5.1 Overview of I&A Security Implementation

A UNICOS system supports the following interactive logins:

- Through `telnet`(1b) or `rlogin`(1b) for ordinary interactive login sessions
- Through `rsh` (see `remsh`(1b)) and `rexecd`(8) for single command executions
- Through `ftp`(1b) for interactive file transfer
- Through `su`(1) for changing user identity during a session
- Through `/dev/console` through the operator workstation (OWS)
- Through `dgdemon`(8) for hardware maintenance access

The following example gives an overview of the I&A sequence on a UNICOS system and highlights the security mechanisms outlined in using the `telnet` command.

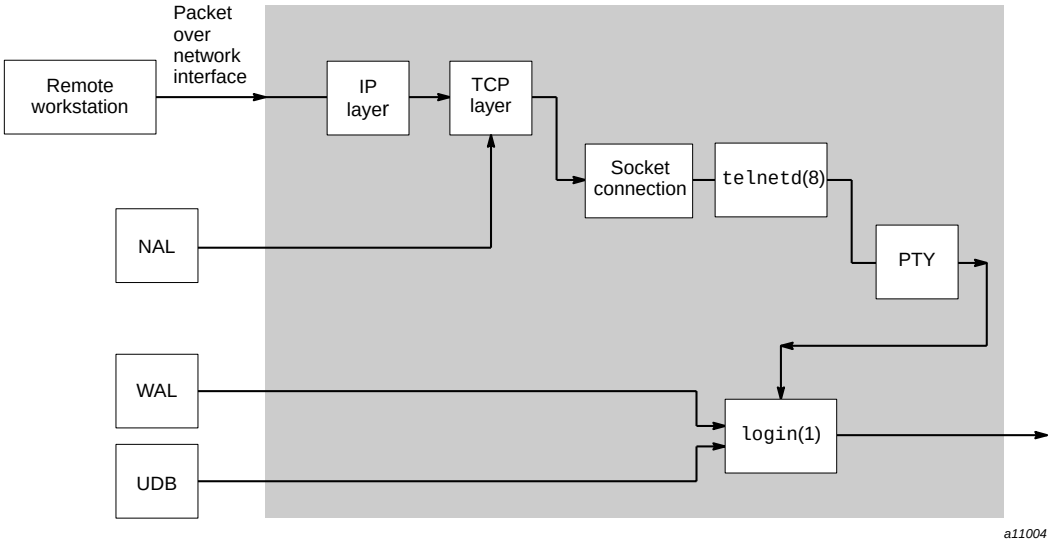


Figure 9. I&A Security Implementation

As shown in Figure 9, the user executes the `telnet`(1b) command to initiate the identification and authentication sequence by sending a packet and associated security label through the network interface to the IP layer.

A major function of the IP layer is to define the basic unit of data transfer on TCP/IP. The IP layer defines the security structure of these units (called an IP datagram). The security structure contains the following information:

Field	Description
Flag	Indicates a loopback packet; it is used for debugging and performance analysis. It is not sent over the network.
Security Option	The IP security option: Basic Security Option (BSO) or Common IP Security Option (CIPSO).
Authority/domain	The protection authority (for BSO) or domain of interpretation (CIPSO).
Active, minimum, and maximum security labels	Active security label of the packet and the minimum and maximum label as determined by the network access list (NAL) and the network interface label ranges.

UNICOS systems support the use of the BSO and CIPSO security options. The BSO security option supports the use of security levels only; no compartments are used. The CIPSO security option supports both levels and compartments. For more information on these security options, see the *UNICOS Networking Facilities Administrator's Guide.*

The IP layer performs label format translation between the BSO/CIPSO and UNICOS representation of labels.

The TCP layer enforces the security policy at the kernel/subject boundary for TCP users and at the system/network boundary for incoming datagrams. After receiving the IP packet from the IP layer, TCP validates an incoming datagram's active security label against the security label range of the network interface, the security label range defined in the NAL for the remote host, and the security label of the socket. A failure results in the packet being dropped and a violation logged.

The NAL controls remote host access. It grants or denies access to the local UNICOS system based on the security labels of the remote hosts (or networks). The NAL can have an entry for each remote host or network that is allowed to connect to the UNICOS system. A default entry can be used in the NAL.

Each NAL entry describes the security attributes associated with the remote host/network and the IP security option to be applied for that host's/network's communication with the local UNICOS applications. A remote host for which there is no specific host record, no applicable network entry, and there is no default NAL defined, is denied access to the UNICOS system.

After the checks are successfully completed, TCP sets the security label of the newly created socket. The label of the socket is set to the label of the incoming packet, and the label range of the socket is set to the intersection of the label range of the remote host's/network's NAL entry and the label range of the network interface.

TCP port numbers less than 1024 are considered privileged. A TCP peer that communicates using a privileged port from a Cray ML-Safe host is considered to be Cray ML-Safe. The following UNICOS processes use the privilege necessary to use these ports: `ftpd`(8), `inetd`(8), `lpd`(8), `rexecd`(8), `rlogind`(8), `rshd`(8), `telnetd`(8), `portmap`(8), and NQS.

`telnetd` opens the first pseudo pty available and sets the security label on the master and corresponding slave and makes it available for the user process.

The session is initiated with the security attributes that are determined by the centralized I&A mechanism.

8.5.2 Login Procedures

The following sections describe login procedures and features on a UNICOS system. Also, the user exits for the UNICOS MLS identification and authentication mechanism are explained.

8.5.2.1 Interactive Logins

On a UNICOS system, the security administrator creates and maintains the user database (UDB) that contains the following security-relevant information for each user who is allowed to access the system:

- Minimum and maximum security levels
- A default security level
- Active compartment(s)
- Authorized compartment(s)
- Minimum compartment set
- Permissions
- Maximum integrity class (obsolete)
- Active integrity class (obsolete)
- Authorized category (or categories)
- Active category (or categories)
- An encrypted password

This information is assembled for use by `login`(1) to authenticate each user who tries to access the UNICOS system.

The interactive login procedures for both a UNICOS and a Cray ML-Safe system configuration are explained in more detail in the *UNICOS Multilevel Security (MLS) Feature User's Guide.*

8.5.2.2 Remote Logins with SecurID Card

The UNICOS system supports the use of the SecurID card, which is manufactured by Security Dynamics, Inc. This authentication mechanism makes it harder to break into accounts because new passcodes are generated for each authentication and a passcode cannot be used more than one time.

Note: Use of SecurID is optional on a Cray ML-Safe system configuration.

The SecurID hardware, software, and documentation must be ordered from Security Dynamics, Inc. (see "Other publications" in the preface for the address). For more information on how to use the SecurID card, refer to instructions provided by Security Dynamics, Inc.

8.5.3 Centralized Identification and Authentication (I&A)

The UNICOS system supports the centralized identification and authentication (I&A) mechanism.

The following commands and daemons use centralized I&A on the UNICOS system:

- `dgdemon`(8)
- `ftpd`(8)
- `login`(1)
- NQS
- `rexecd`(8)
- `rshd`(8)
- `su`(1)
- `cron`(8)

`rlogind`(8) and `telnetd`(8) indirectly use centralized I&A, as both of these daemons use `login` through the `exec`(2)) system call. `cron` uses only the `ia_mlsuser`(3) library routine.

8.5.3.1 Checks and Operations

The I&A mechanism includes the following checks and operations, not all of which may be performed (depending on the request). For example, requesting passwords for batch jobs is not always necessary.

- Identify the user
- Check the password authentication
- Check SecurID authentication (if used)

- Check the workstation access list (WAL)
- Check password expiration
- Check the user's disabled flag
- Determine the session's security attributes
- Audit the successful creation of a session
- Audit the failed creation of a session and list the reason
- Update the user database (UDB) with successful or failed login information
- Disable the account after too many failed login attempts

The workstation access list (WAL) controls user access to a UNICOS system from remote hosts and workstations. It grants or denies access to services on the local system based on the identification of the user and the host or workstation from which the service request originated. The WAL can have an entry for each network address (host or workstation) that can connect to the system. If there is not an entry for the network address, it is automatically allowed to connect to the system.

Each WAL entry specifies the users and groups allowed access to a UNICOS system from that address and the services authorized for each of them to use at that address. See the *UNICOS/mk Networking Facilities Administration* for a complete list of supported services and the rules for gaining access through the WAL.

8.5.3.2 Library Routines Supporting I&A

The following library routines implement the I&A mechanism and provide a consistent, common mechanism for user identification, user authentication, auditing, user database (UDB) updating, and calculating a user's session security attributes. For more information on these routine, see the man pages in the *UNICOS System Libraries Reference Manual*:

Routine	Description
`ia_user`(3)	The common UNICOS I&A authentication mechanism.
`ia_failure`(3)	The I&A failure processing routine. It manages updating the UDB entry authentication failure information, performs I&A failure auditing, and processes login delay.

`ia_success`(3)	The I&A success processing routine. It manages updating the UDB entry authentication success information and performs the I&A success auditing.
`ia_mlsuser`(2)	Determines a user's session MLS attributes.

The `ia_mlsuser` routine uses six security labels when determining the security label of a session. Two of these labels are related to the network connection, three are related to the UDB entry of the user, and the last label is a requested label. In this description, a security label is defined as x:y, where x is the security level and y is the compartment bit mask.

The network labels used are the maximum label on the connection (referred to as `netmax` in the following examples) and the active label on the connection (referred to as `netmin` in the following examples). The range of the connection is defined as `netmin` to `netmax`.

The UDB labels include the following:

- The `maxlvl` and `compart` fields, which define the maximum security label for the user. In the following examples, this label is referred to as `udbmax`.
- The `deflvl` and `defcomps` fields, which define the default security label for the user. In the following examples, this label is referred to as `udbdef`.
- The `minlvl` and `mincomps` fields, which define the minimum security label for the user. In the following examples, this label is referred to as `udbmin`.

The value of the `deflbl_as_minlbl` field (product `login`) of the configuration file option forces the user's default UDB security label to serve as the user's minimum UDB security label.

The range of the user's security label is defined as `udbmin` to `udbmax`. The range of the session's security label is determined first. The session's security label is defined as the intersection of the connection's security label range and the user's security label range. The session's maximum security label is the greatest lower bound (GLB) of the user's and connection's maximum security labels. The session's minimum security label is the least upper bound (LUB) of the user's and connection's minimum security labels.

LUB is the greater of the two levels and the union of the two compartment sets. The LUB of 0:011 and 1:001 is 1:011. The LUB can be thought of as the least label that dominates both labels.

The GLB is the lesser of the two levels and the intersection of the two compartment sets. The GLB of 0:011 and 1:001 is 0:001. The GLB can be thought of as the greatest label that is dominated by both labels.

The session's security label range determines the minimum and maximum security labels for the session. The user is not allowed on the system if there is no intersection between the ranges.

Next, the active security label of the session is determined. If a label is requested, the active label is set to the requested label. Access is denied if the requested label is not within the session's security label range.

If a label is not requested, the session's active label is set to `udbdef`, if `udbdef` is within the session's security label range. The session's active label is set to `udbmin`, if `udbdef` is not within the session's security label range.

The following examples show how the session's security label is determined, based on the security labels of the connection and the UDB entry values.

In the following example, assume the following values have been defined for the connection's security label range and the user's UDB entries:

```
netmin = 1:00001
netmax = 4:00001
udbmin = 0:00000
udbdef = 0:00010
udbmax = 5:00010
```

The connection's security label range is 1:00001 through 4:00001, while the user's security label range is 0:00000 through 5:00010.

Access would be denied, because there is no intersection between the connection's and user's security label ranges. The network connection has a minimum compartment set of 00001 and the user's security label range does not include this compartment set.

In the following example, assume following values have been defined for the connection's security label range and the user's UDB entries and that the `deflbl_as_minlbl` field has not be configured:

```
netmin = 1:00000
netmax = 4:01110
udbmin = 0:00000
udbdef = 2:00010
udbmax = 5:01010
```

The connection's security label range is 1:00000 through 4:01110, while the user's security label range is 0:00000 through 5:01010. The session's security label range is 1:00000 through 4:01010.

The `udbdef` is within the session's security label range, so the active security label is set to the value of udbdef (2:00010).

In the following example, assume following values have been defined for the connection's security label range and the user's UDB entries and that the `deflbl_as_minlbl` field has not be configured:

```
netmin = 1:00010
netmax = 1:01110
udbmin = 0:00000
udbdef = 2:00010
udbmax = 5:01010
```

The connection's security label range is 1:00010 through 1:01110, while the user's security label range is 0:00000 through 5:01010. The session's security label range is 1:00010 through 1:01010.

The `udbdef` is not within the session's security label range, so the active security label is set to the session's minimum label (1:00010). Access would have been denied if `deflbl_as_minlbl` was configured. The user's security label range would have had a minimum level of 2 and the maximum connection level is 1.

8.5.4 I&A User Exits

Warning: The centralized I&A user exits should not be used on a Cray ML-Safe system configuration.

Seven user exits are supported on a UNICOS system. The user exits allow a site to control user I&A, including I&A success and failure process. Some examples are as follows:

- Supporting use of a local password format
- Allowing validation information to be held on a remote (that is, front-end) host
- Disallowing multiple logins
- Bypassing password processing
- Limiting access to a selected group of users or selected network address
- Disabling the use of the `su`(1) command to become `root`

The ia_user(3) routine supports three user exits: one at the beginning, which allows for complete replacement of the routine; a second one, which comes after normal I&A for additional site-specific authentication; and a third at the end of the routine. These routines are as follows:

User exit	Description
ia_uex_authrep	Provides the ability to replace functionality performed by ia_user or modify the ia_user parameters. The ia_uex_authrep routine has write access to all of the ia_user parameters, allowing ia_uex_authrep to modify the parameters. In addition, this routine can request that ia_user does not perform normal processing, but returns on return from ia_uex_authrep.
ia_uex_authadd	Provides the ability for additional authentication after successful normal authentication. It is called by ia_user after identification and the requested authentication has been performed. It is called before the password aging, maximum logins, and other login/password checking mechanisms are processed. This exit has write access to all of the ia_user parameters, allowing ia_uex_authadd to modify these parameters.
ia_uex_authend	Provides capability for a user exit to be called at the end of the ia_user routine regardless of the status that would be returned by ia_user. ia_uex_authend has write access to all of the ia_user parameters, allowing ia_uex_authend to modify the parameters.

The ia_failure(3) routine supports two user exits: one at the beginning, which allows for complete replacement of the routine; and a second one, which comes after normal auditing for additional site-specific auditing. These routines are as follows:

User exit	Description
ia_uex_failure	Provides for the complete replacement of ia_failure or the ability to modify the ia_failure parameters. The ia_uex_failure routine is called at the beginning of ia_failure.

`ia_uex_failaudit`	Provides for additional processing to be performed after normal I&A failure logging is complete, but before log-delay processing. The `ia_uex_failaudit` routine is called at the end of `ia_failure` before log-delay processing. The `ia_uex_failaudit` routine has write access to all of the `ia_failure` parameters, allowing `ia_uex_failaudit` to modify the parameters.

The `ia_success`(3) routine supports two user exits: one at the beginning, which allows for complete replacement of the routine; and a second one, which comes after normal auditing for additional site-specific auditing. These routines are as follows:

User exit	Description
`ia_uex_success`	Provides for the complete replacement of `ia_success` or the ability to modify the `ia_success` parameters. The `ia_uex_success` routine is called at the beginning of `ia_success`.
`ia_uex_succaudit`	Provides for additional processing to be performed after normal I&A success logging is complete. The `ia_uex_succaudit` routine is called at the end of `ia_success`. The `ia_uex_succaudit` routine has write access to all of the `ia_success` parameters, allowing `ia_uex_succaudit` to modify the parameters.

In addition, `login`(1) and `su`(1) supports the use of a user exit that is called before setting process attributes for the user's process. This routine is as follows:

User exit	Description
`ia_uex_preattr`	Provides for additional processing before the `login` and `su` processes set their attributes to that of the user. The `ia_uex_preattr` routine is called while the process is still running with special attributes (that is, as super user or with privilege).

The `ia_mlsuser`(3) routine does not support the use of user exits.

The user exits called at the beginning of the `ia_user`, `ia_failure`, and `ia_success` routines support the ability to request normal processing after returning from the user exit. If normal processing is not requested, the entire

routine is bypassed after the user exit processing is completed. In this case, the corresponding user exits in the `ia_user`, `ia_failure`, and `ia_success` routines are also bypassed.

8.5.5 Password Security

Most systems use passwords to regulate access to the system and prevent unauthorized individuals from logging into the system. It is important to establish a set of password guidelines for system users, educate the users on the importance of selecting, protecting, and using their passwords, and then enforce the guidelines by monitoring and auditing password use.

On a UNICOS system using a Cray ML-Safe configuration, it is of critical importance to enforce the correct use of passwords, and to use the monitoring and auditing features provided, especially in the case where repeated invalid password attempts are made.

The following list provides some rules to enforce, configuration parameters that can be used for password protection, and commands that can be used to audit password use on a UNICOS system:

- Change the `root` password often and audit its use. Limit the number of people who are allowed access to the `root` password.
- Remove old login accounts or disable them.
- Use the `CONSOLE_MSG`, `DISABLE_ACCT`, `MAXLOGS`, `LOGDELAY`, `DISABLE_TIME`, and `DELAY_MULT` parameters to protect MLS logins and passwords from unauthorized use and to lock a user's account after a specified number of failed login attempts; see Section 8.5.5.11, page 220, for more information.
- Enable the machine-generated password feature by setting the `RANDOM_PASS_ON` parameter. This feature generates a pronounceable, yet hard-to-guess password for users. See Section 8.5.5.10, page 217, for more information.
- Force users to change their passwords often by using the password aging feature; see Section 8.5.5.3, page 214, for more information.
- Use the password locking feature to lock the passwords of users on vacation or absent for any extended amount of time. See Section 8.5.5.6, page 215, for more information.
- Monitor and audit the use of passwords. This can be done in a variety of ways.

  - Use the spcheck(8) command to monitor failed attempts from the su program. See See Section 8.8.9, page 341, for more information.
  - Use the cll(8) command to display a user's invalid login attempts. See Section 8.5.6, page 225, for more information.
  - Use the user trap feature to monitor the activity of suspicious logins. See Section 8.5.5.7, page 216, for more information.
  - Use the reduce command to audit login activity from security log entries. See Section 8.8.7.9, page 287.

- Do not run rshd(8).

Ultimately, it is the responsibility of the security administrator to select the password features to be used, to educate system users on how to use these features, and to enforce the proposed guidelines. The remaining sections explain the password protection features in more detail.

8.5.5.1 Last Login Notification

At the time of each login, this feature allows the system to display the last login date, the last login time, the number of intervening login failures, and the ID of the terminal at which the user last logged in. If the user recognizes a discrepancy, the password compromise should be reported to the security administrator and investigated; see Section 8.5.6, page 225, for more information.

8.5.5.2 Generic Login Message

This feature allows the system to display a generic Login incorrect message when an unsuccessful login attempt is detected. Because it does not explicitly identify the incorrect portion of the login entry, this form of reply makes it harder to guess user names and passwords.

8.5.5.3 Password Aging

Use of the age field in the UDB allows a maximum and minimum number to be assigned to each user password. The maximum number specifies the maximum number of weeks that the password can be used. When this limit is reached, the user is forced to change his or her password during login.

The minimum number specifies the number of weeks that a user must keep a password before changing it again. Use of the minimum number prohibits users from changing their password and then immediately changing it back to the original password. The limits assigned to each user must be greater than or

equal to and less than or equal to 63. The following example shows how to use the udbgen(8) command to assign or change maximum or minimum numbers (the maximum number is the first number in the age field):

```
/etc/udbgen -c "update:jack:age:6,1:"
```

In the preceding example, Jack can use his password for a maximum of 6 weeks before being forced to change it and must use it for a minimum of 1 week before he is allowed to change it.

The system or security administrator can also force users to change their passwords before the next login attempt, as shown in the following example:

```
/etc/udbgen -c "update:jack:age:force:"
```

The force field forces the user to change the password during the next login attempt. See udbgen(8) and udb(5) for more information on password aging.

8.5.5.4 Password Suppression

By default, this feature allows the system to suppress the display or printing of passwords supplied by users.

8.5.5.5 Password Encryption

This feature allows each user password to undergo a one-way encryption before it is written to the UDB. At the time of login, the password supplied by the user is also encrypted and compared to the encrypted password in the UDB. This method prevents the display, storage, or reading of raw passwords.

8.5.5.6 Password Locking

This feature allows the security or system administrator to lock a specific user password to prohibit access to the system. This may be necessary during periods of inactivity (for example, vacations). The disabled UDB security field is used to disable a login, as shown in the following example:

```
/etc/udbgen -c "update:jack:disabled:1"
```

When 1, the user is not allowed access to the system, even if a valid password is given. A value of 0 (the default value) enables the login.

8.5.5.7 User Trapping

When password compromise is suspected, this feature allows the security or system administrator to put the suspect user into a trap mode by assigning the `usrtrap` permission in the user's UDB entry.

Assigning this permission causes the system to log all auditable events, regardless of the system-wide auditing configuration (except for `SLG_STATE`). This includes all discretionary and mandatory access attempts made by the user. Use of the `usrtrap` permission results in the generation of many audit records, which means that the size of the internal kernel audit buffer may have to be increased.

To review the actions of this trapped login, use the `reduce`(8) command as shown in Section 8.5.6, page 225.

8.5.5.8 Restricted Directory

The UDB contains a root directory for a user's environment; at login time, `login`(1) issues a system call to `chroot`(2), causing this directory to become the root directory for the user. This mechanism confines users to a given root directory and its subdirectories. This effectively confines the user to a subset of the file system hierarchy. Consequently, the user cannot access files in directories outside the restricted environment.

A typical restricted environment should permit access to at least the following subdirectories: `/bin`, `/etc`, `/tmp`, and `/dev`; the security administrator should place separate copies of these subdirectories in the restricted environment. This allows users in a restricted environment to execute command files residing in the restricted directory. Regardless of whether a restricted environment is established, user access to directories is always subject to mandatory and discretionary access controls.

8.5.5.9 Login Attempts

This feature defines the maximum number of login attempts per connection. When this limit is reached, `login`(1) exits and the user must reestablish a connection to the UNICOS system before trying to login again. Use of this feature does not affect or supersede the use of the `MAXLOGS` or `DISABLE_TIME` parameters (see Section 8.5.5.11, page 220). Also, use of this feature does not depend on the MLS feature being enabled.

8.5.5.10 Machine-generated Passwords

Although the password is usually the first line of defense in protecting access to a system, and in spite of any attempts to educate users on the importance of selecting proper passwords, many users continue to select simple, easy-to-guess passwords.

Use of machine-generated passwords enables the system to force users to select a randomly-generated password that is both easy to remember and hard for anyone else to guess, thus making password cracking harder to accomplish. You should note that use of this feature makes it impossible for users to pick their own passwords (that is, the "normal" UNICOS password mechanism is overridden when the machine-generated password feature is enabled); the only passwords available are the ones generated by this feature.

To enable the machine-generated password feature, the `RANDOM_PASS_ON` parameter must be set to `ON`. To do this, use the `Configure system->Multilevel security (MLS) configuration->Login / Password options->Machine-generated passwords?` selection in the UNICOS Installation and Configuration Menu System.

In addition, the minimum length of the generated password is controlled by the `PASS_MINSIZE` parameter. The default minimum size is 8. Cray established this default, because use of a smaller number leads to a substantially greater chance of generating duplicate passwords.

To change the `PASS_MINSIZE` parameter, use the `Configure system->Multilevel security (MLS) configuration->Login / Password options->Minimum machine-generated password length` selection in the UNICOS Installation and Configuration Menu System.

The maximum length of the generated password is controlled by the `PASS_MAXSIZE` parameter. The default maximum size is 8. UNICOS does not support the use of passwords that are greater than 8 characters, so `PASS_MAXSIZE` cannot be set greater than 8. Also, `PASS_MAXSIZE` must be greater than or equal to `PASS_MINSIZE`; if not, the password generator forces the maximum size to be greater than or equal to the minimum size.

To change the `PASS_MAXSIZE` parameter, use the `Configure system->Multilevel security (MLS) configuration->Login / Password options->machine-generated password length` selection in the UNICOS Installation and Configuration Menu System.

The algorithm used by the machine-generated password feature generates passwords that are easy to remember, but are difficult to guess. The following list contains examples of the type of passwords generated:

- lempamdo
- coochona
- vethsymy

Note: Although the chances are extremely small, the password-generating algorithm may produce a password that may seem offensive to a user. The appearance of such a password is random and is not intended to be offensive. A user has the choice of rejecting any generated password and picking a subsequent password.

As explained in previous paragraphs, because passwords of less than 7 characters substantially increase and less than 8 increase the possibility of generating duplicate passwords, and the fact that 8 is the maximum length that can be used on a UNICOS system, a default of 8 is used for both PASS_MINSIZE and PASS_MAXSIZE on a UNICOS system. It is recommended that your site uses the default setting of 8 for both parameters.

Enabling this feature introduces user interface changes to the passwd(1) and nu(8) command interfaces; these commands use the ranpass utility to generate the passwords. There are no changes to login(1), because it executes passwd to change passwords.

When machine-generated passwords are enabled the passwd command prompts the user with a password, as shown in the following example:

```
$ passwd
Changing password for jane
Old password:
Your new password is: kudniqui
Re-enter password or (CR) to get another:$
```

The user has the option to accept the password or to have another password generated, as shown in the following example:

```
$ passwd
Changing password for jane
Old password:
Your new password is: kudniqui
Re-enter password or (CR) to get another:
Your new password is: keltifok
Re-enter password or (CR) to get another:
```

```
Your new password is: coochona
Re-enter password or (CR) to get another:
Your new password is: rhecirou
Re-enter password or (CR) to get another:
Your new password is: osniyuib
Re-enter password or (CR) to get another:
$
```

The new password cycle continues until the user reenters the generated password, signifying that he or she accepts the password.

You cannot use the `passwd -b` (batch) option when machine-generated passwords are enabled.

When administrators uses the `nu -a` or `nu -m` options, they can no longer select a user's password as before, but must select one of the machine-generated passwords, as shown in the following example:

```
$ nu -a
cmd/nu/nu.c   80.4   9/25/91 15:51:03 (sn18:/etc/nu.cf60)
Login name? (1-8 characters) [quit] bob
New password is: ralcuhyd
Re-enter new password or (CR) to get another:
New password is: osniyuib
Re-enter new password or (CR) to get another:
Enter actual user name: Bob Smith
      .
      .
      .
```

After the password is selected, the `nu` sequence is the same as when the machine-generated password feature is not enabled.

When using this feature, it is important that you as a security administrator do not select a new password for yourself or other users in the presence of another person (or, at the very least, shield your screen from the other person's view). Also, when you are done selecting a password, remove or erase the screen, so that no one else can obtain the new password.

8.5.5.11 MLS Login and Password Protection Features

The UNICOS MLS feature provides six site-configurable parameters to protect and monitor the login process and the use of passwords. The parameters are as follows:

- MAXLOGS
- DISABLE_ACCT
- DISABLE_TIME
- CONSOLE_MSG
- LOGDELAY
- DELAY_MULT

All are found in `uts/cf.SN/config.h`.

Use of these parameters is site-dependent and can be used in a variety of combinations to protect MLS logins and passwords. The CONSOLE_MSG and DISABLE_ACCT parameter work independently of each other. That is, CONSOLE_MSG does not have to be on for DISABLE_ACCT to work and vice versa, although both can be on and continue to work successfully. Each of these two parameters affect how the MAXLOGS parameter is used; DISABLE_ACCT affects if DISABLE_TIME works.

Regardless of how the parameters are used, it is important for the security administrator to audit password usage on a daily basis and investigate any excessive or suspicious failed login attempts.

The following sections explain how to use the UNICOS installation and configuration tool to set these parameters and provide examples of how the parameters work.

8.5.5.11.1 The MAXLOGS, DISABLE_ACCT, and DISABLE_TIME Parameters

MAXLOGS defines the maximum number of consecutive failed login attempts allowed to a user. For example, if MAXLOGS is set to 3, a user is allowed only three attempts at selecting a correct password. Whether the user is allowed to log in on the fourth (or subsequent) correct attempt depends on the state of DISABLE_ACCT and DISABLE_TIME.

The `logfails` field in a user's UDB entry is incremented each time the user makes an incorrect login attempt. When `logfails` equals or exceeds MAXLOGS, the account is disabled only if the DISABLE_ACCT parameter is ON. If

DISABLE_ACCT is OFF, then the user's login attempts are not restricted by the limit set by MAXLOGS.

DISABLE_TIME defines the number of seconds a user is disabled from logging on after exceeding the limit set by MAXLOGS (assuming that DISABLE_ACCT is enabled). It is recommended that DISABLE_TIME always be set to a nonzero number. Setting this parameter to a negative number disables the user indefinitely (or until a security administrator intervenes). When using a positive number, it is recommended that you use a value no less than 60 seconds, as smaller values render this parameter ineffective.

See the examples in Section 8.5.5.11.5, page 223, for more information on the use of these parameters.

MAXLOGS does not affect root and security administrator accounts. That is, root can log in from the console (/dev/console) and authorized security administrators can log in from any terminal even though MAXLOGS is exceeded.

The MAXLOGS parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Maximum login attempts (MAXLOGS) selection in the UNICOS Installation and Configuration Menu System.

The DISABLE_ACCT parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Disable account after MAXLOGS attempts? selection in the UNICOS Installation and Configuration Menu System.

The DISABLE_TIME parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Disable time after max logins (seconds) selection in the UNICOS Installation and Configuration Menu System.

8.5.5.11.2 The CONSOLE_MSG Parameter

When enabled, CONSOLE_MSG sends a message to the console when a user equals or exceeds the login limit set by MAXLOGS. See the examples in Section 8.5.5.11.4, page 222, for more information on the use of these parameters.

The CONSOLE_MSG parameter is configured by using the Configure system->Multilevel security (MLS) configuration->Login / Password options->Console messages upon reaching MAXLOGS? selection in the UNICOS Installation and Configuration Menu System.

8.5.5.11.3 The LOGDELAY and DELAY_MULT Parameters

LOGDELAY defines the number of seconds that must elapse between login prompts after a failed login attempt. For example, if LOGDELAY is set to 10 seconds, then the login prompt would not appear for 10 seconds after each failed login attempt.

The LOGDELAY parameter is configured by using the `Configure system->Multilevel security (MLS) configuration->Login / Password options->Seconds to delay between login tires` selection in the UNICOS Installation and Configuration Menu System.

When enabled, DELAY_MULT multiplies the number of seconds defined by LOGDELAY by the number of failed login attempts to linearly lengthen the delay between each login prompt after a failed attempt. This parameter is configured by using the `Configure system->Multilevel security (MLS) configuration->Login / Password options->Increment login delay time?` selection in the UNICOS Installation and Configuration Menu System. See Section 8.5.5.11.7, page 225, for more information.

8.5.5.11.4 Using the CONSOLE_MSG and MAXLOGS Parameters

When the CONSOLE_MSG parameter is on, and DISABLE_ACCT is off, MAXLOGS is used only to determine when a message is sent to the console. That is, when MAXLOGS is equaled or exceeded, a message alerts the operator or administrator that someone is attempting one or more invalid login attempts. This configuration does not prohibit a user from logging in with a correct password after MAXLOGS has been equaled or exceeded, however.

This situation is shown in Table 6. Assume that CONSOLE_MSG is ON, DISABLE_ACCT is OFF, MAXLOGS is set to 5, and DISABLE_TIME is set to 60.

Table 6. Login Protection Parameter Configuration, Example 1

Login sequence	User results	Console message
1 valid attempt	Login accepted	No message
1 invalid / 1 valid	Login accepted	No message
4 invalid / 1 valid	Login accepted	No message
5 invalid / 1 valid	Login accepted	1st console message
6 invalid / 1 valid	Login accepted	2nd console message

As shown in Table 6, messages are sent when MAXLOGS is equaled or exceeded, and the user is allowed to successfully log in when a valid password is entered. In other words, the user (malicious or otherwise) is allowed access and it is up to the operator or administrator monitoring the console to decide what to do. The table also shows that the DISABLE_TIME parameter is not used in this situation.

The console that receives the message is defined by the SYSTEM_ADMIN_CONSOLE parameter (see Section 8.7.5.2.2, page 233, for more information on this console). The appearance of the message on the console screen is preceded by a warning bell.

The format of the message is as follows:

Warning: user *loginname* has reached or exceeded MAXLOGS on *tty/pty line* from host *hostname*

The fields are defined in the following list:

Field	Description
loginname	Login name of the user
tty/pty line	The device name
hostname	The name of the user's host

The following is an example of the warning message:

```
Warning: user tim has reached or exceeded MAXLOGS
on ttyp065 from host branch15
```

The warning message generated from NQS is slightly different than the one shown previously, in that the *tty/pty line* portion is replaced with via NQS. Otherwise, the field descriptions are the same as defined previously and NQS uses the password protection features in the same way as login. The following example shows a warning message for a NQS user:

```
Warning: user mary has reached or exceeded MAXLOGS via
NQS from host cray
```

8.5.5.11.5 Using the DISABLE_ACCT, MAXLOGS, and DISABLE_TIME Parameters

When the DISABLE_ACCT parameter is on, and CONSOLE_MSG is off, MAXLOGS and DISABLE_TIME are used to prohibit incorrect login attempts.

This situation is shown in Table 7. Assume that CONSOLE_MSG is OFF, DISABLE_ACCT is ON, MAXLOGS is set to 5, and DISABLE_TIME is set to 60.

Table 7. Login Protection Parameters Configuration, Example 2

Login sequence	User results	Console message
1 valid attempt	Login accepted	Not applicable
1 invalid / 1 valid	Login accepted	Not applicable
4 invalid / 1 valid	Login accepted	Not applicable
5 invalid / 1 valid	Login denied	Not applicable
5 invalid / Time-out expired / 1 valid	Login accepted	Not applicable

As shown in Table 7, the user is granted access until MAXLOGS is equaled (line 4 of the table). When the user makes five invalid attempts, he or she is denied access even though a valid attempt is then made. This is caused by the use of the DISABLE_TIME parameter, which defines the number of seconds a user's account is disabled after exceeding the MAXLOGS limit.

When this defined amount of time has expired, the user is allowed one more attempt at logging in. If successful, the user is granted access (as shown in line 5 of the table). If unsuccessful, then the user must again wait for the time specified by DISABLE_TIME before being allowed one more attempt to log in. Notice that no messages are sent to the console; this is because CONSOLE_MSG is off.

The `Login incorrect` message appears for any login attempt made after MAXLOGS is equaled or exceeded and prior to the expiration of the time limit set by DISABLE_TIME,

The DISABLE_TIME sequence continues until a successful login attempt is completed or a security administrator intervenes by resetting the login failures. As previously explained, each failed login attempt is tracked by the UDB `logfails` field. At no time is the `logfails` field set to 0 when DISABLE_TIME expires.

8.5.5.11.6 Using CONSOLE_MSG and DISABLE_ACCT Parameters

When both the DISABLE_ACCT and CONSOLE_MSG parameters are set to on, then login is denied when MAXLOGS is equaled or exceeded, and messages are sent to the console. This is shown in Table 8. Assume that CONSOLE_MSG is ON, DISABLE_ACCT is ON, MAXLOGS is set to 5, and DISABLE_TIME is set to 0.

Table 8. Login Protection Parameters Configuration, Example 3

Login sequence	User results	Console message
1 valid attempt	Login accepted	No message
1 invalid / 1 valid	Login accepted	No message
5 invalid / 1 valid	Login denied	1st console message
5 invalid / Time-out expired / 1 valid	Login accepted	1st console message
5 invalid / Time-out expired / 1 invalid	Login denied	2nd console message
5 invalid / Time-out expired / 2 invalid	Login denied	3rd console message

In Table 8, DISABLE_ACCT, MAXLOGS, and DISABLE_TIME work as explained for Table 7, page 224. In addition, because CONSOLE_MSG is enabled, the correct number of messages are sent to the console for invalid attempts.

8.5.5.11.7 Using the DELAY_MULT Parameter

A site can further restrict login attempts by setting the DELAY_MULT parameter; when enabled, it multiplies the LOGDELAY parameter by the number of successive failed attempts. This lengthens the delay between each login prompt after a failed login attempt.

For example, if LOGDELAY is set to 10 seconds, the login prompt would not appear for 20 seconds after the second failed login attempt, 30 seconds after the third failed login attempt, and so on.

Caution: Use of the DELAY_MULT parameter tends to disclose information about valid users. This information becomes apparent to unauthorized users because a valid user name shows a delay between successive failed attempts.

8.5.6 Password Auditing

To audit password usage, the security administrator can use the spcheck -q command (which reports excessive failure of the su(1) command) and check the output of the reduce(8) command, as shown in the following examples. The following example displays the users who have exceeded the limit defined by MAXLOGS:

```
/etc/reduce -t logn | grep Disabled
```

The following example displays a line of information that contains the user name and login ID for any user who had a password failure when logging in:

```
/etc/reduce -t logn | grep Password
```

To help determine the guesser's identity, the output of the login record through the `reduce` command identifies the host where the attempts took place. See Section 8.8.7.9, page 287, for more information on this record.

A properly authorized user can also use the `cll`(8) command to display a specific user's failed login attempts, as shown in the following example. See Section 8.5.6.1, page 226, for a definition of a properly authorized user for the `cll` command:

```
$/etc/cll -l jack
cll: user <jack> has 0 login failures.
```

The `cll -L` command can be used to display the failed login attempts (if greater than 0) for all user logins, as shown in the following example:

```
$/etc/cll -L

User    0 <root   > has    1  login failures
User    6 <nqs    > has    5  login failures
User  128 <tom    > has   25  login failures
User  146 <alice  > has    2  login failures
User  149 <sue    > has    1  login failures
User  204 <mary   > has    7  login failures
```

8.5.6.1 Reenabling Accounts

The configuration parameters explained in Section 8.5.5.11, page 220, can be used to prohibit incorrect login attempts or permanently disable a user's account. The following information assumes that you understand how these parameters can be used to disable a user's account.

To reenable a disabled user's account, use the `cll`(8) or `udbgen`(8) commands, as shown in the following examples.

You must be properly authorized to use the `cll` command to reset the `logfails` field in the UDB for one user or all users. Properly authorized is defined as follows:

- On a system with PRIV_SU enabled, you must be the super-user.
- On a system using the PAL-based privilege mechanism, you must have an active secadm or sysadm category.

The following example shows how to use the cll -r command to reset the logfails field for the user jack:

```
/etc/cll -r jack
```

To clear the logfails field in the UDB, use the udbgen command as shown in the following example:

```
/etc/udbgen -c "update:mary:logfails:0:"
```

You must be properly authorized to use the udbgen command. Properly authorized is defined as follows:

- On a system with PRIV_SU enabled, you must be the super-user.
- On a system using the PAL-based privilege mechanism, you must have an active secadm category to change all UDB fields. If you have an active sysadm category, you can change all UDB fields except for security-sensitive fields.

8.6 Object Reuse

The UNICOS kernel manages shared resources (for example, process table entries, inodes, disk blocks, I/O buffers, and user memory). To ensure proper operation, these resources must be initialized when they are released to the system and reallocated. For example, when a file is removed, its data blocks are released and made available for reallocation. If the data block contents were not initialized before allocation to a different file, it would be possible for users to search for sensitive data in reallocated disk blocks.

The UNICOS kernel ensures that the user-visible contents of all the kernel-managed data resources are overwritten with zeroes or initialized to new, correct values before they are reallocated. In the case of disk blocks, the memory representation of a disk block is filled with zeroes or the data specified by the calling process whenever the block is allocated. This prevents access to the previous contents.

Most data resources are initialized at allocation time, as this saves time if the resource is never reused. Some data resources are initialized at deallocation because reuse is a certainty or there is no way to ensure correct initialization at

reallocation time. In either case, the system call interface used by the UNICOS operating system prevents nonprivileged user-level processes from obtaining the contents of a data resource until it is allocated. This means there is no mechanism by which nonprivileged processes can bypass the UNICOS object reuse mechanisms.

For sites concerned about data that may remain on file system media when the media are removed, the UNICOS operating system provides the following additional mechanisms that manually or automatically overwrite disk blocks before they are released:

- The `spclr`(1) command, which overwrites the contents of files and deletes them. This allows users to ensure that the data in their files has been erased from the disk before the file is removed.
- The `SECURE_SCRUB` configuration parameter, which enables the overwriting of the contents of all disk blocks with zeroes before releasing them.

The following configuration parameters affect the behavior of the `spclr` command:

- `SANITIZE_PATTERN`
- `DECLASSIFY_DISK`
- `OVERWRITE_COUNT`
- `DECLASSIFY_PATTERN`

The `spclr -s` command overwrites disk space with a pattern determined by the `SANITIZE_PATTERN` parameter. It is recommended that the `SANITIZE_PATTERN` parameter be set to zeros. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Scrub disk write pattern` selection in the UNICOS Installation and Configuration Menu System.

The `DECLASSIFY_PATTERN` parameter allows you to set the original overwrite pattern used when the `spclr -d` command is executed (the default pattern is 0). You can also use the `-p` option with the `-d` option. In this case, disk space is overwritten with the pattern, then with the negated pattern, and finally with a random pattern.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Disk options->Disk declassify write pattern` selection in the UNICOS Installation and Configuration Menu System.

The OVERWRITE_COUNT parameter (default is three times) determines the number of times that the DECLASSIFY_PATTERN is written when the spclr -d command is executed. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Disk options->Disk declassify overwrite count selection in the UNICOS Installation and Configuration Menu System.

The DECLASSIFY_DISK parameter must be set to YES for the OVERWRITE_COUNT and DECLASSIFY_PATTERN parameters to work correctly. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Disk options->Allow disk declassification selection in the UNICOS Installation and Configuration Menu System.

When the SECURE_SCRUB parameter is enabled (ON), the disk space is automatically overwritten once when a file is removed. It is recommended that the overwrite pattern be set to zeros (which is done by setting the SANITIZE_PATTERN parameter).

If the SECURE_SCRUB parameter is disabled (OFF), the disk space is not overwritten unless the user specifically executes the spclr -s command.

To set the SECURE_SCRUB parameter, use the Configure system->Multilevel security (MLS) configuration->Disk options->Scrub data blocks on file selection in the UNICOS Installation and Configuration Menu System. This parameter is not required by the TCSEC, because UNICOS does not allow a user to access any residual disk data.

Caution: Significant performance degradation occurs if the SECURE_SCRUB parameter is enabled.

8.7 MLS Installation and Configuration

The following sections provide MLS installation and configuration information for UNICOS and Cray ML-Safe system configurations:

- Startup and shutdown procedures
- Organization of the MLS menus used on the UNICOS Installation and Configuration Menu System (ICMS)
- Installation procedures
- Defining MLS portions of UDB entries

- Directory initialization
- Labeling commands (privcmd(8))
- Configuring a Cray ML-Safe system configuration

8.7.1 System Startup Procedure

The MLS feature, enabled by default, does not alter system startup operations. When the MLS feature is enabled, however, the following message format is displayed at the operator's console during startup:

```
SECURE_SYS.levels =  PZ_MINLVL/PZ_MAXLVL, Compartment = xxxxxxxxxxxxxxxxxx
```

PZ_MINLVL (which is usually 0) and PZ_MAXLVL (which is usually 16) are the parameters that define the operating system's minimum and maximum security levels. The Compartment = xxxxxxxxxxxxxxxxxx field defines the operating system's compartment set. The following example shows the display:

```
SECURE_SYS levels =  0/16, compartment = 077777777777777777777
```

You can use the spset -s command to change the security level and compartment for the operating system. You must be properly authorized to use this option. See Section 8.7.5.2.1, page 232, for more information.

You can use the spget -s command to display the system's security label. Anyone can use this option to display the information.

During system startup (even after a system failure), the /dev/tty* entries are automatically accessed and cleared by spwcard(8).

8.7.2 Subsystem Startup Procedure

Daemons are started automatically during system startup from within the /etc/rc file through the /etc/sdaemon command, which is executed on entry to the multiuser state. You must be a properly authorized administrator to manually start or restart a daemon.

8.7.3 System Shutdown Procedure

The MLS feature, enabled by default, does not alter system shutdown operations. The shutdown(8) command sends a shutdown warning message on a UNICOS system through the wall(8) command to all users regardless of their security labels.

If you want to send a separate message to all users, then you must be properly authorized when executing the `wall` command. If you are not properly authorized then only those users executing at the same security label and have messages turned on receive the message.

A properly authorized user is defined as the super user on systems with `PRIV_SU` enabled; for systems using the PAL-based privilege mechanism, properly authorized is defined as having an active `secadm` or `sysadm` category.

8.7.4 System Clearing Procedure

Before the UNICOS system is started, the Cray System Clear package can be used to clear (or scrub) the Cray hardware environment.

The Cray System Clear utility is a stand-alone routine that writes over the data in all memory (mainframe, IOS buffer memory and IOP local memory), stacks, internal buffers, and registers, as well as on selected disks and specified high-speed channel buffers. The interactive session or a shell script can be used to activate and control the hardware scrubbing process.

It is important to note that this utility has not been integrated with the UNICOS startup process; therefore, system scrubbing operations must be initiated manually. See your Cray engineer for instructions on how to run this utility.

Note: The Cray System Clear utility is not supported for use on a Cray ML-Safe system configuration.

8.7.5 MLS Configuration Parameters

The following files are used to configure the MLS feature on a UNICOS system:

- `sys/secparm.h`
- `uts/cf.SN/config.h`
- `cf/seclabs.c`

The `secparm.h` file cannot be configured by your site; the remaining two files can be changed by the site. The following sections describe these files.

8.7.5.1 The `secparm.h` File

In pre-6.0 releases, the `secparm.h` file contained all the security parameters that could be configured by a site. For post-6.0 releases, it contains parameters that are used by the UNICOS kernel and cannot be changed by site personnel.

The file contains definitions for the following:

- The names and representations for the user permissions
- The definitions of the process 0 initial security parameters
- The macros used to identify certain Cray ML-Safe processes

8.7.5.2 The `uts/cf.SN/config.h` File

The `uts/cf.SN/config.h` file contains all the parameters that can be configured by the site (except for the site-specific level naming conventions and the site-specific compartment definitions, which are defined in the `seclabs.c` file). These `uts/cf.SN/config.h` parameters can be set by using the `Configure system->Multilevel security (MLS) configuration->`*appropriate selections* in the UNICOS Installation and Configuration Menu System.

The *appropriate selections* are as follows:

- `System options`
- `Network security options`
- `Login / Password options`
- `Disk options`
- `Security log configuration`
- `Import the security configuration ...`

Each security submenu has help files that explain the various selection or refers you to the proper documentation in this manual. The selections found in these menus are documented throughout the relevant sections of this MLS chapter and are not be repeated here, except for the following selections, which did not fall into any other discussion.

See the *UNICOS System Configuration Using ICMS* for more information on the selections and help files available for all MLS configuration menus.

8.7.5.2.1 Setting the System's Security Label

The operating system's default minimum and maximum security levels (`MINSLEVEL` and `MAXSLEVEL`, respectively) are defined in `/uts/cf.SN/config.h`.

To configure MINSLEVEL, use the Configure system->Multilevel security (MLS) configuration->System options->Minimum security level selection in the UNICOS Installation and Configuration Menu System.

To configure MAXSLEVEL, use the Configure system->Multilevel security (MLS) configuration->System options->Maximum security level selection in the UNICOS Installation and Configuration Menu System. This information is resident in memory.

The operating system's valid set of compartments (SYSVCOMPS) is also defined in uts/cf.SN/config.h. This octal mask can be set by using the Configure system->Multilevel security (MLS) configuration->System options->Valid system compartment mask (octal) selection in the UNICOS Installation and Configuration Menu System. For a compartment to be available on the system, its associated bit setting in this octal mask must be set to 1.

An appropriately authorized user can use the spset -s command to change this mask. See the *UNICOS User Commands Reference Manual* for more information on the spset(1) command.

8.7.5.2.2 Configuring Consoles

Access to the operator or administrator ID can be restricted to a console through use of the following parameter, which is found in uts/cf.SN/config.h:

Parameter	Description
SYSTEM_ADMIN_CONSOLE	Allows the site to set the system console to the named console; reset at time of system initialization. This parameter must be set to /dev/console.
SECURE_SYSTEM_CONSOLE	Use of this parameter is obsolete as of the UNICOS 9.2 release.
SECURE_OPERATOR_CONSOLE	Use of this parameter is obsolete as of the UNICOS 9.2 release.

To set the SYSTEM_ADMIN_CONSOLE, use the Configure system->Multilevel security (MLS) configuration->System

`options->Default system console` selection in the UNICOS Installation and Configuration Menu System.

Note: These parameters will no longer be available in the UNICOS 10.0 release.

8.7.5.3 The `seclabs.c` File

The `seclabs.c` file contains the site-specific compartment definitions and the security level naming options. You can define these parameters by selecting `Configure system->Multilevel security (MLS) configuration->Site labels configuration->Compartments` or `Levels` selections in the UNICOS Installation and Configuration Menu System.

8.7.5.3.1 Security Level Naming

If the security level names were imported during the installation process, then, by default, security levels (0, 1, 2,...) are assigned names (`level0`, `level1`, and so on). These security level names can be changed. For example, `level0` could be named `public`, `level2` could be named `private`, and so on.

It is possible to define new, or redefine existing security level names and values without rebuilding commands, utilities, and libraries (only the UNICOS kernel must be rebuilt). This capability is supported by using the `getsectab`(2) system call and `sectab`(5) structure. The UNICOS system allows the `getsectab`(2) system call to return the configured security tables, regardless of whether MLS is enabled. This allows you to to configure various MLS information (for example, security-relevant fields in the user database (UDB)) before actually booting a UNICOS MLS kernel.

8.7.5.3.2 Compartment Definition

Compartment use and definition should be determined by the security personnel before a Cray ML-Safe configuration of the system is installed. Redefining compartments after the system is installed and running can present many security holes.

For example, redefining a compartment, and then assigning the new compartment to a new user, could possibly grant the new user unwanted permission to an object labeled with the old compartment definition.

8.7.5.4 Permission Definitions

The following paragraphs clarify the definitions and differences between the following user permissions:

- `permbits` (permission bits for user permissions)
- `permits` (permission bits for user permissions on UNICOS systems)
- `sitebits` (permission bits reserved for site definition)

These permissions are defined in the user database (UDB) on UNICOS and Cray ML-Safe system configurations. These permissions are assigned to a user at login. See the `udbgen`(8) man page for more information on assigning these permissions. The permissions are used throughout the session unless explicitly turned off by a process.

`permbits` are used on both UNICOS systems with MLS and without MLS, and provide users the authority to perform certain functions such as changing the ownership of their files. `permbits` are not used within the Cray ML-Safe configuration kernel for determining privileges, although the tape subsystem uses the `tape-manage`, `bypasslabel`, and `wrunlab` permbits.

`permits` are similar in function to `permbits`. When `PRIV_SU` is enabled, the UNICOS system uses the `suidgid` and `usrtrap` `permits`. These `permits` have no effect on systems using the PAL-based privilege mechanism. The `permits` are permanently defined in the security parameter file (`sys/secparm.h`).

The `suidgid` permission gives the user explicit permission to set the set-user-ID (setuid) and/or set-group-ID (setgid) bits for a file. Restricted management of setuid and setgid files is enforced only on UNICOS systems with the `FSETID_RESTRICT` configuration parameter enabled. This permission is used only on systems with `PRIV_SU` enabled.

The `usrtrap` permission can be used on any UNICOS configuration. This permission is not, by a true definition of the word, a permission, but a process attribute. It is described here, as it is defined in `sys/secparm.h`. The `usrtrap` permission sets the user in trap mode, causing the system to log all auditable events, regardless of the system-wide auditing configuration (except for `SLG_STATE`). This includes all discretionary and mandatory access attempts by this user.

For example, if `SLG_ALL_VALID` is disabled, but the user has the `usrtrap` permission assigned, all valid file access attempts for the user are still logged. For

more information on the security log, see Section 8.8.1, page 260. Do not assign the `usrtrap` permission to the security administrator.

You may see instances where the the `reclsfy`, `lbypass`, `wrunlab`, and `install` permissions are displayed in audit records and other UNICOS outputs. These are obsolete `permits` and they have no effect on a UNICOS system.

`sitebits` are similar in function to `permbits`, but the meaning of each bit is defined by the site. Their usefulness is limited only by the site's need to maintain compatibility with other UNICOS sites. They can be used on a UNICOS system with `PRIV_SU` enabled, with the understanding that the site is aware of how any security-related `sitebits` could affect the default security policies enforced on either of these security configurations. The introduction of site-written code that uses `sitebits` to enforce security policies on a Cray ML-Safe system configuration should not be used unless proper accreditation procedures are followed.

8.7.6 Defining MLS UDB Entries

The user database (UDB) has fields that pertain specifically to the UNICOS MLS feature. These fields include the following:

Field	Description
`maxlvl`	Maximum security level
`minlvl`	Minimum security level
`defcomps`	Active compartments
`deflvl`	Default security level
`mincomps`	Minimum compartment set
`comparts`	Authorized compartments
`permits`	User permissions
`intcat`	Active category
`valcat`	Authorized categories

The `udbgen`(8) command creates and maintains the UDB.

Before booting a UNICOS system to multiuser mode, you should define all users in the UDB. All user entries should be carefully defined, because a user with more privileges than necessary can (inadvertently or maliciously) corrupt or destroy system or user data.

Defining the security administrator entry is of special importance to successfully installing and maintaining the UNICOS system. Administrative user entries should be set up according to the following guidelines:

- Assign each administrative user a unique, non-root login ID. If you need more than one person to function as the same type of administrator on non-PRIV_SU systems, define a unique login ID for each user and assign each login account the same administrative category. On UNICOS systems that depend on the super user for administration (that is, the system has PRIV_SU enabled), require administrators to use the su(1) command to become the root user.

Table 9 gives guidelines for defining the UDB security fields for administrators, operators, and general users on a UNICOS system.

Table 9. Suggested Values for UDB Security Fields

UDB	Security administrator	System administrator	System operator	Users
minlvl	0	0	0	0
maxlvl	16	*	*	*
deflvl	0	0	0	0
comparts	*	*	*	*
defcomps	None	None	None	None
mincomps	0	0	0	0
permits	suidgid	suidgid	suidgid	**
intcat	None	None	None	None
valcat	secadm	sysadm	sysops	None

* Definition of this field is site-specific.

**Assign the suidgid only if necessary. Assign the usrtrap permission if you with to log all the user's discretionary and mandatory access requests in the security log.

In addition to establishing a user's UDB entry, the security administrator should ensure that the user's home directory and any initial files (for example, .profile, .cshrc, and .login) are labeled to match the user's UDB entry.

8.7.7 Directory Initialization Procedures

Several directories must be either converted to multilevel directories (MLDs) or assigned the wildcard label to enable the concurrent processing of user clearances and file classifications at multiple security labels.

Assigning the wildcard label to many of the directories is done at system startup time by the `spwcard`(8) command, which is run automatically from the `/etc/rc` script. NQS and the tape daemon also assign the needed wildcard labels through their respective initialization routines. See Section 8.4.1.2.2, page 175, and Section 8.4.1.2.1, page 174, for more information on converting to MLDs and wildcard directories, respectively.

Note: Wildcard directories cannot be used on a Cray ML-Safe system configuration. Where wildcard labels were used on UNICOS systems (that is, non-Cray ML-Safe UNICOS systems), they have been replaced with MLDs. If your site has already configured the system to support a Cray ML-Safe system configuration, using the `spwcard` command does not relabel any file or directory (for example, relabel a MLD with a wildcard label).

The tape daemon and the NQS installation and startup procedures also do not support the use of wildcard labels on a Cray ML-Safe system configuration.

The `spwcard` command assigns a wildcard label to the following directories when it is executed by the `/etc/rc` script:

- `/tmp`
- `/usr/tmp`
- `/usr/mail`
- `/usr/spool/mqueue`

During the initial installation of NQS, the `qstart` utility executes `qconfigchk` and looks for the `NQE_NQS_MAC_DIRECTORY` variable. If the variable is set to 1, then the NQS spool directories will be created as MLDs.

If you are upgrading your NQS configuration to use MLDs, you must do the conversion manually. For more information on converting the NQS spool directories, see the *NQE Administration*.

The directories assigned the wildcard security label and automatically set by the NQS process are as follows:

- `/usr/spool/nqs/private/root/control`
- `/usr/spool/nqs/private/root/data`

- /usr/spool/nqs/private/root/failed
- /usr/spool/nqs/private/root/interproc
- /usr/spool/nqs/private/root/output
- /usr/spool/nqs/private/root/chkpnt
- /usr/spool/nqs/private/root/FIFO
- /usr/spool/nqs/private/root/LOGFIFO
- /usr/spool/nqs/private/root/reconnect
- /usr/spool/nqs/private/requests

8.7.8 The privcmd Command

The privcmd command sets file privileges, the security label, permissions mode, owner, owning group, and security flags on system objects. For more information on using this command, see the privcmd(8) man page in the *UNICOS Administrator Commands Reference Manual*.

8.7.9 MLS Installation and Configuration Procedures

Warning: In UNICOS 9.2 and later releases, all sites are required to assign PALs. The supported privilege configurations are as follows:

- PALs augmented by PRIV_SU
- PALs only

There are several procedures you can use for configuring the MLS feature on a UNICOS 10.0 system, depending on whether you are performing an initial install or an upgrade. The following three procedures are documented in *UNICOS System Configuration Using ICMS*:

- An initial UNICOS 10.0 installation (with MLS)
- A UNICOS 9.3 system with MLS upgraded to a UNICOS 10.0 system with MLS
- A UNICOS 9.1 or earlier system without MLS upgraded to a UNICOS 10.0 system with MLS

The procedures for migrating to a Cray ML-Safe system configuration and migrating from a single-level UNICOS system to a multilevel UNICOS system are outlined in the following sections:

- A UNICOS 10.0 system with MLS to a Cray ML-Safe system configuration
- A single-level UNICOS system to a multilevel UNICOS system

8.7.9.1 Cray ML-Safe Configuration

A Cray ML-Safe configuration of the UNICOS system is established when the specified configuration of the UNICOS system is enabled. In order to install the UNICOS portion of a Cray ML-Safe configuration, the following documentation must be used, depending on your system:

- *UNICOS Installation Guide for Cray J90se and Cray SV1 Series GigaRing based Systems*
- *UNICOS Installation Guide for Cray T90 and Cray T90 IEEE Model E based Systems*
- *UNICOS Installation Guide for Cray T90 and Cray T90 IEEE GigaRing based Systems*

In addition, several software releases and related documentation are needed for the proper installation of other Cray ML-Safe software components. Installation manuals include:

- *NQE Administration*
- *UNICOS System Configuration Using ICMS*
- *OWS x.x Release and Installation Notes*
- *Support System and IOS-E Release Overview*
- *Cray Data Migration Facility (DMF) Release Overview and Installation Guide*
- *SWS-ION Release Overview*

The following procedures assume you have already installed and are running a UNICOS 10.0 system.

Before you configure a Cray ML-Safe system, you should make the following preparations:

- You must have the security administrator, system administrator, and system operator roles defined by assigning the correct category to the appropriate

site personnel before installing the Cray ML-Safe configuration. For more information, see Section 8.7.6, page 236.

- You should review your `crontab` jobs before migrating to a Cray ML-Safe configuration. The following example shows the type of change that should be made; failure to make these changes means that the jobs will fail on a Cray ML-Safe configuration, as they call commands that require privilege. For example, change the following in `root crontab` as shown:

  ```
  su sys -c "/usr/lib/sa/sa1"
  ```

  to

  ```
  setucat secadm >/dev/null; su sys -c "/usr/lib/sa/sa1"
  ```

  The `umask` setting in `skl/etc/profile` and `skl/etc/cshrc` must be copied to `/etc/profile` and `/etc/cshrc`, respectively, to ensure that the proper setting of 077 is used. For more information, see Section 8.3.1, page 170

- The trivial file transfer protocol (TFTP) must be disabled on a Cray ML-Safe configuration. To disable this feature, use the `Configure system->Network Configuration->TCP/IP Configuration->Generic Internet Daemon Configuration` selection in the UNICOS Installation and Configuration Menu System.

In addition to these preparations, the following products should be installed before installing a Cray ML-Safe configuration:

- If your site is running the Cray Data Migration Facility (DMF), refer to the *Cray Data Migration Facility (DMF) Release and Installation Guide* for installation instructions for a Cray ML-Safe configuration.
- If your site is using online tapes for nonadministrative users, the Cray/REELlibrarian (CRL) must be installed. CRL is not required for administrative-only access to tapes. Refer to the *Cray/REELlibrarian (CRL) Administrator's Guide* for installation instructions for a Cray ML-Safe configuration.
- On UNICOS systems using an IOS-E, install the operator interface (`opi`) in Cray ML-Safe mode. Refer to the *OWS Operator Workstation Administrator's Guide*, publication SG-3038, for installation instructions for a Cray ML-Safe configuration. The OWS-E interface must have the `admin` option set.

Use the following instructions to build and install a Cray ML-Safe configuration. Unless otherwise noted, these steps may be completed in any order prior to booting the Cray ML-Safe kernel. Cray recommends that this procedure be done in dedicated time.

1. If the system is in multiuser mode, go to single-user mode and unmount all file systems. Sync the file system and flush the ldcache (if any).

2. Label the `root`, `/usr`, `/tmp`, `spool`, `src`, the file system on which the job temporary directories reside, and the file system on which the security logs reside (by default, this is `/usr/adm/sl`, although it can be site-defined) with a `syslow` to `syshigh` label range by using the `labelit`(8) command, as shown in the following example. The `core` file system must be labeled with a `syshigh` label.

```
/etc/labelit -l syslow -u syshigh /dev/dsk/usr
/etc/labelit -l syslow -u syshigh /dev/dsk/tmp
/etc/labelit -l syslow -u syshigh /dev/dsk/spool
/etc/labelit -l syslow -u syshigh /dev/dsk/src
/etc/labelit -l syslow -u syshigh /dev/dsk/jtmp
/etc/labelit -l syslow -u syshigh /dev/dsk/usr_adm_sl
/etc/labelit -u syshigh /dev/dsk/core
/etc/labelit -l syslow -u syshigh /dev/dsk/root
```

Caution: Do not sync the file systems after this point.

3. Reboot your initial UNICOS 10.0 `PRIV_SU` system, which makes the new label range on the `root` file system effective. Go to multiuser mode.

4. As `root`, prepare for entering the Installation and Configuration Menu System by ensuring the `src` file system is mounted. If it is not mounted, mount it at this time, as shown in the following example:

```
/etc/mount /usr/src
```

   Set the `TERM` environment variable so you can run the Installation and Configuration Menu System, as shown in the following example:

```
export TERM=xterm
eval ‘resize‘
```

   Run the Installation and Configuration Menu System, as shown in the following example:

```
cd /etc/install
./install
```

5. For any product that is specified as `YES` in the UNICOS Installation and Configuration Menu System, the menu system must automate the configuration. To ensure that the settings are set to `YES`, check the

following product settings in the `Configure system->Configurator automation options` menu selection:

```
Major software configuration?              YES
Mainframe hardware configuration?          YES
IOS configuration?                         YES
Kernel configuration?                      YES
Multilevel security (MLS) configuration?   YES
Tape configuration?                        YES
Cray/REELlibrarian configuration?          YES
Host address configuration?                YES
Network address configuration?             YES
Services configuration?                    YES
Network interface configuration?           YES
Network hardware address configuration?    YES
TCP/IP configuration?                      YES
TCP/IP protocols configuration?            YES
TCP/IP lookup configuration?               YES
NFS configuration?                         YES
NQS configuration?                         YES
System daemons configuration?              YES
Startup (/etc/rc) configuration?           YES
```

6. If you have manually changed any configuration files since the last system build, or you are now automating a major configuration item from the previous step, you must import those files at this time. If the files are not imported, this information is lost. For instructions on importing UNICOS 9.3 configuration information, see *UNICOS System Configuration Using ICMS*.

7. In the next step, you make the necessary changes for a Cray ML-Safe configuration. Before doing so, check your NAL set configuration. Any NAL set name that has security ranges lower than `B1` must be removed. Change any of the NAL sets that are needed at this time. All NAL entries must be `B1` or greater on a Cray ML-Safe configuration.

8. Enable the Cray ML-Safe configuration by manually setting the following items in the Installation and Configuration Menu System.

In the `Configure System->Major software configuration` menu selection, make the following settings:

```
Kerberos network data encryption: OFF
Network Information Service (NIS): OFF
Cray based network monitor: OFF
Network testing tools: OFF
DCE Distributed File Service (DFS): OFF
File Transfer Agent (FTA): OFF
```

In the `Configure System->Multilevel security (MLS) configuration` menu selection, make the following settings.

System options:

```
Enforce system high/low security labels?  ON
  /tmp and /usr/tmp minimum security level: SYSLOW
  /tmp and /usr/tmp maximum security level: SYSHIGH
  Enforce strict device labeling rules?  ON
  Enforce socket usage for syslogd?  ON
  Super-user privilege policy?  OFF
```

Network security options:

```
Strict B1 evaluation rules: YES
  Default to multi-level privileged sockets for compatibility: NO
  Traditional hosts.equiv & .rhosts: NO
```

In the `Configure System->Multilevel security (MLS) configuration->Network security options->Network-Protocols Security Configuration` menu selection, make the following settings.

MLS Network Access List (NAL) Sets:

```
Security class: Must be B1, B2, B3, or A1. D, C1, and C2 are not evaluated.
```

MLS Network Security Definitions:

```
Item type: Must be either ip_host or ip_net. 'station' is not evaluated
```

In the `Configure System->Tape configuration->Select tape subsystem options` menu selection, make the following settings.

General options:

```
Front end servicing at startup: NO
  Allow unprotected tapes: NO
  Secure front end: NO
  Ask operator permission to switch label type: YES
```

Check options:

```
Check file identifier: YES
  Check protection flags: YES
```

Default options:

```
Is servicing front end mandatory?  NO
  Operator verify each scratch mount: YES
```

In the `Configure System->Network configuration->General network configuration->Network interface configuration` menu selection, make the following setting:

```
Address family: inet [afnet]
```

In the `Configure System->Network configuration->TCP/IP configuration` menu selection, make the following settings:

Host/address lookup:

```
Use Domain Name (DN) service?  NO
```

Kernel parameters:

```
IP forwarding?  NO
```

In the `Configure System->Network configuration->NFS configuration->List of exported file systems` menu selection, make the following setting:

```
Require Kerberos authentication (krb): <NULL> [cannot set to 'krb']
```

In the `Configure System->Ctartup (/etc/rc) configuration` menu selection, make the following setting:

```
Start system accounting?  NO
```

9. Exit the current menu by pressing `e`. Activate the configuration by using the `Configure system->Activate the configuration...` selection.

10. Build the new Cray ML-Safe kernel by using the `Build/Install System` menu and setting the build selections as shown in the following example:

```
M-> Build options ==>
    /usr/src reconfiguration files ==>
    Build action to take               install
    Build object                       all objects
    Components to build                specific component
    Major components selection ==>
    Specific component to build        uts
    Do the build in batch?                        NO
    NQS submission options ==>
    Do the build ...
    Restart the build ==>
    Review last build summary ...
    Escape to a chroot shell ...
```

    Execute the build by selecting the `Do the build ...` selection. This results in building and installing a Cray ML-Safe system configuration.

    Note: If you build and install any product on a Cray ML-Safe system, it must be done from within the Installation and Configuration Menu System. For products that are not currently supported on the menu system, the administrator must invoke the menu system and then move to a shell before attempting to build and install the product. This is necessary, because the base Installation and Configuration Menu System binary enables the necessary privileges to build and install the various UNICOS binaries. The following example shows the sequence:

```
# setucat secadm
# cd /etc/install
# ./install
```

    Then, from the Installation and Configuration Menu System, you would escape to a `chroot` shell as follows:

```
M-> Build/Install System ==>
       A-> Escape to a chroot shell ...
```

11. Transfer the Cray ML-Safe kernel to the workstation (SWS, OWS, or console) by hand. For lists of files and their locations, see *UNICOS System Configuration Using ICMS*. Keep the old `PRIV_SU` kernel on the workstation. If the Cray ML-Safe kernel will not boot into multiuser mode, the old `PRIV_SU` kernel can be booted into single-user mode to aid in analyzing the problem. Events such as incorrect kernel configuration or incorrect

or missing system file labels and PALs can prevent the booting of a Cray ML-Safe kernel.

12. Escape to the shell by using the ! command.

13. Configure NQS by using the NQE configuration tool instead of the Installation and Configuration Menu System. Configurable MLS features are disabled by default. For information on configuring NQS, see *NQE Administration*.

    The following list summarizes the configuration changes for NQS on a Cray ML-Safe configuration:

    - Set the following parameters in the `/etc/nqeinfo` configuration file:

      ```
      NQE_NQS_MAC_COMMAND        1
      NQE_NQS_MAC_DIRECTORY      1
      NQE_NQS_PRIV_FIFO          1
      ```

    - Ensure that NQS user validation is set to either `password` or `file` (`no_validation` is not acceptable for a Cray ML-Safe configuration). Edit the NQS configuration file and search for the `set validation` configuration command. The argument must be either `password` or `file`. If the argument is `no_validation`, change it to `password` or `file`.

    - Ensure that the NQS configuration does not use FTA as an output agent. Edit the NQS configuration file text and search for `output_agent`. For each machine ID that currently has FTA as an output agent, delete the FTA line.

    - Ensure that all the NQS spool directories, logfile, and console file are on a file system with a `syslow` to `syshigh` label range.

      The Installation and Configuration Menu System should be set as follows:

      ```
      Configure system ==>
        Multilevel security (MLS) configuration ==>
          System Options ==>
            Enforce system high/low security labels ==>  ON
      ```

    - Convert all NQS wildcard directories to MLDs. The `nqsdaemon` must not be running when you convert the directories. For information on converting directories, see *NQE Administration*.

The following is a list of directories that must be converted (NQE_NQS_SPOOL is defined in /etc/nqeinfo):

```
NQE_NQS_SPOOL/private/requests
NQE_NQS_SPOOL/private/root/chkpnt
NQE_NQS_SPOOL/private/root/control
NQE_NQS_SPOOL/private/root/data
NQS_SPOOL/private/root/failed
NQE_NQS_SPOOL/private/root/interproc
NQE_NQS_SPOOL/private/root/output
NQE_NQS_SPOOL/private/root/reconnect
```

14. If you have not already done so, all wildcard labeled directories must be converted to MLDs. The following list contains the directories that require the use of MLDs on a Cray ML-Safe configuration:

    - /usr/mail
    - /usr/spool/mqueue
    - /usr/spool/cron/crontabs
    - /usr/spool/cron/atjobs
    - lpr(1) and lpd(8) spool directories (/usr/spool/*)
    - /tmp
    - /usr/tmp

    If you want to preserve existing job temporary directories and carry them forward to the new system, then each existing job temporary directory must be converted into a MLD under the /tmp.mld/jtmp directory.

    Note: The conversion of /tmp and /usr/tmp must be done in single-user mode, as some of the daemons use this directory while in multiuser mode.

    See Section 8.4.1.2.4, page 179, for more information on converting the directories.

15. The umask setting as shown in skl/c1/etc/profile and skl/c1/etc/cshrc must be copied to /etc/profile and /etc/cshrc, respectively. The umask setting for a Cray ML-Safe configuration must be 077.

16. The device labels on the various tty lines to the OWS are assigned by getty(8) via the /etc/inittab file. The file must be set as shown in the following example. Any other tty line that is in the /etc/inittab file

should be changed with the `getty -L -C` command, as shown in the following example:

```
co::respawn:/etc/getty -L 0-16 -C 0-077777777777777777777
console console
01:2:respawn:/etc/getty -L 0-16 -C 0-077777777777777777777
tty01 9600
02:2:respawn:/etc/getty -L 0-16 -C 0-077777777777777777777
tty02 9600
```

Cray platforms support several lines to the OWS. Access to these devices is controlled through `opi` in the Cray ML-Safe mode from the OWS side and by the device labels on the UNICOS side.

17. Change the label range on the local host interface from `syslow` to `syshigh` and 0 compartments to all available compartments. Go to the `Configure system->Network configuration->General Network Configuration->Network interface configuration selection` and select the `localhost (lo0)`. Change the following items: Minimum security label to `syslow`; Maximum security label to `syshigh`; Maximum security compartments to `077777777777777777777`.

18. Ensure that the `root`, `/usr`, `/src`, and administrative file systems have been mounted, as follows:

```
/etc/mount /usr
/etc/mount /usr/src
/etc/mount /usr/spool  # if site has one
/etc/mount /usr/adm    # if site has one
/etc/mount /usr/adm/sl # if site has one
```

19. Two objects are not included in the base Cray ML-Safe database file (`/etc/privdb/mls.db`). If the following lines are not already included in `/etc/privdb/mls.db` (and if your site has the `/usr/src/cmd/sp/privdb/mls.db` file), add them, as follows:

```
file = { name = /etc/privdb/gen.db; MAC = [syslow,none]; };
#if exists(/usr/spool/logs/airlog)
file = { name = /usr/spool/logs/airlog; MAC = [syshigh,none]; };
#endif
```

20. Run the `privcmd` command to label system files with the file privilege states, permission modes, owner, and group, and to assign privilege assignment lists (PALs) and security labels, as shown in the following example:

```
/etc/privcmd
```

Note: Any time you change (or add to) your system configuration, you must execute the `privcmd` command to label these files. This step is required for rebuilds also.

If your current UNICOS system was not configured with the `syshigh` and `syslow` labels (`SECURE_MAC`), you will see error messages, such as `WARNING: syshigh/syslow MAC labels are not set.` Ignore these messages at this time.

21. Protect the NFS ID maps and the scripts that produce the maps (see page 1291, for more information).

    Every script or program used in ID map generation must have a `syslow` label. These scripts and programs must be executed at the `syslow` label, and the resulting map files must have a `syslow` label. The NFS commands in the `/etc/uidmaps` directory and the directory itself are automatically installed with the `syslow` label. Add the `syslow` label to locally-written scripts. When `cron`(8) is used to execute the scripts, ensure that the `cron` job is set to run with the `syslow` label. Existing ID map files are overwritten by the `nfsmerge`(8) command without changing their labels; you should remove all existing ID map files at the start of the ID map generation process.

22. Shut down the currently running UNICOS `PRIV_SU` system

23. Boot the Cray ML-Safe system configuration, and stay in single-user mode.

24. Disable auditing by executing the `/etc/spaudit -d state` command.

25. Execute the following sequence of commands:

    ```
    /etc/mount /usr
    /etc/mount /usr/src
    /etc/mount /usr/spool  # if the site has one
    /etc/mount /usr/adm    # if the site has one
    /etc/mount /usr/adm/sl # if the site has one
    /etc/privcmd
    ```

    This sequence of commands must be executed because the various objects labeled by `privcmd` do not have the correct security label unless the `SECURE_MAC` configuration option is enabled.

26. Reenable auditing by executing the `/etc/spaudit -e state` command.

27. Unmount all file systems, execute the `sync` command, and reboot the Cray ML-Safe configuration, as follows:

```
cd /
/etc/umountem
sync
sync
sync
/etc/ldsync
<reboot>
```

28. Go to multiuser mode by executing the `init 2` command.

8.7.9.2 Single Level UNICOS System to a Multilevel UNICOS System

Use the following procedure to convert your single-level UNICOS system to a multilevel UNICOS system:

1. Configure and build a UNICOS kernel. This can be done during normal production hours. Enter the menu system as follows:

```
cd /etc/install
./install
```

2. To administer your UNICOS MLS configuration options using the UNICOS Installation and Configuration Menu System, enable it by setting the `Configure system->Configurator automation options->Security configuration option` to `YES`.

3. Import the default security level names by selecting `Configure system->Multilevel Security (MLS) configuration->Import the security configuration option` in the UNICOS Installation and Configuration Menu System.

4. Set up all new site-defined security level and compartment names at this time so you do not have to rebuild the kernel later. To define new security level names, use the `Configure system->Multilevel security (MLS) configuration->Site labels configuration->Levels` selection in the UNICOS Installation and Configuration Menu System. To define new security compartment names, use `Configure system->Multilevel security (MLS) configuration->Site labels configuration->Compartments` selection in the UNICOS Installation and Configuration Menu System.

5. Set all the necessary configuration parameters found in the `Configure system->Multilevel security (MLS) configuration` menu in the UNICOS Installation and Configuration Menu System.

   One of the most important selections is the system management mechanism. Cray recommends enabling the `PRIV_SU` mechanism, which means the super-user policy is enforced. Enable this option by using the `Configure system->Multilevel security (MLS) configuration->System options->Super-user privilege policy?` selection in the UNICOS Installation and Configuration Menu System.

6. Configure the security levels and compartments on the networking interfaces found by using the `Configure system->Network configuration->General network configuration->Network interface` selections in the UNICOS Installation and Configuration Menu System.

   If these interfaces and routes are not labeled, the authorized security levels and compartments of the users connecting to the UNICOS system are set to 0 and none, respectively.

7. When all the configuration options have been set, build the kernel by activating the security and system configuration. Activating the configuration, which updates the appropriate source files, must be done before building the UNICOS kernel. To activate the configuration, use the `Configure system->Activate the configuration` selection in the UNICOS Installation and Configuration Menu System.

   Each time a configuration is activated, the Installation and Configuration Menu System determines which files are affected by the changes. The following message is then displayed:

   ```
   Do you want to proceed with the configuration update (y/n)?
   ```

   Respond by typing `y`.

8. Build the kernel. This involves three selections in the `Build/install system` selection of the UNICOS Installation and Configuration Menu System. First, use the `Build/install system->Components to build` selection and set to selected major components. Then use the `Build/install system->Components to build` selection to set all selections to `NO`, except for the UNICOS kernel selection, which must be set to `YES`. Last, use the `Build/install system->Do the build` selection to rebuild the kernel selection.

9. Transfer the newly built UNICOS kernel to the boot medium.

10. Label all file systems that are to be mounted under the UNICOS kernel. At a minimum, these file systems must be labeled by using the `labelit`(8) command, as shown in the following example:

    ```
    /etc/labelit -s -u 0 -l 0 -c 0 /dev/dsk/user1
    ```

    Note: You must use the `-u`, `-l`, and `-c` options when using the `-s` option of the `labelit` command.

    If your site uses nonzero levels and compartments, then use the following guidelines for labeling file systems:

    Note: If `SECURE_MAC` is enabled, then the range of labels should be `syslow` to `syshigh`.

    - The `root` file system must be labeled with all the security levels and compartments that are assigned to users on the UNICOS system. This is necessary because pipes are usually allocated through the `root` device. Also, tty devices change labels to match that of the user through the `root` device.
    - The file system that contains the NQS, mail files, and line printer queues must be labeled with all the security levels and compartments assigned to users on the UNICOS system.
    - The `/usr/tmp` and `/tmp` files must also be labeled with all the security levels and compartments that are assigned to users on the UNICOS system.
    - If the system is configured with `SECURE_MAC` is enabled and `syslow` through `syshigh` security labels, you must ensure the core file system (where dumps are to be written) is authorized for `syshigh`.
    - If the system is configured with `SECURE_MAC` enabled and `syslow` through `syshigh` security labels, you must ensure the job temporary file system (if it differs from `/tmp`) is authorized for `syslow`.

11. Your site may need to specify a maximum security level and authorized compartments values that are nonzero. For example, if your site plans on using security levels 0 through 16 and compartment 04700, then your `root` file system must be labeled as follows:

    ```
    /etc/labelit -s -l 0 -u 16 -c 04700 /dev/dsk/root
    ```

Another example is if the system is configured with SECURE_MAC enabled, and syslow through syshigh security labels, then the root file system must be labeled as follows:

```
/etc/labelit -s -l syslow -u syshigh -c 07777 /dev/dsk/root
```

You cannot label the root file system while mounted, which means you must boot an alternate root file device. Label the original root file system as previously described in this procedure and boot this labeled root file system.

12. All user file systems must be correctly labeled, as shown in the following example:

    ```
    /etc/labelit -s -l 0 -u 3 -c 01000 /dev/dsk/user1
    ```

    Boot the new UNICOS kernel into single-user mode. For more information about booting into single-user mode, see the installation guide for your Cray system.

13. Disable auditing in single-user mode as shown in the following example:

    ```
    /etc/spaudit -d state
    ```

    If you run a PRIV_SU and PALs system, then execute the privcmd command, as shown in the following example:

    ```
    /etc/privcmd
    ```

14. Unmount all file systems mounted in the previous step.

15. Create or update a UDB login for the security administrator. See Section 8.7.6, page 236, for more information on setting up this entry. Depending on the privilege mechanism(s) used, the security administrator login should be given the additional UDB fields with the following values:

    - For PRIV_SU systems: permits:suidgid:;
    - For systems with PALs: valcat:secadm:

Ensure that the chown permbit is assigned to the security administrator account, as shown in the following example:

```
/etc/udbgen
udbgen: 1>update:user:
udbgen: 2>valcat:secadm:maxcls:0:
udbgen: 3>permits:suidgid:
udbgen: 4>permbits+:chown:
udbgen: 5>quit
Updated 1 record
```

Additionally, if the security administrator is to reclassify user files, assign all the security levels and compartments for the system to the security administrator account.

16. For UNICOS systems, the system console device entries under the /dev directory must be labeled. The getty -L and getty -C commands specify the security level and compartment ranges, respectively, for the system console devices, /dev/console, /dev/tty00, /dev/tty01, and so on. Additionally, you can use the getty -m command to indicate that the specified console device is a multilevel device.

    If SECURE_MAC is enabled, label the console to syslow.

    The getty command for these devices are specified in the /etc/inittab file, as shown in the following example:

```
co::respawn:/etc/getty -L 0-16 -C 0-077777777777777777777 console console
01:23:respawn:/etc/getty -L 0-16 tty01 9600
02:23:respawn:/etc/getty -L 0-5 tty02 9600
```

17. If TCP/IP network access is to be permitted to the UNICOS system (for example, telnet or ftp), entries for the following Internet address(es) must appear in the network access list (NAL):

    - Localhost (127.0.0.1)
    - Local network interfaces for all directly-connected networks
    - Network interfaces on the OWS (if network access is permitted to the UNICOS system from the OWS-E).

18. On a UNICOS system, a default NAL entry controls the label given to a host not specified in the kernel NAL table. If a default entry is not in the NAL, then only specified hosts are allowed to connect to the UNICOS system. Add the default entry to the NAL if you want hosts not specified in the kernel NAL table to connect to the UNICOS system. The default entry can

be added by using the Configure System->Multilevel Security (MLS) Configuration->Network Security Options->Network Protocol Security Configuration->MLS Network Access List (NAL) Sets and MLS Network Security Definitions selection in the UNICOS Installation and Configuration Menu System. The default entry host name to use is default.

The following is the NAL definition from the file /etc/config/spnet.conf, which is generated by the Installation and Configuration Menu System. In this example, the default entry is allowed only a nonzero label connection.

```
ip net "default" {
    name = "%NBY";
    class = B1;
    max label = level0, 0;
}
```

See the spnet(8) man page for more information on setting up the NAL.

19. If used, the workstation access list (WAL) entries for the UNICOS system should also be generated. The WAL is not required and may be configured at a later time. To configure the WAL entries at this time, use the Workstation Access List (WAL) Sets selections in the UNICOS Installation and Configuration Menu System. See the spnet(8) man page for more information on setting up the WAL.

20. If the local networks on the site support the Commercial IP Security Option (CIPSO), you must create the security label translation tables. The CIPSO Map Domain Sets selection of the UNICOS Installation and Configuration Menu System defines the maps, and the Network Security Definitions selections associate CIPSO hosts with their CIPSO maps. See the spnet(8) man page for more information on setting up the CIPSO maps.

21. The security administrator must also define each user in the UDB. Each UDB entry defines the active and authorized security levels, active category, authorized categories, active and authorized compartments, and permissions. Proper definition of a user in the UDB is critical to maintaining a Cray ML-Safe configuration of the UNICOS system. Ensure that each user is given only those levels, categories, compartments, and permissions that are absolutely necessary.

    The security administrator can use the nu(8) or udbgen(8) commands to create or modify UDB entries. (The udbgen command does not perform all the initialization tasks that the nu command performs.) This can be done in

single-user mode, but to do so successfully, mount all file systems that the nu command needs to create the user login directories. If mounted, these file systems should be unmounted before going to the next step.

It is not recommended that the system be put in multiuser mode at this point, but bringing up multiuser mode without the network is an alternative to running the nu command in single-user mode.

Set proper security labels on the home directories of users with a nonzero default level and/or a nonnull default compartment set. This can be done by using the `spset -l` and `spset -c` command.

The nu command creates and initializes a home directory for each new user, but it cannot set it to a nonzero label. If users try to log in they will not be able to access their home directory unless it has been changed from its nonzero status. Any other files associated with the user's home directory may also require new label settings if the user is allowed to access them (for example, `.cshrc` or `.profile`). Files created after a user has logged in are automatically set with the correct security label.

22. Determine if MLDs are to be used and which directories must be converted to MLDs. Sites can use all wildcards, all MLDs or a mix of both. If `mail` is to be used in a nonzero label environment, convert the `/usr/mail` and `/usr/spool/mqueue` directories to MLDs.

    If `cron`(1), `at`(1), or `lpr`(1)) are to be used in a nonzero label environment, then their corresponding spool directories must be set up as MLDs. See Section 8.4.1.2.2, page 175, for more information. For information on using MLDs for NQS, see *NQE Administration*.

23. Prior to booting to multiuser mode, reenable auditing, as shown in the following example:

    ```
    /etc/spaudit -e state
    ```

    Go to multiuser mode by doing the following:

    ```
    /etc/init 2
    ```

    After entering multiuser mode, you can change the UNICOS MLS configuration parameters, but the kernel must be rebuilt and rebooted after the changes are made.

8.8 MLS Auditing on a UNICOS System

MLS auditing on a UNICOS system consists of collecting data on security-relevant events. As shown in Figure 10, the kernel determines whether an auditable event should be audited based on the audit selection criteria. These criteria are saved in the low memory table in the kernel. The initial settings of the criteria are defined by the configuration parameters, which are set by the UNICOS Installation and Configuration Menu System; these settings can be changed by using either the UNICOS Installation and Configuration Menu System or `spaudit`(8). If an event is being audited, when the event occurs, a record is written to the security log pseudo device, `/dev/slog` (see `slog`(4)).

The security logging daemon (`slogdemon`(8)) reads `/dev/slog` and writes the records to the disk-resident security log (`/usr/adm/sl/slogfile`). The security administrator can generate (by using the `reduce`(8) command) the audit records from the security log to produce information on how the Cray ML-Safe configuration of the UNICOS system is being used.

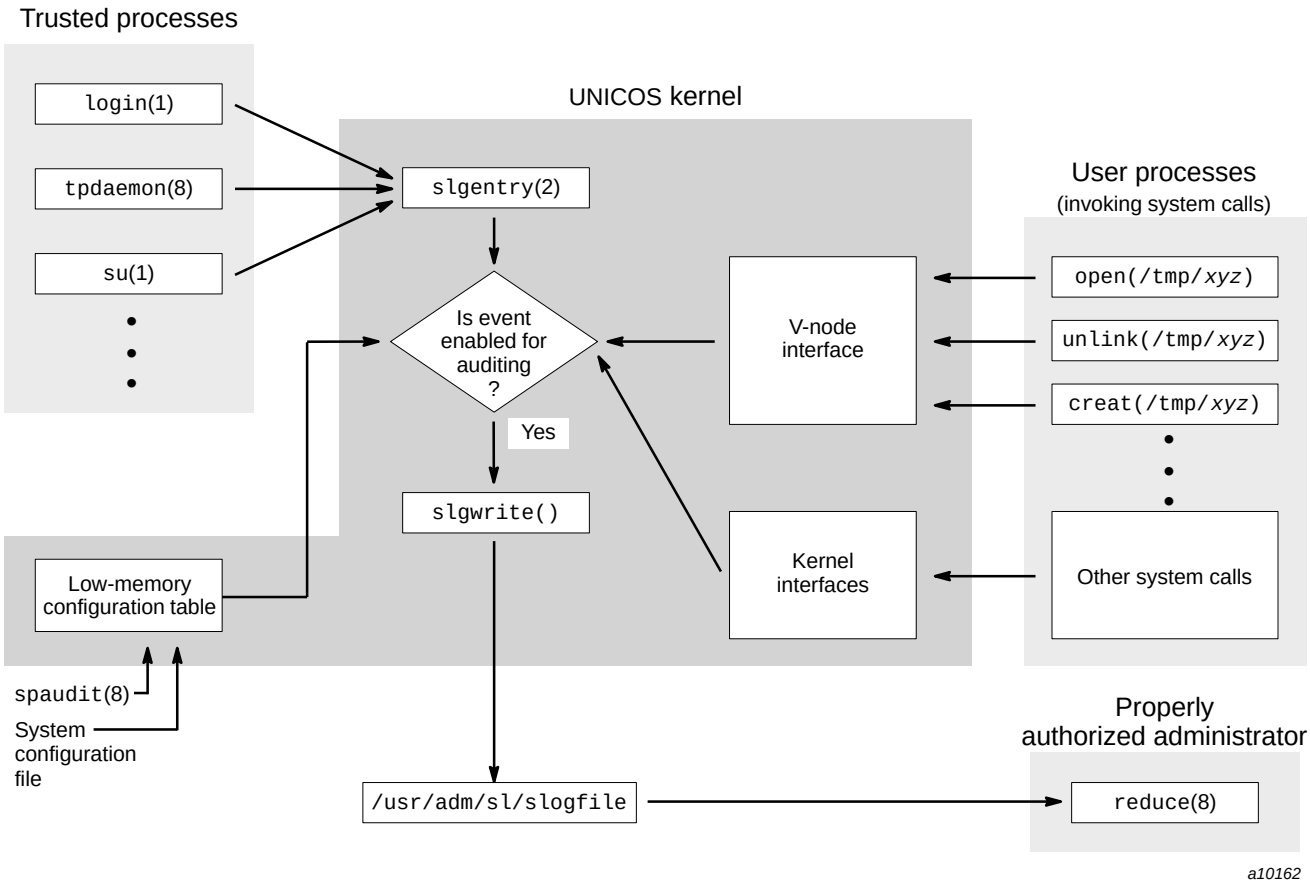


Figure 10. Overview of Security Auditing

For UNICOS systems using the PAL-based privilege mechanism, the security log is protected with a `syshigh` label and you must be an authorized user with the `exec` privilege text to use the `reduce` command to access the log.

The UNICOS system maintains a lockout parameter that is common to the security log device driver to ensure activation of only a single security log file and daemon.

The following sections explain auditing by providing the following information:

- Description of the security log and how to enable and configure it
- Description of the security log daemon

- How to enable the type of events to be recorded in the log
- Description of the security log record types
- Examples of how to use the `reduce` command

8.8.1 Security Log Overview

The `/etc/slogdemon` (see `slogdemon`(8)) program is the system security logging daemon. The daemon collects security log records from the operating system by reading the security log pseudo device, `/dev/slog` (see `slog`(4)), and writing the records to a disk-resident security log (`/usr/adm/sl/slogfile`).

`/dev/slog` is a read-only pseudo device that buffers security log records. By default, it holds approximately 1000 security log records (see the description of `SLG_BUFSIZE` later in this section for more information on defining the size of `/dev/slog`).

If `/dev/slog` becomes more than 50% full, each process that tries to generate audit records is tested to see if it matches one of the following conditions:

- If the buffer is more than 50% full and `slogdemon` is not running, all nonadministrative processes are put to sleep.
- If the buffer is more than 87% full, all nonadministrative processes are put to sleep.
- If the buffer is 93% full, all processes, except for the security log daemon and the idle process are put to sleep.

Once the buffer is emptied, a wakeup is sent to the sleeping processes.

If standard system buffering is done through `ldcache`, and the system panics, the flush-on-panic routine, if it is enabled, attempts to flush the `ldcache` buffers and system buffers to disk. Repeated attempts are made to flush buffers to disk. Because of this flushing mechanism, the potential for losing records is decreased, but records can still be lost under the following set of circumstances:

- When records are being transferred into a system buffer by a `write`(2) system call.
- When records are being transferred out of a system buffer.

The situation that provides the greatest chance for losing records is when the records reside in `/dev/slog` and the flush-on-panic routine is not enabled. The number of lost records depends on the size of `/dev/slog`. If the default setting

is used, then approximately 1000 records would be lost. /dev/slog can be read from a system dump by using the slog and rslog commands of crash(8).

If the security logging daemon is not running, /dev/slog eventually fills up, and all processes that require security log entries sleep while waiting for the device to be emptied by the daemon. The amount of processing that can be accomplished when the daemon is not active depends upon the size of /dev/slog and the volume of security log data being sent to it. Even if the security log daemon is running, the size of /dev/slog can be configured too small and a system panic or hang can result.

The parameters that define the state, location, and size of the security log are found in uts/cf.SN/config.h, and are as follows:

Parameter	Description
SLG_STATE	Defines if the security log is enabled. By default, the security log is ON.
SLG_DIR	Defines the disk-resident security log directory for both active and retired security logs. The default is /usr/adm/sl.
SLG_FILE	Defines the file name, relative to SLG_DIR, of the active disk-resident security log file (that is, SLG_DIR/SLG_FILE defines the file name for the active security log file). The default is slogfile.
SLG_FPREFIX	Defines the file name prefix, relative to SLG_DIR, of the retired disk-resident security log file (that is, SLG_DIR/SLG_FPREFIX defines the file name prefix for the retired security log files). The default is s.
SLG_MAXSIZE	Defines the threshold value, which once reached, causes the disk-resident security log to be retired. The default is 8,192,000 bytes.
SLG_BUFSIZE	Defines the size of /dev/slog; the default is 163,840 bytes.

The SLG_STATE parameter allows you to enable or disable the security log. If disabled, all other security logging options are ignored. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Enable security logging? selection in the UNICOS Installation and Configuration Menu System.

The SLG_DIR parameter allows you to specify the directory in which both active and retired disk-resident security logs are kept. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Directory for security logs` selection in the UNICOS Installation and Configuration Menu System.

The SLG_FILE parameter allows you to specify the actual name (within the directory specified by SLG_DIR) of the active disk-resident security log. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Filename of active security log` selection in the UNICOS Installation and Configuration Menu System.

The SLG_FPREFIX parameter allows you to define the file name prefix (within the directory specified by SLG_DIR) of a disk-resident security log file that is archived. The path to the archived security log file must reside on the same file system as SLG_FILE. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Prefix for retired security logs` selection in the UNICOS Installation and Configuration Menu System.

The SLG_MAXSIZE parameter defines the threshold value (in bytes), that once reached, causes the disk-resident security log to be retired. The actual maximum size of the security log may be larger than SLG_MAXSIZE by a small amount (less than 1024 bytes). It is recommended to make SLG_MAXSIZE no smaller than the default; increments should be made in 4096-word blocks.

When the log reaches SLG_MAXSIZE, it is renamed with a SLG_DIR/SLG_FPREFIX*yymmddhhmmss* name, where *yymmddhhmmss* is the time of the renaming, and a new log file is created. The security / system administrator must archive the renamed logs promptly to prevent the file system containing the security log from becoming full. If the system runs out of disk space for the security log, `/dev/slog` eventually fills, causing the system to stop.

To set the SLG_MAXSIZE parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Maximum log file size` selection in the UNICOS Installation and Configuration Menu System menu.

The SLG_BUFSIZE parameter defines the size of `/dev/slog`. The default size is 163,840 bytes (approximately 1000 security log records). The default size should not be decreased unless your site has never enabled auditing. In this case, SLG_BUFSIZE can be set to 0. If the SLG_ALL_NAMI or SLG_ALL_VALID configuration parameters are enabled, the default should be increased by a

factor of two or three. Any increase to the default should be made in 8-byte increments (that is, multiples of 8).

To set SLG_BUFSIZE, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log buffer size selection in the UNICOS Installation and Configuration Menu System.

8.8.2 Security Logging Daemon

The security logging daemon is capable of running in all states (that is, single-user mode, multiuser mode, and so on). The default is for the daemon to run only in multiuser mode.

The security logging daemon should be initiated through inittab(5) or the /etc/rc script. At installation time, the install scripts create the pseudo device for /dev/slog. If it is necessary to recreate it, refer to the mknod(8) man page. The major device number is 20 and the minor device number is 0 for /dev/slog.

To restart slogdemon in multiuser mode, locate the process ID of the slogdemon process. Then, send a termination signal (15) to the process and restart the daemon, as shown in the following example:

```
kill -15 ‘cat /etc/slogd.pid‘ && sleep 1 && /etc/slogdemon
```

If the daemon is started manually, then it must be done by root (UID = 0). The security administrator and appropriate operators should be the only users who know the password for this login ID.

8.8.3 Security Logging Daemon in Single-user Mode

There are two methods to initiate the security logging daemon in single-user mode. The first method is to add a command to inittab to mount all file systems in the path to the security log file and then start the daemon. This is shown in the following example. In this example, a separate sl file system contains the security log; where the files are actually placed is site-dependent:

```
slg::sysinit:/etc/mount /dev/dsk/sl /etc/sl&&/etc/slogdemon #start slog
```

Ensure that the path to the security log in inittab is the same as specified by the security log configuration parameter in uts/cf.SN/config.h. To do this, set the SLG_DIR parameter by using the Configure system->Multilevel security (MLS) configuration->Security

`log file configuration->Directory for security logs` selection in the UNICOS Installation and Configuration Menu System.

The second method is to start the daemon manually. To do this, you must first mount the file system and then start `slogdemon`. Before going to multiuser mode, you must kill `slogdemon` and unmount the file system. If you restart the kernel in single-user mode, you must first kill `slogdemon` and unmount the file system.

8.8.4 The `spaudit` Command

The `spaudit`(8) command allows a properly authorized administrator to change the security auditing criteria while the UNICOS system is running. This command can also be used to change the security log edition.

The `spaudit -e` *enableopts* command enables the specified security logging configuration option(s). The `spaudit -d` *disableopts* command disables the specified security logging configuration option(s). The valid options are as follows:

Option	Description
`all_nami`	Logs all `mkdir`, `rmdir`, `link`, and `rm` calls
`all_rm`	Logs all remove requests
`all_valid`	Logs all access requests
`audit`	Logs all security auditing criteria changes
`chdir`	Logs all change directory requests
`config`	Logs all UNICOS configuration changes
`crl`	Logs Cray/REELlibrarian activity
`dac`	Logs discretionary access changes
`discv`	Logs discretionary access violations
`filexfr`	Logs all file transfer requests
`ipnet`	Logs all network violations
`jend`	Logs end of job
`jstart`	Logs start of job
`linkv`	Logs all link (`ln`) violations
`mandv`	Logs mandatory access violations
`mkdirv`	Logs all make directory (`mkdir`) violations
`netcf`	Logs network configuration changes

netwv	Logs network violations (not currently used)
nfs	Logs all NFS activity
nqs	Logs NQS activity
nqscf	Logs NQS configuration changes
object_path	Logs object's full path name on accesses
operator	Logs operator actions
priv	Logs use of privilege
removev	Logs all remove violations
rmdirv	Logs all remove directory (rmdir) violations
secsys	Logs all security system call requests
setuid	Logs all setuid requests
shutdown	Logs system shutdown requests
startup	Logs system startup requests
state	Defines if security log is on (1) or off (0)
sulog	Logs all su attempts
tapes	Logs tape activity
time_change	Logs system time change
trust	Logs Cray ML-Safe process activity
user	Logs user password for failed login attempts

The command can be used in either single-user or multiuser mode to enable / disable security logging options. See the spaudit(8) man page for more information on using this command.

8.8.5 Security Logging Configuration Parameters

The auditing of all security-relevant events can be configured on a UNICOS system. The following list summarizes the configuration parameters that enable / disable the logging of security events:

Parameter	Description (default setting)
SLG_DISCV	Logs discretionary access violations (on)
SLG_MANDV	Logs mandatory access violations (on)
SLG_NETWV	Currently not used
SLG_MKDIRV	Logs all make directory (mkdir) violations (on)

SLG_RMDIRV	Logs all remove directory (`rmdir`) violations (on)
SLG_LINKV	Logs all link (`ln`) violations (on)
SLG_ALL_RM	Logs all remove (`rm`) requests (off)
SLG_REMOVEV	Logs all remove violations (on)
SLG_ALL_NAMI	Logs all `mkdir`, `rmdir`, `link`, and `rm` calls (off)
SLG_ALL_VALID	Logs all access requests (off)
SLG_ALL_NETW	Currently not used
SLG_PHYSIO_ERRORS	Currently not used
SLG_CF_NET	Logs all network configuration changes (off)
SLG_FILEXFR	Logs all file transfers (off)
SLG_PATH_TRACK	Tracks all path names on accesses (on)
SLG_SULOG	Logs all `su` attempts (on)
SLG_NFS	Logs all Cray-to-Cray NFS requests (off)
SLG_CF_UNICOS	Logs UNICOS configuration changes (off)
SLG_CF_NQS	Logs NQS configuration changes (off)
SLG_JSTART	Logs start of job (on)
SLG_JEND	Logs end of job (on)
SLG_SUID_RQ	Logs all setuid requests (on)
SLG_USER	Logs name and password on login password failure (off)
SLG_ACT_NQS	Logs NQS activity (off)
SLG_T_PROC	Logs Cray ML-Safe process activity (off)
SLG_LOG_PRIV	Logs use of privilege in system calls (off)
SLG_LOG_AUDIT	Logs all changes of audit criteria (off)
SLG_LOG_CHDIR	Logs all change directory requests (off)
SLG_LOG_CRL	Logs Cray/REELlibrarian activity (off)
SLG_LOG_DAC	Logs discretionary access changes (on)
SLG_LOG_IPNET	Logs all network violations
SLG_LOG_OPER	Logs operator actions (off)
SLG_LOG_SECSYS	Logs security system calls (off)
SLG_LOG_STARTUP	Logs system startup (off)
SLG_LOG_SHUTDWN	Logs system shutdown (off)
SLG_LOG_TAPE	Logs tape activity (off)

SLG_LOG_TCHG	Logs system time change (off)

8.8.6 Security Log Record Types

There are different records that can be logged when the security log is enabled (SLG_STATE = ON). The types of records are summarized in the following list:

- System start record (SLG_GO)
- System logging stop record (SLG_STOP)
- System configuration change record (SLG_CCHG)
- System time change record (SLG_TCHG)
- Discretionary access violation record (SLG_DISC_7)
- Mandatory access record (SLG_MAND_7)
- Operational access record (SLG_OPER)
- Login validation record (SLG_LOGN)
- Tape activity record (SLG_TAPE)
- End-of-job record (SLG_EOJ)
- Change directory record (SLG_CHDIR)
- Security system call record (SLG_SECSYS)
- Discretionary access change record (SLG_DAC_CHNG)
- setuid system call record (SLG_SETUID)
- su attempt record (SLG_SU)
- File transfer logging record (SLG_FXFR)
- Network security violations record (SLG_IPNET)
- Cray NFS request record (SLG_NFS)
- Network configuration change record (SLG_NETCF)
- Audit criteria selection change record (SLG_AUDIT)
- NQS configuration change record (SLG_NQSCF)
- NQS activity record (SLG_NQS)

- Cray ML-Safe process activity record (SLG_TRUST)
- Use of privilege record (SLG_PRIV)
- Cray/REELlibrarian activity record (SLG_CRL)

Table 10 summarizes the record types, the configuration parameter that enables the generation of the record, the spaudit argument that enables generation of the record, and the reduce option used to view the record.

Table 10. Security Log Records

Record type	Configuration parameter	spaudit -e	reduce -t
SLG_GO	SLG_LOG_STARTUP	startup	go
SLG_STOP	SLG_LOG_SHUTDWN	shutdown	stop
SLG_CCHG	SLG_CF_UNICOS	config	cchg
SLG_TCHG	SLG_LOG_TCHG	time_change	tchg
SLG_DISC_7	SLG_DISCV	discv	disc
SLG_MAND_7	SLG_MANDV and/or SLG_ALL_VALID	mandv, all_valid	
SLG_OPER	SLG_LOG_OPER	operator	oper
SLG_LOGN	SLG_JSTART	jstart	logn
SLG_TAPE	SLG_LOG_TAPE	tapes	tape
SLG_EOJ	SLG_EOJ	jend	eoj
SLG_CHDIR	SLG_PATH_TRACK and SLG_LOG_CHDIR	object_path, chdir	chdir
SLG_SECSYS	SLG_LOG_SECSYS	secsys	secsys
SLG_NAMI	SLG_MKDIR, SLG_RMDIRV, SLG_LINKV, SLG_REMOVEV, SLG_ALL_RM, and/or SLG_ALL_NAMI	mkdirv, rmdirv, linkv, removev, all_rm, all_nami	nami
SLG_DAC_CHNG	SLG_LOG_DAC	dac	dac
SLG_SETUID	SLG_SUID_RQ	setuid	setuid
SLG_SU	SLG_SULOG	sulog	sulog
SLG_FXFR	SLG_FILEXFR	filexfr	xfer

Record type	Configuration parameter	spaudit -e	reduce -t
SLG_IPNET	SLG_LOG_IPNET	ipnet	netip
SLG_NFS	SLG_NFS	nfs	nfs
SLG_NETCF	SLG_CF_NET	netcf	netcf
SLG_AUDIT	SLG_LOG_AUDIT	audit	audit
SLG_NQSCF	SLG_CF_NQS	nqscf	nqscf
SLG_NQS	SLG_ACT_NQS	nqs	nqs
SLG_TRUST	SLG_T_PROC	trust	trust
SLG_PRIV	SLG_LOG_PRIV	priv	priv
SLG_CRL	SLG_LOG_CRL	crl	crl

Assigning the usrtrap permission to a user or the trapr or trapw flags to an object overrides any security logging configuration parameter setting except for SLG_STATE.

The SLG_PATH_TRACK parameter allows the logging of all path names on access requests. No specific record is used when this parameter is enabled, but instead it forces more data to be stored with each access type record generated by the other parameters. See Section 8.8.7.6, page 276, for an example of the information produced (see the example in which the -p option is used). This parameter must be enabled in order for the reduce -p command to work correctly.

To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Track all pathnames on accesses? selection in the UNICOS Installation and Configuration Menu System.

The use of this parameter means that all chdir(2) requests must also be logged. See Section 8.8.7.12, page 297, for more information on enabling this record type.

If object path tracking is required for auditing of file manipulations that are being done by file descriptors, the SLG_ALL_VALID parameter must be enabled. If this parameter is enabled, then the initial file access that is made through the open(2) system call is logged and the object path name is included in this record. This allows the reduce utility to determine the full path name for any succeeding file descriptor operations that occur on this file.

For more information on enabling the SLG_ALL_VALID parameter, see Section 8.8.7.8, page 284.

8.8.7 Auditing on a Cray ML-Safe System Configuration

The security auditing policy for a site running a Cray ML-Safe configuration can be defined by the site. That is, you can run a Cray ML-Safe configuration with or without security auditing enabled. It is recommended that the site policy be determined before the system is up and running and that it be applied consistently at all times. Consistent and proper use of the MLS auditing features helps ensure site security.

Also, there are no settings specified for the the security logging options on a Cray ML-Safe configuration, with the one exception noted in the next paragraph. Cray ships the UNICOS release with default settings for the security logging options. See Section 8.8.5, page 265, for information on the default settings.

To fulfill TCSEC auditing requirements, you must be able to identify an object by its full path name. A Cray ML-Safe configuration can meet this requirement only if the path-tracking configuration option (SLG_PATH_TRACK) is enabled. This option is on by default.

8.8.7.1 Security Log Record Header Definition

Each security log record has a header (`slg_hdr`), followed by record-specific information. The kernel provides the header information for all kernel-generated security log records. A non-kernel Cray ML-Safe process that is performing an activity on behalf of a user formats its own security log record header. However, the kernel also places information in the header fields for the record.

To meet TCSEC requirements regarding the ability to select records by the object label, it was necessary to expand the number of fields in the security log record header.

To maintain compatibility with earlier releases of UNICOS with the MLS feature, the older version of the security log header (`slghdr0`) is retained and a new header structure (`slghdr`) that contains the same information as `slghdr0` is provided, plus the fields necessary to record the security label information (that is, both the security level and compartments) of the subject and the object. The older version of the header contains only the security level of the subject and object.

The `reduce` command uses the `-S` and `-L` options, which allow you to select either the new version (`-S`) or older version (`-L`) of header information in the display produced by the `reduce` command.

The use of the `-S` option is shown in some of the examples in the following sections. If the `-S` or `-L` options are not specified, the older version of the security

log header is displayed by `reduce` (that is, the `-L` display is the default for the UNICOS system).

See Section 8.8.8.2, page 335, for more information on these options.

The security log record header using the older version is as follows:

```
Date_time             Type        o_lvl:n   s_lvl:n   jid:n  pid:n
  r_ids:[ssss(n),ssss(n)]      e_ids:[ssss(n),ssss(n)]       **********
Login uid: ssss(n)
```

The new version of the security log record header is as follows:

```
Date_time   Type   jid:n   pid:n   r_ids:[ssss(n),ssss(n)]
e_ids:[ssss(n),ssss(n)]
    S_Label: level,cmpts  O_Label: level,cmpts
    Login uid: ssss(n)
```

The fields are defined as follows:

Field	Description
`Date_time`	The date and time at which the kernel entered the record in `/dev/slog`.
`Type`	The type-of-record.
`o_lvl:n`	The security level of the object; this is not meaningful if no object is involved. This field is used only on the older version of the record header.
`s_lvl:n`	The security level of the subject. This field is used only on the older version of the record header.
`jid:n`	The job ID.
`pid:n`	The process ID.

`r_ids:[ssss(n),ssss(n)]`

The real user and group names and IDs in the form of [*username(user ID),groupname(group ID)*].

`e_ids:[ssss(n),ssss(n)]`

The effective user and group names and IDs in the form of [*username(user ID),groupname(group ID)*].

`S_Label: level,cmpts`

The security label of the subject. This field is used only on the new version of the record header.

`O_Label: level,cmpts`

The security label of the object. This field is used only on the new version of the record header.

`Login uid: ssss(n)`

The user name and ID at the time of job initiation in the form of *username(user ID*). This is the most reliable identifier in the log entry because it cannot be changed after logging in. This ID is the one for which the password was checked. The login user ID is in the initial `SLG_LOGN` record for any job, regardless of origin. The only exception to this occurs when a user executes the `setsid`(2) or `setjob`(2) system call to create a new session. In this case, the login user ID of the new session is shown in the `SLG_SECSYS` record, that is generated by the `setsid` or `setjob` system call.

The format of the record-specific information following the header can differ, although some records have the same output format, as described in the following sections.

Note: Some of the security audit records described in the following sections use a `Class` field. This field is no longer used and its contents are not meaningful. The class value should be set to 0 by a properly authorized user, as nonzero values are no longer used on UNICOS systems.

8.8.7.2 System Start Record (`SLG_GO`)

The `SLG_GO` record is written as the system is booted into single-user mode. The `SLG_LOG_STARTUP` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log`

`system startup?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e startup` command to enable the generation of this record or the `spaudit -d startup` command to disable the generation of this record.

In addition to the header information, the `SLG_GO` record displays the following information:

```
System Node  Release Version  Machine Mem: Avail_user_mem/Max_mem
```

The fields are defined as follows:

Field	Description
`System`	Value of the `SYS` parameter in `utsname`
`Node`	Value of `NODE` parameter in `utsname`
`Release`	Release level of operating system
`Version`	Version level of operating system
`Machine`	Type of machine
`Mem: Avail_user_mem/Max_mem`	The amount of available user memory and configured memory, respectively

To display a `SLG_GO` record, use the `reduce` command, as shown in the following example.

```
$ /etc/reduce -t go
Feb  5 17:00:00 1992  Startup     o_lvl: 0  s_lvl: 0  jid:0  pid:0
  r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]       **********
Login uid: root(0)

  sn1405 sn1405 8.0.0ah d80.55 Cray Y-MP Mem: 32595968/33554176
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.3 System Shutdown Record (SLG_STOP)

The SLG_STOP record is written when the security log daemon receives a SIG_TERM signal, causing the daemon to exit. This occurs only when an operator issues the shutdown(8) command. The only information recorded in this record is the header information described in Section 8.8.7.1, page 270, for more information.

The SLG_LOG_SHUTDWN parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log system shutdown? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e shutdown command to enable the generation of this record or the spaudit -d shutdown command to disable the generation of this record.

8.8.7.4 System Configuration Change Record (SLG_CCHG)

The SLG_CCHG record is written if the system's configuration is changed by using the udbgen(8), nu(1), or xadmin(8) commands.

The SLG_CF_UNICOS parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log UNICOS configuration changes? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e config command to enable the generation of this record or the spaudit -d config command to disable the generation of this record.

To display a SLG_CCHG record, use the reduce command, as shown in the following example. The fields in this record type are self-explanatory:

```
$ /etc/reduce -t cchg
Jul 15 10:34:19 1993 Configuration o_lvl: 0 s_lvl: 0 jid:1231 pid:33951
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

                  Subtype: UDB change
           login directory: /
```

```
          login root: /
         login shell: /bin/sh
          login name: operator
              action: changed
                 uid: 9
          valid gids: 9
            permbits:
       login failures: 0
        default level: level0
            max level: level0
            min level: level0
    default comparts: none
      valid comparts: none
      authorizations: none
               flags: none
  default categories: none
    valid categories: sysops
            password: same
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.5 System Time Change Record (SLG_TCHG)

The SLG_TCHG record is written when the system's time is set during the system startup sequence. This record is also generated when an administrator changes the system's time with the date(1) command.

The SLG_LOG_TCHG parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log system time change?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e time_change` command to enable the generation of this record or the `spaudit -d time_change` command to disable the generation of this record.

In addition to the header information, the SLG_TCHG record displays the following information:

`Time changed to: time`

This information displays the system's new time.

To display a SLG_TCHG record, use the reduce command, as shown in the following example:

```
$ /etc/reduce -t tchg
Apr  1 09:09:38 1991  Time        o_lvl: 0  s_lvl: 0  jid:2  pid:4
  r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]       **********
Login uid: root(0)

  Time changed to: Mon Apr  1 09:09:22 1991
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.6 Discretionary Access Violation Record (SLG_DISC_7)

The discretionary access violation record is written for all discretionary access failures. The SLG_DISC_7 record is used.

The SLG_DISCV parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log discretionary access violations? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e discv command to enable the generation of this record or the spaudit -d discv command to disable the generation of this record.

A discretionary access violation record is generated for any discretionary access violation that occurs when a process attempts any operation that involves evaluation of access to a file system object or its path.

Specifically, discretionary access violation records can be generated for the following system calls: access(2), acct(2), chacid(2), chdir(2), chmod(2), chown(2), chroot(2), creat(2), dacct(2), exec(2), getacl(2), jacct(2), join(2), lchown(2), link(2), lsecstat(2), lstat(2), mkdir(2), mknod(2), mount(2), msgsys(2), open(2), pathconf(2), readlink(2), rename(2), rmdir(2), rmfacl(2), secstat(2), semsys(2), setacl(2), setfcmp(2), setflvl(2), setpal(2), shmsys(2), stat(2), statfs(2), symlink(2), unlink(2), and utime(2).

In addition to the header information, the SLG_DISC_7 record displays the following information:

```
Function: func (nn)   Violation: text (nnn)
           System call: syscall(number)
Subject: Compartments : sub_comps
          Permissions : sub_permits
                Class : sub_intcls
           Categories : sub_intcat
          Access Mode : access_mode
Object: Level:lvl  uid: user(ID)  gid: group(ID)  device: maj,min
inode: INUM
              Pathname : file
          Compartments : obj_comps
                Class : obj_intcls
            Categories : obj_intcat
                  Mode : obj_mode
```

The fields are defined as follows:

Field	Description
`Function: func (nn)`	The function name and its system call number.
`Violation: text (nnn)`	The error message associated with this violation followed by the error number. If no violation occurred, the value `NONE(0)` is shown. See Section 8.8.10, page 343, for more information.
`System call: syscall(number)`	The system call and its number. The system call is printed after the function. This field is included, as the function that causes the discretionary access violation can be different than the system call.
`Subject: sub_comps`	The subject's active security compartments.
`Permissions: sub_permits`	The subject's authorized permissions (as defined in the UDB).
`Class: sub_intcls`	The subject's active class. This value is no longer used.

`Categories: sub_intcat`

The subject's active category.

`Access Mode: access_mode`

The subject's requested access mode for the desired object. This field is available only on 7.0 and later UNICOS systems with the MLS feature.

`Object: Level: lvl`

The object's security level.

`uid: user(ID)`

Owner of the object and the owner's user ID number.

`gid: group(ID)`

Owning group of the object and the owning group's group ID number.

`device: maj,min`

Major and minor device numbers of the device the object resides on.

`inode: INUM`

The inode number of the object.

`Pathname: file`

The full pathname to the object (displayed if the `-p` option of the `reduce` command is used and the path tracking option is enabled).

`Compartments: obj_comps`

The object's security compartments.

`Class: obj_intcls`

The object's integrity class. This value is no longer used.

`Categories: obj_intcat`

The object's category.

`Mode: obj_mode`

The object's UNICOS mode permission bits.

To display a discretionary access violation record, use the `reduce` command, as shown in the following example.

```
$ /etc/reduce -t disc
Apr  1 09:50:00 1991  Discretionary o_lvl: 0 s_lvl: 0 jid:167  pid:7026
  r_ids:[operator(9),operator(9)] e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

  Function: open (5)     Violation: Permission denied (13)
            System call : access (33)
  Subject: Compartments : none
            Permissions : suidgid
                  Class : 0
             Categories : none
            Access Mode : write
  Object: Level: 0  uid: root(0) gid:root(0) device:0, 235 inode:3412
           Compartments : none
                  Class : 0
              Categories : none
                    Mode : 100755
```

The following example shows the discretionary access violation record that is produced when the `-S` option is specified. When this option is used, the `Subject: Compartments`, `Object: Level`, and `Compartments` information is not printed in the body of the record, but in the record header. See Section 8.8.8.2, page 335, for more information on this option:

```
$ /etc/reduce -t disc -S
Oct 21 13:24:22 1993  Discretionary   jid:945  pid:79982
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 0,none
Login uid: operator(9)

  Function: open (5)     Violation: Permission denied (13)
            System call : access (33)
  Subject: Permissions : none
                  Class : 0
             Categories : none
            Access Mode : write
          Object: 0  uid: root(0) gid:root(0) device:0, 235 inode:3412
```

```
          Compartments : none
                 Class : 0
            Categories : none
                  Mode : 40755
```

To show the full path name to the object, use the `-p` option, as shown in the following example. The `SLG_PATH_TRACK` parameter must be enabled to generate this information.

```
$ /etc/reduce -p -t disc

Apr  1 09:50:00 1991  Discretionary o_lvl: 0 s_lvl: 0 jid:167 pid:7026
  r_ids:[root(0),root(0)] e_ids:[root(0),root(0)]           **********
Login uid: root(0)

  Function: open (5)     Violation: Permission denied (13)
            System call : open (5)
  Subject: Compartments : none
            Permissions : suidgid
                  Class : 0
             Categories : none
            Access Mode : write
  Object: Level: 0  uid: root(0) gid:root(0) device:0, 235 inode:3412
              Pathname : /etc/utmp
          Compartments : none
                 Class : 0
            Categories : none
                  Mode : 100755
```

For the `shmsys`, `semsys`, and `msgsys` system calls, the `SLG_DISC_7` record format displays the header information and the following information:

```
            System call: syscall(number)  Violation: text (nnn)
Subject: Categories   : sub_intcat
Object:      Owner uid: user(ID)
             Owner gid: group(ID)
            Creator uid: creator(ID)
            Creator gid: group(ID)
            Access mode: mode
  Slot sequence number: seq_number
                   Key: key_number
```

The majority of these fields are self-explanatory, except for the following fields, which are defined as follows:

Field	Description
`Slot sequence number`	Indicates the sequence number of the slot for the IPC object. A slot is the entry in a kernel table for the IPC object.
`Key`	Indicates the user-selected, 64-bit identifier for the IPC object. For a private IPC object, the key is 0; in other cases, it is a hex number used in the `reduce` output.

The following example shows the record displayed for a `semsys` system call:

```
$ /etc/reduce -t disc

Jul 27 17:34:19 1994  Discretionary o_lvl:0 s_lvl:0 jid:34  pid:1804
r_ids:[vsx0(36056),vsxg0(31000)] e_ids:[root(0),vsxg0(31000)] ******
   Login uid: vsx0(36056)

       System call : semsys (53)   Violation: Permission denied (13)
  Subject: Categories : none
  Object:    Owner uid : 36056 (vsx0)
             Owner gid : 31000 (vsxg0)
           Creator uid : 36056 (vsx0)
           Creator gid : 31000 (vsxg0)
           Access Mode : read write
  Slot sequence number : 1
                   Key : 0
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.7 Discretionary Access Change Record (SLG_DAC_CHNG)

The discretionary access change record is written for all successful and failed attempts to change the discretionary access control (DAC) attributes of a file. It also records changes to the access control list (ACL) and owner of a file. The `SLG_DAC_CHNG` record type is used to record this information.

On UNICOS 7.0 and earlier systems, this record type was called the `SLG_SUID` record, because only changes to setuid file attributes were being audited

(although this record type was used to indicate changes to the file permissions mode bits also).

DAC changes are logged for the following system calls: `getfacl`(2), `setfacl`(2), `rmfacl`(2), `chmod`(2), `fchmod`(2), `chown`(2), and `fchown`(2).

The `SLG_LOG_DAC` parameter must be on to log this record type To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log discretionary access changes?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e dac` command to enable the generation of this record or the `spaudit -d dac` command to disable the generation of this record.

In addition to the header information, the `SLG_DAC_CHNG` record with path tracking enabled contains the following information:

```
Function: func (nnn)     Violation: text (nnn)
          System call : syscall (nnn)
Subject: Compartments : sub_cmpts
          Permissions : sub_permits
                Class : sub_intcls
           Categories : sub_intcat
Object: Level: lvl  uid: user(ID)  gid: user(id)  device: maj, min
inode: INUM
             Pathname : /pathname
         Compartments : obj_comps
                Class : obj_intcls
           Categories : obj_intcat
                 Mode : obj_mode
```

The field definitions are the same as for for the discretionary access violation record. See Section 8.8.7.6, page 276, for a description of the format.

To display a DAC change record for a `fchmod`(2) system call with object path tracking enabled, use the `reduce` command, as shown in the following example. The `-S` option is used in this example, which means that the `Subject: Compartments` and `Compartments : obj_comps` fields are not printed in the body of the record, but in the record header. See Section 8.8.8.2, page 335, for more information on this option:

```
$ /etc/reduce -t dac -S -p
Oct 21 13:24:22 1993  DAC Change    jid:945  pid:79982
```

```
    r_ids:[root(0),root(0)]  e_ids:[root(0),root(0)]
        S_Label: 0,none  O_Label: 0,none
        Login uid: root(0)

       Function: chown (16)     Violation: NONE (0)
                 System call : chown (16)
        Subject: Permissions : none
                       Class : 0
                  Categories : none
      Object: Level: 51  uid: root(0) gid: root(0)
                                device: 0, 5 inode: 555
                    Pathname : /etc/mnttab
                       Class : 0
                  Categories : none
                        Mode : 40755
```

The following example shows the DAC change record for attaching an ACL to a file:

```
$ /etc/reduce -t dac -p
Jul 15 10:12:29 1993  DAC Change  o_lvl: 0 s_lvl: 0 jid:1214 pid:33078
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

   Function: setacl (87)     Violation: NONE (0)
             System call : setacl (87)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
   Object: Level: 0  uid: operator(9)  gid: operator(9)
                           device: 0, 8  inode: 5008
                Pathname : cmd/myfile
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100600
```

The following is an example of a DAC change record for a chown(2) system call:

```
$ /etc/reduce -t dac -p
Jul 15 10:59:13 1993  DAC Change   o_lvl: 0 s_lvl: 0 jid:1274 pid:34497
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

   Function: chown (16)     Violation: NONE (0)
             System call : chown (16)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
   Object: Level: 0  uid: root(0) gid: root(0)
                           device: 0, 22 inode: 1784
                Pathname : /tmp/maAAAa34497
             Compartments : none
                   Class :   0
              Categories : none
                    Mode : 100000
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.8 Mandatory Access Record (SLG_MAND_7)

The mandatory access record is written for successful file system object accesses, for mandatory access violations that occur during access to a file system object or its path, and for mandatory access violations with process-to-process accesses. The SLG_MAND_7 record is used.

A SLG_MAND_7 record is written for the following system calls when a successful file system object access attempt is made or when an unsuccessful mandatory access violation occurs during access to a file system object or its path: access(2), acct(2), chacid(2), chdir(2), chmod(2), chown(2), chroot(2), creat(2), dacct(2), exec(2), fchmod(2), fchown(2), fcntl(2), fgetpal(2), fstat(2), getacl(2), getdents(2), jacct(2), join(2), lchown(2), lsecstat(2), lstat(2), mkdir(2), mknod(2), open(2), pathconf(2), quotactl(2), readlink(2), rename(2), restart(2), rmfacl(2), secstat(2), select(2), setacl(2), setdevs(2), setfcmp(2), setflvl(2), setpal(2), stat(2), statfs(2), symlink(2), and utime(2).

A SLG_MAND_7 record is automatically written for setflvl(2), setfcmp(2), and setfflg(2) system calls, regardless of success or failure.

The SLG_MAND_7 record for mandatory access failures for process-to-process accesses is written by the following system calls: acctid(2), chkpnt(2), cpselect(2), getlim(2), kill(2), killm(2), limit(2), nicem(2), ptyrecon(2), resume(2), setlim(2), and suspend(2).

The SLG_MAND_7 record for mandatory access failures is automatically written for the following system calls for mandatory access failures on accesses and for mandatory access failures or successes when removing one of the IPC objects: msgctl(2), msgget(2), msgrcv(2), msgsnd(2), semctl(2), semget(2), semop(2), shmat(2),shmctl(2), shmdt(2), orshmget(2). For removal failures to be audited, the SLG_ALL_RM configuration parameter must be enabled. For the mandatory access control failures on accesses to be audited, the SLG_MANDV configuration parameter must be enabled. See Section 8.8.7.6, page 276, for a description of the format of these records.

The SLG_MANDV parameter must be on for mandatory access violations to be logged.

To set this parameter, use the Configure system ->Multilevel security (MLS) configuration->Security log file configuration->Log mandatory access violations? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e mandv command to enable the generation of this record. The spaudit -d mandv command disables the generation of this record.

The SLG_ALL_VALID parameter must be on for all successful file system object accesses to be logged. All of the object fields in the record may be 0, depending on what type of error is encountered. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all access attempts? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e all_valid command to enable the generation of this record for successful accesses. The spaudit -d all_valid command disables the generation of this record.

The format of this record is the same as described previously for the discretionary access violation record. See Section 8.8.7.6, page 276, for a description of the format.

To display a SLG_MAND_7 record, use the reduce command, as shown in the following example. The example shows the actual system call causing the violation is shown. This line is printed when the system call is different than the function. In this example, the create(2) system call called the open(2) function,

causing the violation. The (8) after the creat in the following example is the actual system call number:

```
$ /etc/reduce -p -t mand

Apr  1 09:52:42 1991  Mandatory   o_lvl: 0  s_lvl: 0  jid:17  pid:7476
  r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]      **********
Login uid: operator(9)

  Function: open (5)     Violation: Security category violation (319)
             System call : creat (8)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : secadm sysadm
             Access Mode : write
  Object: Level: 0   uid: root(0) gid:sys(3) device:0, 225 inode:926
                Pathname : /etc/udb
             Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100600
```

The following example shows the record produced for a process-to-process access violation. This example shows the use of the -S option. See Section 8.8.8.2, page 335, for more information on this option:

```
$ /etc/reduce -t mand -S

Nov 19 09:29:51 1993  Mandatory  jid:38  pid:1843
r_ids:[root(0),root(0)]      e_ids:[root(0),root(0)]
    S_Label: 4,comp24  O_Label: 7,none
    Login uid: root(0)

             System call : limit (115)   Violation: No such process (3)
    Subject: Categories : none
    Object:       Job ID : 50
              Process ID : 1810
             Process uid : 0 (root)
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.9 Login Validation Record (SLG_LOGN)

The SLG_LOGN record is written for all successful and unsuccessful login attempts. This includes all login attempts that are done interactively through NQS, ftp(1b), rlogin(1), dgdemon(8), and rshd(8). The cron(8) command also issues this record whenever a crontab(1) or at(1) job is initiated on behalf of a user. For sublogins, the audit record that is written is for the initial successful login only. This matches the user ID and job ID in the job termination record.

Except for sublogins, the login user ID and job ID that are written in this record remain unchanged throughout the life of a job on the system and can be used to track all activities of this job from initiation to completion. For sublogins, the audit record contains the initial user ID or the second login user ID. For sublogins, after the job initiation record is written, audit records that are generated during execution of the job may contain the second login user ID.

The entries are logged through the slgentry(2) system call.

If a login failure is because the user's name is not valid (that is, not in the UDB), the failure is logged as an invalid user ID. The text string for the invalid user name is not logged. This is done to ensure that if a user inadvertently types his or her password in the ID prompt, the clear text password is not written in the security log. The entries are logged through slgentry.

The SLG_JSTART parameter must be on to enable the generation of this record. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log start of job? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e jstart command to enable the generation of this record. The spaudit -d jstart command disables the generation of this record.

The SLG_LOGN record type uses the SLG_LOG_USER record subtype to record clear text passwords for failed logins. If this record is enabled and the user is trapped (by having the usrtrap permission assigned in the user database (UDB) entry), and the user's login attempt fails, the password is recorded as part of the record. This means that passwords are sometimes recorded in the security log.

The SLG_USER parameter must be on to record the clear text password violations (assuming the SLG_JSTART parameter is also enabled). Again, the entered password is recorded only for a failed login attempt when the usrtrap permission is set for the user. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log name and password on login

`password fail?` selection in the UNICOS Installation and Configuration Menu System.

The `spaudit -e user` command enables the user password to be recorded for failed password attempts by a trapped user. The `spaudit -d user` disables this ability.

In addition to the header information, the `SLG_LOGN` record displays the following information:

```
Login: [ssss(n),ssss(n)] : Result  Host  tty line  Failures
```

The fields are defined as follows:

Field	Description
`Login: [ssss(n),ssss(n)]`	The user, group names, and IDs in the form of [*username(user ID),groupname(group ID)*].
`Result`	Indicates whether the login was successful (signified by `Okay`) or displays a violation message if the login was unsuccessful. The following list explains the login violation messages:

Message	Description
`Lastlog`	`login` was unable to update the user's `Last Login Time` entry in the UDB.
`Password`	The user entered an invalid password.
`Setusrv`	User's defined security label is invalid, outside the range of the system's label, or outside the range defined in the NAL.
`Shell exec`	User was not found in the `/etc/utmp` file.
`Locked`	Account is disabled in the UDB.
`Disabled`	Account is disabled because the `logfails` field in the UDB exceeds the limits defined by `MAXLOGS`.
`Dialup`	An invalid dialup password was used.

`Trusted subject`	User has authorized `system` or `daemon` category.
`Root console`	The user attempted to log in as `root` from a console other than the system console.
`Setjob failed`	Unable to set up login process with a new job ID and table entry.
`Attempted to login to localhost`	A login attempt was not allowed for `localhost`.

There are four different message formats:

Format	Description
Message `error`	*Message* describes the reason for the failure (as defined in the previous list).
Message `error (disabled)`	*Message* describes the reason for the failure, while `(disabled)` means that account is disabled because `MAXLOGS` was equaled or exceeded. See Section 8.5.5.11, page 220, for more information on how `MAXLOGS` works. The errors most likely to use this message format are the `Password` and `Disabled` messages. Until an account is disabled, only the message portion appears. For example, if the password is incorrectly entered, the `Password error` message is recorded. The first time `MAXLOGS` is equaled, in addition to the *Message*, the `(disabled)` portion of the message also appears. For example, if an incorrect password is entered until `MAXLOGS` is equaled, the

		`Password error (disabled)` message is recorded in the log. Any subsequent attempts to log into an account after the first time `MAXLOGS` is equaled is logged with the `Disabled error (disabled)` message. The account stays disabled until it is cleared.
	Message `error (disabled)`	*Message* describes the reason for the failure, and `(disabled)` means the account is disabled because `MAXLOGS` was exceeded.
`Host`		Name of remote host
`tty line`		The `tty` line used for the login attempt or the source of the login attempt (for example, `cron` or `rshd`).
`Failures`		Number of login failures to date

To display a `SLG_LOGN` record, use the `reduce` command, as shown in the following examples. The first example shows a record produced when a user logs in correctly and has no previous login errors:

```
$ /etc/reduce -t logn

Apr 29 08:12:57 1991  Validation      o_lvl: 0  s_lvl: 0  jid:0 pid:994
  r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    **********

 Login to [operator(9),operator(9)] : Okay via ows131 on /dev/console
```

The following example shows the record produced when a user logs in correctly, but has a previous login failure:

```
$ /etc/reduce -t logn

Apr 29 08:14:21 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:1067
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
*********
Login uid: operator(9)
```

```
 Login to [operator(9),operator(9)] : Okay   via juniper04
 on /dev/ttyp001 -- 1 previous failures
```

The following example shows the record produced when a user entered an incorrect password:

```
$ /etc/reduce -t logn

Apr 29 08:14:15 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:1067
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
*********
Login uid: operator(9)

  Login to [operator(9),operator(9)] : Password error  via juniper04
on /dev/ttyp001 -- 0 previous failures
```

The following example shows the record produced when a user has exceeded the maximum number of login attempts (defined by MAXLOGS):

```
$ /etc/reduce -t logn

Apr 29 09:07:55 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:2456
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
Login uid: operator(9)

  Login to [operator(9),operator(9)] : Disabled error (disabled)
via fir28 on /dev/ttyp066 -- 5 previous failures
```

The following example shows the record produced when an administrator has set the disabled field in the UDB for a user. This record is produced when the disabled user attempts to log in:

```
$ /etc/reduce -t logn

Apr 29 09:10:28 1991  Validation     o_lvl: 0  s_lvl: 0  jid:0 pid:2907
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
Login uid: operator(9)
```

```
  Login to [operator(9),operator(9)] : Locked error  via fir28
on /dev/ttyp066 -- 0 previous failures
```

The following example shows the record produced for a trapped user with a password failure and SLG_USER is enabled:

```
$ /etc/reduce -t logn
Jul 15 10:45:47 1993  Validation   o_lvl: 0 s_lvl: 0 jid:0 pid:34302
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Password error  via cherry21
on /dev/ttyp012 - - 0 previous failures
name: operator, password: badpass
```

The following example shows the record produced for a user who logged in through the remote shell:

```
$ /etc/reduce -t logn

Oct 18 10:55:43 1993  Validation o_lvl:0  s_lvl:0  jid:3389  pid:65592
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**** ******
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Okay   via pecan10 on   rshd
```

The following example shows the record produced for a user who initiated a job through cron:

```
$ /etc/reduce -t logn

Oct 18 11:00:01 1993  Validation o_lvl:0  s_lvl:0  jid:3393  pid:66918
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Okay via cron on
```

The following example shows the user to produce the record header with the subject and object label information. See Section 8.8.8.2, page 335, for more information on this option:

```
$ /etc/reduce -t logn -S

Oct 21 13:24:17 1993  Validation  jid:1094  pid:79926
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)
    S_Label: 0,none  O_Label: 0,none
    Login uid: operator(9)

  Login to [operator(9),operator(9)] : Okay via cherry21 on /dev/ttyp043
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.10 Tape Activity Record (SLG_TAPE)

All records generated by `tpdaemon`(8) are logged using the `SLG_TAPE` record. The entries are logged through `slgentry`(2).

The following two additional events generate audit records by `tpdaemon`:

- Mandatory access control (MAC) failures on requests to reserve a device.
- Any tape MAC mediation performed by `tpdaemon`; this event is never recorded on a Cray ML-Safe system configuration, as all tape mediation is performed by Cray/REELlibrarian.

The `SLG_TAPE` record is generated by the following tape commands: `rls`(1), `rsv`(1), `tpapm`(8), `tpcatalog`(1), `tpdev`(8), `tpdstop`(8), `tpfrls`(8), `tpgstat`(8), `tpmls`(8), `tpmnt`(1), `tprst`(1), `tpscr`(8), `tpset`(8), `tpstat`(1), and `tpu`(8).

The `SLG_TRUST` record is generated by the `tpconfig`(8) command. See Section 8.8.7.24, page 324, for more information on this record type.

The `SLG_LOG_TAPE` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log tape activity?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e tapes` command to enable the generation of this record or the `spaudit -d tapes` command to disable the generation of this record.

The `message file` field in the `SLG_TAPE` record is used only by the `rsv`(1) command. The `external VSN` field is used only on commands that take a VSN as a parameter. The `violation` field is the `tpdaemon` error; otherwise, for commands, it is the return code from `tpdaemon`.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t tape
Nov  9 16:45:17 1993  Tape I/O  o_lvl: 0 s_lvl: 0 jid:1158  pid:33559
r_ids:[operator(9),operator(9)]  e_ids:[root(0),root(0)]  **********
    Login uid: operator(9)

            NQS batch job ID:
                  Operation: Tape Mount
                  violation: 0
         permitted privileges: PRIV_NULL
               tape file name: AAAA33531
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/1158tpmn33559
                 message file:
                 external VSN: 001815
```

The following example shows the record generated by the `tpstat` command:

```
$ /etc/reduce -t tape
Dec  3 11:12:43 1993  Tape I/O   o_lvl: 0  s_lvl: 0  jid:514  pid:32969
r_ids:[operator(9),operator(9)]   e_ids:[root(0),root(0)]   **********
    Login uid: operator(9)

            NQS batch job ID:
                  Operation: Tape Status Request
                  violation: 0
         permitted privileges: PRIV_NULL
               tape file name:
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/514tpst32969
                 message file:
```

The following example shows the record generated by the tape daemon for a `rsv` command:

```
$ /etc/reduce -t tape
Dec  3 11:13:26 1993  Tape I/O   o_lvl: 0  s_lvl:54  jid:9  pid:32987
r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]          **********
    Login uid: root(0)

            NQS batch job ID:
                   Operation: User Reserve Status Request
                   violation: 0
         permitted privileges: PRIV_NULL
               tape file name:
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/9gsta32987
                 message file:
```

The following example shows the record generated by the `rsv` command:

```
$ /etc/reduce -t tape
Dec  3 11:13:30 1993  Tape I/O  o_lvl: 0  s_lvl: 0  jid:514  pid:32990
r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
          **********
    Login uid: operator(9)

            NQS batch job ID:
                   Operation: Reservation Request
                   violation: 0
         permitted privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                               PRIV_MAC_WRITE
               tape file name:
                 request pipe: /tmp.mld/jtmp/jtmp.000514a/TAPE_REQ_514
                   reply pipe: /usr/spool/tape/514rsv32990
                 message file: /drizzle/operator/tape.msg
```

The following example shows the record generated by the `tpmnt` command:

```
$ /etc/reduce -t tape
```

```
Dec  3 11:14:01 1993  Tape I/O o_lvl: 0  s_lvl: 0  jid:514  pid:32998
r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
          **********
    Login uid: operator(9)

            NQS batch job ID:
                   Operation: Tape Mount
                   violation: 0
         permitted privileges: PRIV_NULL
               tape file name: PSU
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/514tpmn32998
                 message file:
                 external VSN: 002895
```

The following example shows the record generated by the `rls` command:

```
$ /etc/reduce -t tape
Dec  3 11:14:06 1993  Tape I/O  o _lvl: 0  s_lvl: 0  jid:514  pid:33000
r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
          **********
   Login uid: operator(9)

            NQS batch job ID:
                   Operation: Free Device Group
                   violation: 0
         permitted privileges: PRIV_NULL
               tape file name:
                 request pipe: /usr/spool/tape/daemon.request
                   reply pipe: /usr/spool/tape/514rls33000
                 message file:
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.11 End-of-job Record (SLG_EOJ)

The SLG_EOJ record is written when a user's session exits the system, regardless of how the session originates (that is, whether the session is initiated through batch or interactive means, cron(8), rlogin(1), rshd(8), or through the setsid(2) system call). This record can be correlated with a SLG_LOGN record through the login user ID field and the job ID field. Except for sublogins, these

fields are the same in both the SLG_LOGN and SLG_EOJ records. For sublogins, the login user ID in a SLG_EOJ record is the initial login user ID; in this case, the job ID should be used to track a sublogin SLG_EOJ record.

The SLG_JEND parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log end of job? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e jend command to enable the generation of this record or the spaudit -d jend command to disable the generation of this record.

Only the header information is produced for this record, as shown in the following example:

```
$ /etc/reduce -t eoj

Apr  1 09:56:46 1991  End of Job   jid:188  uid:operator(9) **********
    Login uid: operator(9)



Apr  1 09:57:05 1991  End of Job   jid:58  uid:operator(9) **********
    Login uid: operator(9)



Apr  1 09:57:05 1991  End of Job   jid:59  uid:root(0) **********
    Login uid: root(0)
```

See Section 8.8.7.1, page 270, for a description of the header information.

8.8.7.12 Change Directory Record (SLG_CHDIR)

The SLG_CHDIR record is produced each time a chdir(2) system call is made (that is, whenever users change their directories).

The SLG_LOG_CHDIR parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log chdir requests? selection in the UNICOS Installation and Configuration Menu System.

In addition, the SLG_PATH_TRACK parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Track all path names on accesses?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e chdir,object_path` command to enable the generation of this record. The `spaudit -d chdir,object_path` command disables the generation of this record.

In addition to the header information, the SLG_CHDIR record displays the following information:

```
Relative directory path: rel_path
Absolute directory path: abs_path
```

The fields are defined as follows:

Field	Description
`rel_path`	The relative path name to the object
`abs_path`	The absolute path name to the object (usually starts with `/`). It is possible to get a path name that does not start with `/`. For example, if you are using `reduce` to examine a new security log file and your original login (and the `cd $HOME`) record was in an archived log file, then `reduce` would be unable to determine where you started. Instead, it creates as much of the path name as possible.

To display a SLG_CHDIR record, use the `reduce` command, as shown in the following example, which shows the `-S` option used to produce the record header with the subject and object label information: See Section 8.8.8.2, page 335, for more information on this option.

```
$ /etc/reduce -S -t chdir
Oct 21 13:24:18 1993  Change Directory jid:1094 pid:79963
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 0,none
    Login uid: operator(9)

    Relative directory path: ./libc
    Absolute directory path: /ulib/usr/src/lib/libc
```

```
$ /etc/reduce -t chdir -s 04010930
Apr  1 09:59:10 1991  Change Directory o_lvl:0 s_lvl:0 jid:192 pid:9232
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
Login uid: root(0)

    Relative directory path: /usr/spool/nqs/private/root
    Absolute directory path: /usr/spool/nqs/private/root
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.13 Security-related System Call Record (SLG_SECSYS)

The SLG_SECSYS record is written whenever one of the following security-related system calls is executed: fsetpal(2), setjob(2), setpal(2), setsid(2), setucat(2), setucmp(2), setulvl(2), and setusrv(2).

It is also written if an authorization error or system error occurs when a process issues the slgentry(2) system call. The slgentry system call is used by the Cray ML-Safe process to write the record to the security log.

The SLG_LOG_SECSYS parameter must be on for this record to be generated. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log security system calls? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e secsys command to enable the generation of this record. The spaudit -d secsys command disables the generation of this record.

There are three formats for this record. The first format is used when the setucat(2) and setusrv(2) system calls are executed. This format is also used for an authorization error or system error occurs when a process issues the slgentry(2) system call. For the first format, in addition to the header information, the SLG_SECSYS record displays the information shown in the following example.

```
   Function: func (nn)         Violation: text (nnn)
  Subject: Active Compartments : sub_active_comps
            Valid Compartments : sub_val_comps
                   Permissions : sub_permits
                         Class : sub_intcls
             Active Categories : sub_act_cat
              Valid Categories : sub_val_cat
```

The fields are defined as follows:

Field	Description
`Function: func (nn)`	The function name and its system call.
`Violation: text (nnn)`	The error message associated with this violation followed by the error number. If no violation occurred, the value `NONE(0)` is shown. See Section 8.8.10, page 343, for more information.
`sub_act_comps`	The subject's active security compartments.
`sub_val_comps`	The subject's authorized security compartments.
`sub_permits`	The subject's authorized permissions (as defined in the UDB).
`sub_intcls`	The subject's active integrity class. This value is no longer used.
`sub_act_cat`	The subject's active category.
`sub_val_cat`	The subject's authorized categories.

To display the first format of a `SLG_SECSYS` record, use the `reduce` command, as shown in the following example:

```
$ /etc/reduce -t secsys

Apr  1 10:02:35 1991 Security Syscall o_lvl:0 s_lvl:0 jid:212 pid:11855
 r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]        **********
Login uid: root(0)

   Function: setusrv (89)     Violation: NONE (0)
  Subject: Active Compartments : none
           Valid Compartments : none
                  Permissions : suidgid
```

```
                    Class : 0
             Active Categories : none
              Valid Categories : secadm


Apr  1 10:02:37 1991 Security Syscall o_lvl:0 s_lvl:0 jid:212 pid:11855
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]        **********
Login uid: root(0)

   Function: setucat (154)     Violation: NONE (0)
  Subject: Active Compartments : none
           Valid Compartments : none
                  Permissions : suidgid
                        Class : 0
            Active Categories : none
             Valid Categories : system



Apr  1 10:02:39 1991 Security Syscall o_lvl:0 s_lvl:0 jid:212 pid:11855
  r_ids:[root(0),operator(9)]   e_ids:[root(0),operator(9)]  ********
Login uid: operator(9)

   Function: setusrv (89)     Violation: NONE (0)
  Subject: Active Compartments : none
           Valid Compartments : none
                  Permissions : suidgid
                        Class : 0
            Active Categories : none
             Valid Categories : secadm
```

The second format of the SLG_SECSYS record is used by the setsid(2) and setjob(2) system calls to audit the creation of a new job or session ID from within an existing session. The purpose of this record is to document the changing of the job or session ID so that all actions taken by the new job or session can be traced to the original session and user ID.

In addition to the header information, this form of the SLG_SECSYS record displays the following information:

```
   Function: func: (nn)     Violation: text (nnn)
  Subject: Active Compartments : sub_active_comps
            Valid Compartments : sub_val_comps
                   Permissions : sub_permits
                         Class : sub_TFM_class
             Active Categories : sub_act_cat
              Valid Categories : sub_val_cat
                    New job id : new_job_id
                   New job uid : new_uid (nnnn)
```

To display the second format of a SLG_SECSYS record, use the reduce command, as shown in the following example:

```
$ /etc/reduce -t secsys

Oct 18 14:47:33 1993 Security Syscall o_lvl:0 s_lvl:0 jid:4455 pid:65926
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

   Function: setsid (117)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm sysadm
                    New job id : 4456
                   New job uid : dnn (1346)
```

A third form of the SLG_SECSYS record is written when the setpal(2) system call is executed. The following example shows this type of record. The subject and object security labels are supplied in the following format: level, compart1 compart2,.... The first record is generated by a setprivs(8) command and no privilege assignment lists (PALs) entries were used. The second record was generated by a setpal(1) command and three PAL entries were set.

This record type can contain up to eight PAL entires, which is enough for all software released by Cray. If a record generated by setpal(2) contains more

than eight PAL entries, the additional records are written until all PAL entries have been logged:

```
$ /etc/reduce -t secsys

Jul 15 11:15:12 1993 Security Syscall o_lvl:0 s_lvl:0 jid:1274 pid:34930
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
  Login uid: operator(9)

     Function: setpal (220)     Violation: NONE (0)
     Subject: active categories: secadm
                         Label: 0, none
     Object:                 Label: 0, none
                           uid: operator(9)  gid: operator(9)
                        device: 0, 8  inode: 5244
                      Pathname: mybin
              allowed privileges: PRIV_NULL
               forced privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                  PRIV_MAC_WRITE
       set effective privileges: PRIV_DAC_OVERRIDE,
                                   PRIV_MAC_READ,PRIV_MAC_WRITE
               total PAL entries: 0
              PAL version number: 0
                     PAL entries: 0

Jul 15 11:15:38 1993 Security Syscall o_lvl:0 s_lvl:0 jid:1274 pid:34933
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
  Login uid: operator(9)

    Function: setpal (220)     Violation: NONE (0)
    Subject: active categories: secadm
                          Label: 0, none
    Object:                 Label: 0, none
                            uid: operator(9)  gid: operator(9)
                         device: 0, 8  inode: 5244
                       Pathname: mybin
             allowed privileges: PRIV_NULL
              forced privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                                 PRIV_MAC_WRITE
      set effective privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
```

```
                    PRIV_MAC_WRITE
  total PAL entries: 3
 PAL version number: 0
        PAL entries: 3

      privilege set: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                     PRIV_MAC_WRITE
     privilege text: exec
privileged category: secadm

      privilege set: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                     PRIV_MAC_WRITE
     privilege text: exec
privileged category: sysadm

      privilege set: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                     PRIV_MAC_WRITE
     privilege text: exec
privileged category: system
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.14 NAMI Function Record (SLG_NAMI)

This record is produced whenever a file's path is searched by any of the following system calls: link(2), mkdir(2), rmdir(2), and unlink(2). If logging of this record is enabled, it is written in addition to any other type of discretionary or mandatory access records that are written.

The format is the same as described for the discretionary access violation record. See Section 8.8.7.6, page 276, for a description of the format.

The SLG_MKDIR, SLG_RMDIRV, SLG_LINKV, SLG_REMOVEV, SLG_ALL_RM, and / or SLG_ALL_NAMI parameters must be on for this record to be generated.

The SLG_MKDIRV parameter allows the logging of all mkdir(2) system call violations.

To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all directory make (mkdir) violations? selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e mkdirv` command to enable the generation of this record or the `spaudit -d mkdirv` command to disable the generation of this record.

The `SLG_RMDIRV` parameter allows the logging of all `rmdir`(2) system call violations.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all directory remove (rmdir) violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e rmdirv` command to enable the generation of this record or the `spaudit -d rmdirv` command to disable the generation of this record.

The `SLG_LINKV` parameter allows the logging of all link violations, which includes mandatory access violation within link. This violation is also logged if the `SLG_MANDV` or `SLG_ALL_NAMI` configuration parameters are enabled.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all link (ln) violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e linkv` command to enable the generation of this record or the `spaudit -d linkv` command to disable the generation of this record.

All of the object fields in the record may be 0, depending on what type of error is encountered and which routine (`link` or `nami`) generated the record.

The `SLG_REMOVEV` parameter allows the logging of all file removal violations for the `unlink`(2) system call.

To set this parameter, use the `Configure system ->Multilevel security (MLS) configuration->Security log file configuration->Log all remove violations?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e removev` command to enable the generation of this record or the `spaudit -d removev` command to disable the generation of this record.

The `SLG_ALL_RM` parameter allows the logging of all remove (`unlink`(2)) requests regardless of the success or failure of the action.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all remove (rm) requests?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e all_rm` command to enable the generation of this record or the `spaudit -d all_rm` command to disable the generation of this record.

The `SLG_ALL_NAMI` parameter allows the logging of all `mkdir`(2), `rmdir`(2), `link`(2), and `unlink`(2) system calls regardless of success or failure of the call.

To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all mkdir, rmdir, link, and rm calls?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e all_nami` command to enable the generation of this record or the `spaudit -d all_nami` command to disable the generation of this record.

All of the object fields in the record may be 0, depending on what type of error is encountered.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nami -s 04011000 -p

Apr  1 10:04:24 1991  File Control   o_lvl:0 s_lvl:0 jid:102 pid:7496
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
Login uid: operator(9)

   Function: link (9)     Violation: File exists (17)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
   Object: Level: 0  uid: root(0) gid:root(0) device:0, 192 inode:689
                Pathname : /fstest1/ram1/bo125
             Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100644
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.15 Setuid System Call Record (SLG_SETUID)

This record is produced whenever a `setuid`(2) or `setreuid`(2) system call is executed.

The `SLG_SUID_RQ` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all setuid requests?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e setuid` command to enable the generation of this record or the `spaudit -d setuid` command to disable the generation of this record.

In addition to the header information, the `SLG_SETUID` record displays the following information:

```
Setuid call from real_name (RUID) to eff_name (EUID) was text
```

The fields are defined as follows:

Field	Description
`real_name`	The real user login name
`RUID`	The real user ID
`eff_name`	The effective user login name
`EUID`	The effective user ID
`text`	Indicates if call was successful (`successful` or `NOT successful`)

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t setuid -s 04010800 -p

Apr  1 10:27:50 1991  Setuid Syscall o_lvl:0  s_lvl:0 jid:310 pid:21296
  r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

   Setuid call from operator (9) to root (0) was NOT successful
```

```
Apr  1 10:27:58 1991  Setuid Syscall o_lvl:0  s_lvl:0 jid:312 pid:21343
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

   Setuid call from operator (9) to operator (9) was successful


Apr  1 10:28:00 1991  Setuid Syscall o_lvl:0  s_lvl:0 jid:313 pid:21347
 r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]          **********
Login uid: root(0)

   Setuid call from root (0) to root (0) was successful

Apr  1 10:28:12 1991  Setuid Syscall o_lvl:0  s_lvl:0  jid:85 pid:9464
  r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
Login uid: operator(9)

   Setuid call from root (0) to operator (9) was successful
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.16 Su Attempt Record (SLG_SU)

This record is produced for all successful or unsuccessful attempts to execute the `su` command, essentially producing the same information found in `sulog`.

The `SLG_SULOG` parameter must be on to generate this record. To set parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all su attempts?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e sulog` command to enable the generation of this record or the `spaudit -d sulog` command to disable the generation of this record.

In addition to the header information, the `SLG_SULOG` record displays the following information:

```
Setuid call from real_name (RUID) to eff_name (EUID) was text
```

The fields are defined as follows:

Field	Description
real_name	The real user login name
RUID	The real user ID
eff_name	The effective user login name
EUID	The effective user ID
text	Indicates if call was successful (successful or NOT successful)

To display this record, use the reduce command as shown in the following example:

```
$ /etc/reduce -t sulog

Dec 11 05:10:07 1991  Sulog       o_lvl: 0  s_lvl: 0  jid:11  pid:3072
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
    Login uid: root(0)

   Setuid call from root (0) to adm (4) was successful
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.17 Networks Security Violations Record (SLG_IPNET)

This record is produced for all network errors. The entries are logged through the slgentry(2) system call.

This record is generated only if the SLG_LOG_IPNET configuration parameter is enabled. To set parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log IP layer activity? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e ipnet command to enable the generation of this record or the spaudit -d ipnet command to disable the generation of this record.

In addition to the header information, the SLG_IPNET record displays the following information:

The fields are defined as follows:

```
Function: requested_function     Violation: text (nnn)

   IP Security Option : type
    Network Interface : number
        Network Route : number

Subject:         Host : host
                Level : sub_level
          Compartments : sub_comparts
Object:          Level : obj_level
          Compartments : obj_comparts
```

Field	Description
`Function: requested_function`	Defines one of the following functions: Set IP Security Option (1) or Get IP Security Option (2). A record with a requested function of 1 is generated when there is an error in the `ip_output` () routine. A record with a requested function of 2 is generated when there is an error on the IP input in the transport layer routines. In the latter case, the security header information is not relevant because the error occurred in an interrupt handler.
`Violation: text (nnn)`	Violation description and error number. See Section 8.8.10, page 343. for more information.
`IP Security Option : type`	Defines one of the following IP security option types: None (0), BSO (1), or CIPSO (2)
`Network Interface : number`	Reserved for future use
`Network Route : number`	Reserved for future use
`Subject: Host : host`	Subject's host machine. If the remote host is not in `/etc/hosts`, then the Internet address is displayed. The `Subject` field refers

to the local process or the remote host, depending on whether the datagram is outgoing or incoming, respectively.

`Level : sub_level`

Subject's active security level

`Compartments : sub_comparts`

Subject's active compartments

`Object: Level : obj_level`

Security level of incoming datagram

`Compartments : obj_comparts`

Active compartments of the incoming datagram

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t netip

Dec 11 15:34:24 1991 Network IP Layer o_lvl: 0 s_lvl:16 jid:115 pid:5787
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
    Login uid: operator(9)

  Function: Set IP Security Option (1)   Violation: Host not authorized
  in NAL (330)

      IP Security Option : NONE (0)
       Network Interface : 0
           Network Route : 0

   Subject:         Host : pecan01.cray.com
                   Level : 16
            Compartments : none
   Object:         Level : 0
            Compartments : none
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.18 Cray NFS Request Record (SLG_NFS)

This record is produced for all Cray-to-Cray NFS requests, regardless of success or failure of the request.

The SLG_NFS parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all Cray-to-Cray NFS requests` selection in the UNICOS installation and configuration menu system.

You can also use the `spaudit -e nfs` command to enable the generation of this record or the `spaudit -d nfs` command to disable the generation of this record.

In addition to the header information, the SLG_NFS record displays the following information:

```
   Function: requested_function     Violation: text (nnn)

 Remote Server (Client): address
           Session ID   : sid
       Transaction ID   : xid
        Device Number   : dev
          Inode Number  : inode
```

The fields are defined as follows:

Field	Description
`Function: requested_function`	Cray NFS (CNFS) procedure number, which indicates what CNFS operation is returned
`Violation: text (nnn)`	Violation description and error number. See Section 8.8.10, page 343. for more information.
`Remote Server (Client) : address`	The hostname of the remote CNFS server or client, respectively. If the hostname is not known, the Internet address is shown.
`Session ID : sid`	The session identification number of the client process that made the request

`Transaction ID : xid`

The RPC packet transaction identification generated by the CNFS client

`Device Number : dev`

The major or minor device number in the server's file systems of the file being accessed

`Inode Number : inode`

The inode number in the server's file system of the file being accessed

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nfs

Sept 10 15:17:32 1992 NFS     o_lvl: 0 s_lvl: 0 jid:629 pid:44007
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
  **********
    Login uid: operator(9)

   Function: lookup (4)     Violation: NONE (0)

           Remote Server : uss.cray.com
              Session ID : 629
          Transaction ID : 324f0b0bd17d44
           Device Number : 8767
            Inode Number : 4
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.19 File Transfer Record (SLG_FXFR)

This record is produced for all successful and failed `ftpd`(8) attempts. These entries are logged through the `slgentry`(2) system call.

The SLG_FILEXFR parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all file transfers?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e filexfr` command to enable the generation of this record or the `spaudit -d filexfr` command to disable the generation of this record.

In addition to the header information, the SLG_FXFR record displays the following information:

```
Type_of File Transfer
Remote Host: ip_address  Remote Port: host_port  Local Port: host_port

Function: func (nn)   Violation: text (nnn)
Subject: Compartments : sub_comps
          Permissions : sub_permits
                Class : sub_intcls
           Categories : sub_intcat
          Access Mode : access_mode
Object: Level:lvl  uid: user(ID)  gid: group(ID)  device: maj,min
inode: INUM
             Pathname : file
         Compartments : obj_comps
                Class : obj_intcls
           Categories : obj_intcat
                 Mode : obj_mode
```

The fields are defined as follows:

Field	Description
`Type_of File Transfer`	Defines if the file is to be imported or exported by indicating `Incoming File Transfer` or `Outgoing File Transfer`.
`Remote Host: ip_address`	The host name (or internet address) of the remote host.
`Remote Port: host_port`	Remote host port number that is used for the transfer.
`Local Port: host_port`	The Cray port number that is used for the transfer. If this port is registered in `/etc/services`, then the network service text is displayed in parentheses after the port number.

The rest of this record is formatted as described previously for the discretionary access violation record. See Section 8.8.7.6, page 276, for a description of the format.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -p -t xfer
Dec 10 20:00:31 1992 File Transfer o_lvl:0 s_lvl:0 jid:5058  pid:82087
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
Login uid: operator(9)

Incoming File Transfer
Remote Host: fox          Remote Port: 1973  Local Port: 21 (ftpd)

  Function: open (5)     Violation: Security category violation (319)
  Subject: Compartments  : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
  Object: Level: 0   uid: root(3) gid:sys (3) device:0, 225 inode:926
                Pathname : /sn422/os/jnn/text
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100600
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.20 Network Configuration Change Record (SLG_NETCF)

This record is written when changes are made to the workstation access list (WAL), network access list (NAL), interface, and CIPSO map. These entries are logged through the `slgentry`(2) system call.

The `SLG_CF_NET` parameter must be on for this record to be generated. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log all network configuration changes?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e netcf` command to enable the generation of this record or The `spaudit -d netcf` command to disable the generation of this record.

The `SLG_NETCF` record uses the following five record subtypes:

- `SLG_NET_NAL` (records changes to the network access list (NAL))
- `SLG_NET_WAL` (records changes to the workstation access list (WAL))
- `SLG_NET_INTF` (records changes to the interface)
- `SLG_NET_MAP` (records changes to the CIPSO map)
- `SLG_NET_ROUTE` (records changes to the route)

In addition to the header information, the `SLG_NET_NAL` record displays the following fields, which are defined as follows:

Field	Description
`Function`	Can be `add` or `delete`, depending on whether the NAL entry was added or deleted, respectively.
`Address`	The host or network address for the entry
`Netmask`	The netmask for network entries
`Class`	The class value. This value is no longer used.
`Min level`	The minimum security level
`Min comparts`	The minimum compartment set
`Max level`	The maximum security level
`Access modes`	The access modes
`IP security option`	One of the following security options: Domain of interpretation for CIPSO (`DOI`); the required protection authority on input flags for BSO (`Authority in`); the required protection authority on output for BSO (`Authority out`)

In addition to the header information, the `SLG_NET_WAL` record displays the following fields, which are defined as follows:

Field	Description
`Function`	Can be `add` or `delete`, depending on whether the WAL entry was added or deleted, respectively.
`Address`	The host or network address for the entry

`Netmask`	The netmask for network entries
`uid.gid =3D services`	For each WAL permission; `uid` is the numeric user ID. `gid` is the numeric group ID or a * for wildcard. `services` is the mask of allowed services.

In addition to the header information, the `SLG_NET_INTF` record displays the following fields, which are defined as follows:

Field	Description
`Name`	The interface name
`Address`	The interface address
`Netmask`	The interface netmask
`Flags`	The interface flags
`Min level`	The minimum security level
`Min comparts`	The minimum compartment set
`Max level`	The maximum security level
`Max comparts`	The maximum compartment set
`Authority`	The allowed BSO protection authority flags

In addition to the header information, the `SLG_NET_MAP` record displays the following fields, which are defined as follows:

Field	Description
`Map ID`	The numeric map ID
`level hostlevel =3D netlevel`	For each mapped level; `hostlevel` is the numeric host's level. `netlevel` is the numeric network's level.
`compartment hostcmp =3D netcmp`	For each mapped compartment; `hostcmp` is the host's compartment values. `netcmp` is the network's compartment values.

In addition to the header information, the `SLG_NET_ROUTE` record displays the following fields, which are defined as follows:

Field	Description
Function	Can be add, change or delete, depending on whether the route was added, changed, or deleted, respectively.
Destination	The destination of the route
Netmask	The route netmask

8.8.7.21 Audit Criteria Change Record (SLG_AUDIT)

This record is produced whenever a change is made to change the configuration of the security logging options. The security logging configuration is changed when a security administrator uses the spaudit -e or spaudit -d to enable or disable the security logging options, respectively.

The SLG_LOG_AUDIT parameter must be on to generate this record. To set this parameter, use the Configure system -> Multilevel security (MLS) configuration ->Security log file configuration->Log audit criteria changes? selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e audit command to enable the generation of this record or the spaudit -d audit command to disable the generation of this record.

In addition to the header information, the SLG_AUDIT record displays the following information. The output of this record is shows the state of all the auditing options following any changes that have been made:

```
$ /etc/reduce -t audit
Jul 15 11:09:54 1993  Audit Change o_lvl:0 s_lvl:0 jid:1274 pid:34814
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
 **********
    Login uid: operator(9)

             security auditing state: ON
                   all nami requests: OFF
                     all rm requests: OFF
                  all valid accesses: OFF
              audit criteria changes: ON
                      chdir requests: ON
             config changes (UNICOS): ON
                 Cray Reel Librarian: OFF
```

```
                DAC changes: ON
      discretionary violations: ON
        file transfer requests: ON
                   I/O errors: OFF
        IPnet layer activities: ON
                      job end: ON
               job initiation: ON
              link violations: ON
          mandatory violations: ON
              mkdir violations: ON
        network config changes: ON
            network violations: ON
                 NFS activity: OFF
                 NQS activity: ON
            NQS config changes: ON
          object path tracking: OFF
              operator actions: ON
                 privilege use: OFF
              remove violations: ON
               rmdir violations: ON
              security syscalls: ON
                setuid requests: ON
                system shutdown: ON
                 system startup: ON
             system time change: ON
                    su requests: ON
                 tape activity: OFF
     trusted process activity: ON
  user password on login fail: OFF
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.22 NQS Configuration Change Record (SLG_NQSCF)

This record is produced whenever a change is made to the configuration of NQS by using the qmgr(8) command. These entries are logged through the slgentry(2) system call.

The SLG_CF_NQSCF parameter must be on to generate this record. To set this parameter, use the Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log NQS configuration changes? selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e nqscf` command to enable the generation of this record or the `spaudit -d nqscf` command to disable the generation of this record.

In addition to the header information, the `SLG_NQSCF` record displays the following information:

```
             Function: func
            Violation: type_violation
   Job sequence number: seqno
          Origin host: orighost
          Client host: chost
       Origin user id: origuid
       Target user id: tuid
        Complex/Queue: queue
       Object user id: ouid
       Old validation: oldvalid
       New validation: newvalid
```

The fields are defined as follows:

Field	Description
`Function: func`	The request function being audited.
`Violation: type_violation`	The type of violation; `None(0)`, `Insufficient privilege(1)`, `Not secadm(3)`, and `Not job owner(4)`.
`Job sequence number: seqno`	The NQS job sequence number or `N/A`.
`Origin host: orighost`	The job submission host or `N/A`.
`Client host: chost`	The client host (that is, the host generating the audit record).
`Origin user id: origuid`	The user ID of the caller.

`Target user id: tuid`

The user ID of the caller.

`Complex/Queue: queue`

The applicable queue name or `N/A`.

`Object user id: ouid`

The user ID of the object (that is, manager being added or deleted through `qmgr`).

`Old validation: oldvalid`

The prior NQS user validation type (`None`, `File`, `Password`, `Password if received`, or `Else file`).

`New validation: newvalid`

The new NQS user validation type (`None`, `File`, `Password`, `Password if received`, or `Else file`).

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nqscf
May  9 19:31:49 1993  NQS Config. o_lvl: 0 s_lvl: 0 jid:94 pid:6100
r_ids:[operator(9),operator(9)]   e_ids:[root(0),root(0)]  *******
    Login uid: operator(9)

               Function: Abort request(0)
              Violation: Insufficient privilege(1)
    Job sequence number: 13
            Origin host: drizzle
            Client host: drizzle
         Origin user id: operator(9)
         Target user id: operator(9)
          Complex/Queue: N/A
         Object user id: operator(9)
         Old validation: Password
         New validation: Password
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.23 NQS Activity Record (SLG_NQS)

This record is produced whenever a NQS delete request is made. Delete requests include deleting a queued NQS job and sending a signal to an executing job. The signal can be any valid UNICOS signal, including a kill to delete the job signal. These entries are logged through the `slgentry`(2) system call.

If the caller is an NQS administrator, a Cray ML-Safe process activity record is also generated, assuming `SLG_T_PROC` and NQS is configured to enforce MAC rules (that is, `IC_MAC_COMMAND` is enabled).

The `SLG_ACT_NQS` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log NQS activity?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e nqs` command to enable the generation of this record or the `spaudit -d nqs` command to disable the generation of this record.

In addition to the header information, the `SLG_NQS` record displays the following information:

```
    Login uid: user_id

              Function: func
             Violation: type_violation
    Job sequence number: seqno
            Origin host: orighost
            Client host: chost
         Origin user id: origuid
         Target user id: tuid
   Subject compartments: scmp
    Object compartments: ocmp
```

The fields are defined as follows:

Field	Description
`Login uid:`	`user_id` The user ID.
`Function:`	`func` The request function being audited.

`Violation: type_violation`

The type of violation; `None(0)`, `Insufficient privilege(1)`, `Not secadm(3)`, and `Not job owner(4)`.

`Job sequence number: seqno`

The NQS job sequence number or `N/A`.

`Origin host: orighost`

The job submission host or `N/A`.

`Client host: chost`

The client host (that is, the host generating the audit record).

`Origin user id: origuid`

The user ID of the job's original submitter.

`Target user id: tuid`

The target user ID.

`Subject compartments: scmp`

The caller's active compartment set.

`Object compartments: ocmp`

The executing job's compartment set.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t nqs
May 10 08:01:12 1993  NQS Activity o_lvl:0 s_lvl:0  jid:32  pid:1246
r_ids:[operator(9),root(0)]   e_ids:[root(0),root(0)] ******
    Login uid: operator(9)

               Function: Remote job delete(155)
              Violation: None(0)
    Job sequence number: 32
            Origin host: squall
            Client host: drizzle
         Origin user id: operator(9)
         Target user id: ce(10)
   Subject compartments: none
    Object compartments: none
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.24 Cray ML-Safe Process Activity Record (SLG_TRUST)

This record is produced by UNICOS Cray ML-Safe processes to describe the Cray ML-Safe activities that are performed on behalf of the user. This record provides a higher-level view of Cray ML-Safe process activity than is provided by normal kernel-level auditing.

Execution of the following Cray ML-Safe processes generates this record: `cleantmp`(8), `cll`(8), `fsoffload`(8), `fuser`(8), `init`(8), `jstat`(1), `labelit`(8), `mail`(1), `mailx`(1), `msgdaemon`(8), `reduce`(8), `reply`(8), `passwd`(1), `ps`(1), `sdss`(1), `setfs`(8), `spnet`(8), `su`(1), and `tpconfig`(8). It is written in NQS when a NQS administrator is allowed to bypass the mandatory access control checking on job deletion and status requests.

Each `SLG_TRUST` record contains the process name, a list of the permitted privileges for the process, and a brief text description of the the Cray ML-Safe activity performed by the process.

The `SLG_T_PROC` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log trusted process activity?` selection in the UNICOS Installation and Configuration Menu System. Also, for the `jstat`, `sdss`, `fuser`, and `ps` commands to generate a Cray ML-Safe process activity record, the `MAC_COMMAND` parameter must be enabled. See Section 8.4.2.6, page 191, for more information.

You can also use the `spaudit -e trust` command to enable the generation of this record or the `spaudit -d trust` command to disable the generation of this record.

In addition to the header information, the `SLG_TRUST` record displays the two following formats of information. The first format is used when no object label information is needed; the second is used when object label information is

needed. The second format is generated only by the `labelit`(8) and `setfs`(8) commands:

```
Process: process name
permitted privileges: list of process permitted privileges
      privilege text: process privilege text used
      process action: description of process action

Process: process name
permitted privileges: list of process permitted privileges
      privilege text: process privilege text used
minimum compartments: object minimum compartments
maximum compartments: object maximum compartments
       minimum level: object minimum level
       maximum level: object maximum level
              device: device number
      process action: description of process action
```

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t trust
Jul 15 10:37:11 1993 Trusted Process o_lvl:0 s_lvl:0 jid:1254 pid:34142
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator(9)

        Process: spnet
        permitted privileges: PRIV_ADMIN,PRIV_DAC_OVERRIDE,
                              PRIV_MAC_READ,PRIV_MAC_WRITE,
                              PRIV_AUDIT_WRITE,PRIV_AUDIT_CONTROL
             privilege text: TEXT_NULL
             process action: error 17 adding default to NAL

Jul 15 10:37:16 1993  Trusted Process o_lvl: 0 s_lvl:54 jid:1 pid:34156
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
    Login uid: root(0)

        Process: cleantmp
        permitted privileges: PRIV_CHOWN,PRIV_FOWNER,
                              PRIV_DAC_OVERRIDE,PRIV_MAC_UPGRADE,
                              PRIV_MAC_DOWNGRADE,PRIV_MAC_READ,
                              PRIV_MAC_WRITE,PRIV_AUDIT_WRITE,
```

```
                            PRIV_AUDIT_CONTROL
              privilege text: TEXT_NULL
              process action: /etc/cleantmp operator /tmp.mld
                            /jtmp/jtmp.001242a

Jul 15 10:28:50 1993 Trusted Process o_lvl:0 s_lvl:0 jid:1238 pid:33562
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
    Login uid: operator (9)

        Process: ps
        permitted privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_DOWNGRADE,
                            PRIV_MAC_READ ,PRIV_MAC_WRITE,
                            PRIV_AUDIT_WRITE,PRIV_AUDIT_CONTROL
              privilege text: TEXT_NULL
              process action: MAC policy enforced

Jul 15 10:37:16 1993  Trusted Process o_lvl: 0 s_lvl:54 jid:1 pid:34156
    r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)]       **********
    Login uid: root(0)

        Process: labelit
        permitted privileges: PRIV_DAC_OVERRIDE,PRIV_MAC_READ,
                            PRIV_MAC_WRITE,PRIV_AUDIT_WRITE,
                            PRIV_AUDIT_CONTROL
             privilege text: TEXT_NULL
        minimum compartments: none
        maximum compartments: comp24 comp39 comp63
               minimum level: 0
               maximum level: 16
                      device: 0
              process action: File system labeled: /dev/dsk/usa
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.25 Use of Privilege Record (SLG_PRIV)

This record is written to the security log when any system call, except for the `read`(2), `reada`(2), `write`(2), or `writea`(2) system call, invokes a privilege to perform its function or when a system call fails because the calling process lacks a required privilege.

During system call processing, when privileges are checked by the UNICOS kernel, any required active privilege is recorded for that process. Also, for system calls that can only be executed with privilege (for example, the setsysv(2) system call), a privilege failure is recorded if the process that issued the system call does not have the necessary privileges.

The SLG_PRIV record is written for the following system calls: access(2), acct(2), acctid(2), adjtime(2), chacid(2), chdir(2), chkpnt(2), chmem(2), chmod(2), chown(2), chroot(2), cpselect(2), creat(2), dacct(2), exec(2), fchmod(2), fchown(2), fcntl(2), fgetpal(2), fjoin(2), fpathconf(2), fsecstat(2), fsetpal(2), fstat(2), getdents(2), getfacl(2), getpal(2), getsysv(2), getusrv(2), ialloc(2), ioctl(2), jacct(2), join(2), lchown(2), limit(2), limits(2), link(2), listio(2), lsecstat(2), lstat(2), mount(2), nice(2), nicem(2), open(2), pathconf(2), plock(2), ptyrecon(2), quotactl(2), readlink(2), rename(2), restart(2), resume(2), rmdir(2), rmfacl(2), schedv(2), secstat(2), select(2), setdevs(2), setfacl(2), setfflg(2), setgid(2), setgroups(2), setjob(2), setlim(2), setpal(2), setpermit(2), setregid(2), setreuid(2), setsysv(2), settimeofday(2), setucmp(2), setuid(2), setulvl(2), setusrv(2), stat(2), stime(2), suspend(2), tabinfo(2), tabread(2), target(2), trunc(2), ulimit(2), umount(2), unlink(2), upanic(2), utime(2), and wracct(2).

Many system calls can execute without using privilege if the user is not attempting to override a system security policy restriction. These same system calls may also be made by a privileged process that needs to override such restrictions. Because of this situation, privilege failure records are not issued for most system calls. Instead, the failed attempt to perform the requested function is logged as another type of record such as mandatory access failure or discretionary access failure.

The SLG_PRIV record is written when system call processing completes, but before control is returned to the calling process. At this time, a check is made to determine if any privileges were used by the system call or if any privilege failures were recorded. If privileges were used or privilege failures occurred, then a SLG_PRIV record is written.

The SLG_PRIV parameter must be on to generate this record. To set this parameter, use the Configure system -> Multilevel security (MLS) configuration->Security log file configuration->G selection in the UNICOS Installation and Configuration Menu System.

You can also use the spaudit -e priv command to enable the generation of this record or the spaudit -d priv command to disable the generation of this record.

The format is the same as described for the discretionary access violation record, with the addition of the following fields. See Section 8.8.7.6, page 276, for a description of the format. The fields are defined as follows:

Field	Description
`privileged used`	List of privileges that were actually used to perform the described function.
`privilege failed`	List of privileges that are needed by subject, but the subject did not have to perform the given function.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t priv
Jul 15 10:41:49 1993  Privilege Use o_lvl: 0 s_lvl: 0 jid:1259 pid:34239
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),mail(1)]
**********
    Login uid: operator(9)

   Function: stat (147)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm sysadm
                privilege used : PRIV_MAC_READ
             privilege failed : none

Jul 15 10:41:49 1993  Privilege Use o_lvl: 0 s_lvl: 0 jid:1259 pid:34239
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),mail(1)]
**********
    Login uid: operator(9)

   Function: secstat (92)     Violation: NONE (0)
  Subject: Active Compartments : none
            Valid Compartments : none
                   Permissions : suidgid
                         Class : 0
             Active Categories : none
              Valid Categories : secadm sysadm
```

```
  privilege used : PRIV_MAC_READ
privilege failed : none
```

See Section 8.8.8, page 333, for more information on using this command.

8.8.7.26 Cray/REELlibrarian (CRL) Activity Record (SLG_CRL)

There are four types of Cray / REELlibrarian (CRL) activity log records. Each type is defined by the process that logs the record and the class of activity being monitored. The CRL processes that generate the four types are as follows: `tpdaemon`(8), `spset`(1), the CRL daemon, and the CRL client commands. Each CRL record contains a field that defines which of the four processes requested the log entry.

The `tpdaemon` logs access mediation of the tape volumes and the tape files. Access requests are for the information that is resident on the volumes (that is, the tape file contents).

The `spset` command logs the use or attempted use of security administrator privilege to change the security label of a CRL file or volume set.

The CRL daemon logs the following three different classes of catalog access:

- The creation and deletion of volume sets and files
- The requests that change the mandatory access control and discretionary access control attributes of volume sets and files
- The use of CRL administrator privilege.

The CRL client commands (defined as those commands linked to `/bin/RCOM`) log user access of CRL objects (files and volume sets). When appropriate, the client process is specific in its identification of the object whose access is being requested. In other cases, where the CRL request is not object-specific, the user request description is logged.

The `SLG_LOG_CRL` parameter must be on to generate this record. To set this parameter, use the `Configure system->Multilevel security (MLS) configuration->Security log file configuration->Log Cray/REELlibrarian activity?` selection in the UNICOS Installation and Configuration Menu System.

You can also use the `spaudit -e crl` command to enable the generation of this record or the `spaudit -d crl` command to disable the generation of this record.

In addition to the header information, the SLG_CRL record displays the following information:

```
                Subject: Label : level,cmpt
          caller privileges : priv
       requested operation : req_oper
                    result : result
              entry logger : logger
             Object: Label : level,cmpts
```

The fields are defined as follows:

Field	Description
`Subject: Label : level,cmpt`	This is the active level and compartments of the subject requesting the CRL operation.
`caller privileges : priv`	The active privilege set of the caller at the time of the request. This field is valid only for the CRL daemon entries, as the daemon is not aware of the caller's privileges. The CRL daemon assumes the Cray ML-Safe client making the request has validated the requester's privileges and that the client logs the application of any such privileges.
`requested operation : req_oper`	An ASCII description of the CRL operation that was requested. It may take the form of a CRL client/server interface subroutine name (for example, `rl_vid`), a specific CRL daemon object operation (for example, delete), or a generic administrator privilege use message which is quantified by the request parameters field.
`result : result`	An ASCII description of the error that was encountered or `request complete` if no error occurred.
`entry logger : logger`	Describes which process is making the CRL activity log entry.

`Object: Label : level, cmpts`

The level and compartment set of the file or volume set in a specific CRL object reference. In the case of a volume set, it is the lower label of the (possibly) multilevel object.

For a specific volume set object reference, the following fields are used:

Field	Description
`volset name`	The volume set name in valid CRL syntax.
`volset upper level`	The upper level of the volume set. The lower level of the (possibly) multilevel volume set is found in the `Object: Label:` field.
`volset upper compartments`	The upper compartment set of the volume set. The lower compartment set of the (possibly) multilevel is found in the `Object: Label:` field.
`request parameters`	An ASCII string which qualifies the `requested operation` field to help determine the effect of the request.

To display this record, use the `reduce` command as shown in the following example:

```
$ /etc/reduce -t crl
Oct 12 16:22:19 1993  REELlibrarian o_lvl:0 s_lvl:0  jid:135  pid:12066
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)

                  Subject: Label : 0, none
               caller privileges : none
             requested operation : delete
                         result : request complete
                    entry logger : crl daemon
                     volset name : .SL0007
              volset upper level : level0
```

```
          volset upper compartments : none
                      Object: Label : 0, none


Oct 12 16:22:19 1993  REELlibrarian o_lvl:0 s_lvl:0  jid:126  pid:21990
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                   Subject: Label : 0, none
                caller privileges : none
              requested operation : rl_vscratch
                           result : request complete
                     entry logger : crl client
               request parameters : .SL0007



Oct 15 16:05:39 1993  REELlibrarian o_lvl:0 s_lvl:0  jid:64  pid:7618
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                   Subject: Label : 0, none
                caller privileges : none
              requested operation : CRL administrator privilege
                           result : request complete
                     entry logger : crl daemon
               request parameters : rl_nvrec request


Oct 15 08:53:10 1993  REELlibrarian o_lvl:0  s_lvl:0  jid:64  pid:6902
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                   Subject: Label : 0, none
                caller privileges : none
              requested operation : delete
```

```
                        result : request complete
                   entry logger : crl daemon
                      file name : .SL0A09^FILE2
                  Object: Label : 0, none



Oct 15 10:18:19 1993  REELlibrarian o_lvl:0  s_lvl:0  jid:64  pid:6902
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                caller privileges : none
              requested operation : CRL administrator privilege
                           result : request complete
                      entry logger : crl daemon
               request parameters : 'x' mode permission



Oct 15 10:18:41 1993  REELlibrarian o_lvl:3  s_lvl:0  jid:60  pid:91261
    r_ids:[operator(9),operator(9)]   e_ids:[operator(9),operator(9)]
**********
   Login uid: operator(9)


                    Subject: Label : 0, none
                caller privileges : none
              requested operation : rl_vid
                           result : mandatory violation
                      entry logger : crl client
                       volset name : .SL0021
                volset upper level : level3
         volset upper compartments : comp24
                       Object: Label : 3, comp24
```

8.8.8 The reduce Command

The reduce(8) command extracts and formats entries collected by the security log daemon and generates a report of the data. The default is to report all entries

currently in the disk-resident security log (you can select one of the date-defined files to be processed instead of the current log).

On a system with PRIV_SU enabled, you must be root (UID = 0) to use this command.

On a system using the PAL-based privilege mechanism, you must have an active secadm category to use the reduce command.

By using one or more of the reduce options, you can choose the type of event to be extracted and formatted, as explained in the following sections. See the reduce man page in the *UNICOS Administrator Commands Reference Manual* for more information on all the options.

8.8.8.1 Selecting Record Types (-t option)

The -t option allows you to specify the type of entry formatted. The valid security record types are as follows:

Type	Description
go	System startup
stop	System shutdown
tchg	System time change
cchg	System configuration change
dac	Discretionary access change
disc	Discretionary access violation event
mand	Mandatory access event
oper	Not currently used
logn	Login validation process
netw	Not currently used
disk	Not currently used
ssd	Not currently used
eoj	End of job
chdir	Change directory
tape	Tape I/O error
secsys	Security-related system calls
nami	File manipulation system calls
setuid	setuid system calls

`other`	Not currently used
`sulog`	`su` attempts
`netip`	Network IP layer
`nfs`	Cray-to-Cray NFS requests
`xfer`	File transfers
`netcf`	Network configuration changes
`audit`	Security auditing criteria changes
`nqscf`	NQS configuration changes
`nqs`	NQS activity
`trust`	Cray ML-Safe process activity
`priv`	Use of privilege
`crl`	Cray/REELlibrarian (CRL) activity

See the previous sections on auditing records for examples of using the `-t` option.

8.8.8.2 Printing Security Labels in Record Header (`-S` and `-L` Options)

On UNICOS systems, you can print both the security label of the subject and object in the record header by using the `reduce -S` option.

The subject's security label information appears in the following format in the header:

`S_Label:` *level*`,`*comp1*`,`*comp2*`,...,`*compn*

The object's security label information appears in the following format in the header:

`O_Label:` *level*`,`*comp1*`,`*comp2*`,...,`*compn*

When then the `-S` option is used, no subject or object label information is printed in the record body. The format produced by the `-S` option will become the default format in a future UNICOS system release.

The `reduce -L` option prints only the subject's and object's security level as part of the record header. This option has been maintained for compatibility with previous UNICOS releases, but is no longer supported.

8.8.8.3 Selecting Records by Object Label (-O Option)

You can use the reduce -O option to select record entries by the security label of an object. The security label is specified in the following form:

level[,*compartment*[,*compartment*[...]]]

The following example shows what is displayed when the -O option is used; only those records that have the same object label (that is, with a label of security level 4 and compartment comp24) are selected. Also, the new version of the record header is displayed, as the -S option is used:

```
$ reduce -s11231255 -e11231300 -p -S -O 4,comp24
Nov 23 12:55:42 1993  Mandatory      jid:59  pid:5433
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 4,comp24
    Login uid: operator(9)

   Function: stat (147)     Violation: Permission denied (13)
             System call : stat (147)
   Subject: Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : read
   Object: uid: operator(9)  gid: operator(9)  device: 34, 5
                                    inode: 34
                Pathname : /dev/ttyp000  * No current directory *
                   Class : 0
              Categories : none
                    Mode : 20622



Nov 23 12:55:47 1993  Mandatory  jid:13  pid:5458
r_ids:[operator(9),operator(9)]  e_ids:[operator(9),operator(9)]
    S_Label: 0,none  O_Label: 4,comp24
    Login uid: operator(9)

   Function: setfcmp (86)     Violation: NONE (0)
             System call : setfcmp (86)
   Subject: Permissions : suidgid
                   Class : 0
              Categories : none
```

```
          Access Mode : write
Object: uid: operator(9)  gid: operator(9)  device: 34, 22  inode: 693
            Pathname : /tmp.mld/jtmp/jtmp.000013a.mld/000/mxc.5308
                        /mxctcs02_dir/reg_file  * No current directory *
               Class : 0
          Categories : none
                Mode : 100770
```

8.8.8.4 Displaying Path Names (-p Option)

The -p option reconstructs a path name. The path tracking logging option must be enabled in order for this option to work; see Section 8.8.6, page 267, for more information on this option. For each entry that contains a relative path name, this option attempts to reconstruct the full path name being referenced from the previous path history.

For displaying record types that record object-relevant functions, such as the discretionary and mandatory access records, it is helpful to use the -p option. If this option is not used, the only file information displayed is the inode number, and major and minor device numbers, making locating the actual file name time consuming.

8.8.8.5 Tracking a Specific User Name (-l and -u Options)

The -l and -u options of the reduce command allow you to trace a specific user. The -l option tracks a login user ID from the point of login throughout the session. The login user ID is the most reliable identifier in the log entry because it cannot be changed after logging in. This ID is the one for which the password was checked.

The -u option can be used to monitor a login account when a user logs into the account or a setuid call is made to the account.

For example, a user may log in using the name john. Then the user could use the su command to change the user name to ted and then to beth. If you want to see all the activity for user login sessions under the login john, then the -l option should be used.

If you wanted to see all activity done under the user login john (which would include any setuid calls to the login john), then the -u option should be used.

If a user creates a new session with a different real user ID, using the -l option does not work for tracking during the new session. An administrator can tell when a new session is created by a setsid (117) in the Function field of the

security system call audit record (SLG_SECSYS). To track the new session, use the `-l` option to track the new user ID and/or the `-j` option to track the new job ID. The `-j` option is described in the next section.

When tracking sublogin sessions, the records contain either the initial login user ID or the second login user ID. The job ID should be used to track records for sublogins.

The following example shows activity traced for the login `rll`. In this example, `rll` never logged in during the time frame. Rather, the login `jnk` executed under the real user ID of `rll`:

```
$ /etc/reduce -u rll

Apr 29 11:04:12 1991  Setuid Syscall o_lvl:0  s_lvl:0  jid:85 pid:9464
    r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] ******
****
    Login uid: jnk(204)

   Setuid call from root (0) to rll (626) was successful



Apr 29 11:04:12 1991  Discretionary o_lvl:63  s_lvl:0  jid:85 pid:9464
    r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] ******
****
    Login uid: jnk(204)

   Function: open (5)     Violation: Permission denied (13)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
   Object: Level: 63  uid: jnk(204) gid:tng(23510) device:0, 211
   inode: 26
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 100644
```

8.8.8.6 Tracing a User's Login Session (-j Option)

Every user session is assigned a unique job ID at the time of login. This ID can be used to pick out all related activity of a single interactive or batch session by using the -j option of the reduce command. The following example shows how to use the -j option:

```
$ /etc/reduce -j 85

Apr 29 09:05:30 1991 Security Syscall o_lvl:0  s_lvl:0 jid:85 pid:2327
  r_ids:[jnk(204),tng(23510)]   e_ids:[jnk(204),tng(23510)] **********
Login uid: jnk(204)

  Function: setusrv (89)     Violation: NONE (0)
  Subject: Active Compartments : none
           Valid Compartments : none
                 Permissions : suidgid
                       Class : 0
            Active Categories : none
             Valid Categories : secadm



Apr 29 09:05:37 1991 Setuid Syscall  o_lvl:0  s_lvl:0  jid:85 pid:2327
  r_ids:[jnk(204),tng(23510)]   e_ids:[jnk(204),tng(23510)]
*********
Login uid: jnk(204)

   Setuid call from root (0) to jnk (204) was successful

Apr 29 09:05:37 1991 Change Directory o_lvl:0  s_lvl:0  jid:85 pid:2327
  r_ids:[jnk(204),tng(23510)]   e_ids:[jnk(204),tng(23510)] **********
Login uid: jnk(204)

    Relative directory path: /sn131/mktg/tng/jnk
    Absolute directory path: /sn131/mktg/tng/jnk


Apr 29 09:05:48 1991 Setuid Syscall  o_lvl:0  s_lvl:0  jid:85 pid:2344
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)] **********
Login uid: jnk(204)

   Setuid call from jnk (204) to root (0) was successful
```

```
Apr 29 09:10:08 1991  Mandatory    o_lvl:0  s_lvl:0  jid:85 pid:2861
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)] **********
Login uid: jnk(204)

   Function: open (5)     Violation: Security category violation (319)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : read
   Object: Level: 0  uid: root(0) gid:sys(3) device:0, 245 inode:262
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 104755


Apr 29 11:04:12 1991  Setuid Syscall o_lvl:0  s_lvl:0  jid:85 pid:9464
  r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] *********
Login uid: jnk(204)

   Setuid call from root (0) to rll (626) was successful


Apr 29 11:04:12 1991  Discretionary  o_lvl:63  s_lvl:0  jid:85 pid:9464
  r_ids:[rll(626),tng(23510)]   e_ids:[rll(626),tng(23510)] **********
Login uid: jnk(204)

   Function: open (5)     Violation: Permission denied (13)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : write
   Object: Level: 63  uid: jnk(204)  gid: tng(23510)  device: 0, 211
   inode: 26
            Compartments : none
                   Class : 0
              Categories : none
                    Mode : 40700
```

```
Apr 29 11:04:31 1991  Mandatory    o_lvl:0  s_lvl:0  jid:85 pid:9493
  r_ids:[root(0),root(0)]   e_ids:[root(0),root(0)] **********
Login uid: jnk(204)

   Function: open (5)     Violation: Security category violation (319)
   Subject: Compartments : none
             Permissions : suidgid
                   Class : 0
              Categories : none
             Access Mode : read
   Object: Level: 0  uid: bin(2)  gid: bin(2)  device: 0, 245
   inode: 267
            Compartments : none
                   Class : 0
              Categories : secadm
                    Mode : 100755
```

8.8.8.7 Reducing Security Log Input (`-r`, `-R`, and `-f` Options)

The `-r` option of the `reduce` command can be used to save the selected security log entries in raw format. The `-R` option performs the identical function, but while the `-r` option uses a default file to save the entries, you must specify an output file for the `-R` option. The following example shows how to use the `-R` option:

```
$ /etc/reduce -t logn,mand,disc  -R slog.4_1 > /dev/null
$ ls -l slog.4_1
-rw-------   1 root     root       16544 Apr  2 00:01 slog.4_1
```

If a redirect to `/dev/null` is not done (as shown in the previous example), `reduce` echos all of the records that are being placed into the raw log.

The resultant file can be viewed later by using the `reduce -f` command. This file could then be moved offline to another storage medium to conserve space.

8.8.9 Monitoring Security-relevant Events

The `spcheck`(8) command checks a variety of security-relevant events on a UNICOS system. It should be used on a daily basis to detect actual or potential security breaches.

The -a, -g, -l, -p, -q, and -w options are of special importance when checking a UNICOS system:

- The -a option reports users who have the administrative category and/or permissions (you need both root permission and the secadm category to use this option).
- The -g option reports users in groups root, adm, bin, and sys; the -G option does the same thing, but uses the group names listed in /etc/grpcheck. The /etc/grpcheck file contains valid group names and has the following format:

  group1 group2 group3 ... groupn

  You need root permission to use this option.
- The -l option lists any user login IDs that have not logged in for at least 14 days or 180 days. Also, it lists amp;.profile and amp;.cshrc files that can be written by anyone (you need root permission and the secadm category to use this option). See the following example.
- The -p option reports users with duplicate IDs, users who cannot change their password, and users whose passwords do not expire (you need root permission and the secadm category to use this option).
- The -q option reports users who have 10 or more su command failures in the current sulog file (you need root permission to use this option).
- The -w option reports files in the system that can be written by any system user.

This command has some options that can be used by users other than the security administrator. See the *UNICOS Administrator Commands Reference Manual* for more information on the command and its options.

The following example shows how to use the spcheck command:

```
$ /etc/spcheck -l

 =====Users with no login for 14 days =====
        root
        sync
        bin
        sys
        adm
        cron
```

```
          nqs
          uucp
          ce
          ttg
          lms
          mac

=====Users with no login for 180 days =====
          llm
          pim
          mcm

=====Users with .profiles or .cshrc writable by anyone =====
          cet
          kan

$ /etc/spcheck -q

 =====User's with excessive su failures in /usr/adm/sulog=====

        jnn   had 10 su failures on 2/22
```

8.8.10 Security Violation Error Codes

Table 11 describes the return codes that are set in the security log to indicate a security policy violation.

Table 11. Security Violation Error Codes

Error condition	Error code	Description
ESYSLV	300	Indicates the specified security level falls outside the allowed security level range of the process, file system, or UNICOS system.
EREADV	301	Indicates an attempt to gain read access to a file failed because the user's active security level was less than the file's security level.
EWRITV	302	Indicates an attempt to gain write access to a file failed because the user's active security label was not equal to the file's security label.

Error condition	Error code	Description
EEXECV	303	Indicates an attempt to gain execute/search access to a file failed because the user's active security level is less than the file's security level.
ECOMPV	304	Indicates the specified security compartment falls outside the allowed security compartment range of the file.
EMANDV	305	Indicates a mandatory access violation.
EOWNV	306	Indicates that the user is not the owner of the object to be accessed.
ELEVELV	307	Indicates that the requested security level is outside the user's security-level range.
ESECADM	308	Indicates that a subject who does not possess the proper authorization attempted to perform an operation available only to properly authorized users.
EFLNEQ	309	Indicates that a security level violation occurred during the mounting of a file system.
ENOTEQ	310	Indicates buffer level is not equal to the file level.
EPERMIT	311	Indicates a permission violation; the appropriate permission is required.
EACLV	312	Indicates an ACL access violation.
ENOACL	313	Indicates that no ACL list was found.
ESLBUSY	314	Indicates that an attempt to open the security log (`/dev/slog`) failed because pseudo device `/dev/slog` was already open to another process.
ESLNXIO	315	Indicates an attempt to open the security log (`/dev/slog`) for other than a read operation.
ESLFAULT	316	Indicates a read error on the security log (`/dev/slog`).
ESLNOLOG	317	Indicates that the security log state was `OFF` when a security log request was made.
EINTCLSV	318	Indicates the requested integrity class falls outside the allowed integrity class range of the process, or the UNICOS system. The class value is no longer used.
EINTCATV	319	Indicates the requested category is not included in the allowed categories of the process, or the UNICOS system.

Error condition	Error code	Description
ENONAL	320	Indicates that no network access list (NAL) entry was found.
EMNTCMP	321	Indicates that a security compartment violation occurred when a mount file system request was processed.
EFIFOV	322	Indicates a security FIFO violation.
EAPPNDV	323	Indicates that a security violation occurred on an append request.
ETFMCATV	324	Indicates user requested an illegal combination of categories
ECOVERT	325	Indicates a user performed a legal operation that has created a possible covert channel condition.
ERCLSFY	326	Indicates an attempt has been made to change the security label of a file that does not reside in a wildcard directory.
EPRLABEL	327	Indicates that security label printing is disabled.
ENONSECURE	328	Indicates that a MLS system call was made on a system running a non-MLS kernel.
ESECFLGV	329	Indicates a request to set file security flags has failed because the requested flags are not allowed for the system.
EHOSTNAL	330	Indicates access to or from an unauthorized host or workstation was attempted. The host is not authorized in the NAL.
ESLVLNAL	331	Indicates a security level was detected outside of the range of security levels authorized for the host in the NAL. The kernel detected this condition when processing the Internet Protocol (IP) security option associated with a datagram.
ESCMPNAL	332	Indicates a security compartment was detected outside of the range of compartments authorized for the host in the NAL. The kernel detected this condition when processing the IP security option associated with a datagram.
EMODENAL	333	Indicates an illegal mode (send or receive) of transfer was attempted by a host or workstation.
ESLVNIF	334	Indicates a security level was detected outside of the range of security levels authorized for the UNICOS network interface (I/F). The kernel detected this condition when processing the IP security option associated with a datagram.

Error condition	Error code	Description
ESCMPNIF	335	Indicates a security compartment was detected outside of the range of security compartments authorized for the host in the UNICOS I/F. The kernel detected this condition when processing the IP security option associated with a datagram.
ESOCKLVL	336	Indicates an illegal attempt was made to change the security level of a single level socket (SLS) connection. Except for a privilege granted by the security administrator (for example, NFS), all socket connections are created as SLS.
ESOCKCMP	337	Indicates an illegal attempt was made to change the security compartment of a SLS connection. Except for a privilege granted by the security administrator (for example, NFS), all socket connections are created as SLS.
ENFSAUTH	338	Indicates the proper authentication credentials were not passed to NFS.
ESLVLNRT	339	Indicates a security level was detected outside of the range of security levels authorized for the network route selected, or a route with the correct sensitivity label could not be found.
ESCMPNRT	340	Indicates a security compartment was detected outside of the range of security compartments authorized for the network route selected, or a route with the correct sensitivity label could not be found. The kernel detected this condition when selecting routes.
EBADIPSO	341	Indicates an illegal IP security option was detected by the kernel. The kernel performs integrity and security checks against each IP security option received.
ENOIPSO	342	Indicates an IP security option did not accompany an incoming datagram. The kernel detects this condition by using IP security option information defined in the NAL for each host/workstation.
ESLVLMAP	343	Indicates a translation error was detected when mapping the security level (between UNICOS form and network form). When necessary, using the NAL, the kernel translates the UNICOS security label to the network security label (for outgoing datagrams) and translates the network security label to the UNICOS security label (for incoming datagrams).

Error condition	Error code	Description
ESCMPMAP	344	Indicates a translation error was detected when mapping the security compartment (between UNICOS form and network form). When necessary, using the NAL, the kernel translates the UNICOS security label to the network security label (for outgoing datagrams) and translates the network security label to the UNICOS security label (for incoming datagrams).
EAUTHFLG	345	Indicates an authority protection violation was detected for either an incoming or outgoing datagram with a Basic Security Option.
EIPSOMAP	346	Indicates no translation table was available to translate the security label for a given host connection. The kernel could access the mapping table identified in the NAL for the host.

8.9 NQS Operations

For more information on using NQS on a UNICOS or Cray ML-Safe system configuration, see the *NQE Administration*.

8.10 Tape Operations

For more information on using tapes on a UNICOS or Cray ML-Safe system configuration, see *Tape Subsystem Administration*.

Note: If your site plans to allow nonadministrative user access to tapes on a Cray ML-Safe system tapes, Cray/REELlibrarian (CRL) is required. CRL is not required on a Cray ML-Safe system if administrative-only access is allowed.

Sites are not required to run tapes with a Cray ML-Safe system configuration; in this case, CRL is not required. CRL is licensed and charged separately from the UNICOS system. See the *Cray/REELlibrarian (CRL) Administrator's Guide* and the *Cray/REELlibrarian (CRL) User's Guide* for more information on using CRL on UNICOS and Cray ML-Safe system configurations.

8.11 TCP/IP Operations

For information on TCP/IP operations on a UNICOS system with the MLS feature, see the *UNICOS Networking Facilities Administrator's Guide*.

8.12 UNICOS NFS Operations

For more information on NFS operations on a UNICOS system with the MLS feature, see the *UNICOS Networking Facilities Administrator's Guide.*

8.13 MLS Data Migration Operations

The data migration facility supports the UNICOS and Cray ML-Safe system configurations. See the *Cray Data Migration Facility (DMF) Administrator's Guide* for more information.

Administration of Online Documentation [9]

This chapter describes the administrative procedures required for the online glossary, the Cray message system, and local man pages.

9.1 Modifying Online Glossary Files

The UNICOS `define`(1) utility allows quick, online retrieval of Cray technical terms and their definitions, as well as terms and definitions added by a local site, that match a specified search string. With the `define` utility, Cray provides a definitions file called `Craydefs_i`, which contains embedded keywords. Sites can modify this definitions file or can add local definitions files by using the `builddefs`(1) utility.

9.1.1 Modifying the Cray Definitions File

The following procedure shows how to use `builddefs` to modify the Cray definitions file that the `define` utility uses:

Caution: If you make changes to the `Craydefs_i` definitions file, the changes will be lost when a UNICOS revision or update is installed. In this case, back up the `Craydefs_i` file, do the installation, and then reapply the changes you backed up.

1. Copy the `Craydefs_i` definitions file, which has embedded keywords, from the default definitions directory, `/usr/lib/define`, to your working directory.
2. Edit the file to make the desired changes.
3. Run `builddefs` on the edited file, as follows (`Craydefs_i` is the input file, and `Craydefs` is the output file):

   ```
   builddefs CRAYdefs_i CRAYdefs
   ```

   This command produces a `Craydefs` file, which is a definitions file without embedded keywords and a `Craydefs_k` file, which is a keyword file.

4. Set the DEFINEDIR environment variable to the working directory (*directoryname*) that contains the Craydefs and Craydefs_k files. For the Korn and standard shells, set the variable, as follows:

   ```
   DEFINEDIR=directoryname
   export DEFINEDIR
   ```

   For the C shell, set the variable as follows:

   ```
   setenv DEFINEDIR directoryname
   ```

5. Test the modified file by using the define utility on selected terms.
6. Install the Craydefs_i, Craydefs, and Craydefs_k files in the /usr/lib/define directory.
7. If necessary, reset the DEFINEDIR environment variable. For the Korn and standard shells, reset the variable as follows:

   ```
   unset DEFINEDIR
   ```

   For the C shell, reset the variable as follows:

   ```
   unsetenv DEFINEDIR
   ```

9.1.2 Creating a Local Definitions File

The following procedure shows how to use the builddefs(1) utility to create a local definitions file for use by the define(1) utility. When multiple definitions files are in the definitions directory, the define utility reads the files in alphabetical order; that is, it searches file aaa before file bbb.

Caution: If your site begins the installation process from a clean partition, your local definitions files installed in the default define directory, /usr/lib/define, might be removed during the installation of a UNICOS revision or update. In this case, back up your local definitions files, install the UNICOS revision or update, and then reinstall your local definitions files.

1. Prepare an input file that contains embedded keywords according to the keywording rules contained in the following section.
2. Run builddefs on the local file, as shown in the following example. *sitedefs*_i is the input file, and *sitedefs* is the output file. This command produces a *sitedefs* file, which is a definitions file without embedded keywords, and a *sitedefs*_k file, which is a keyword file.

   ```
   builddefs sitedefs_i sitedefs
   ```

3. If you want to test how your local file works, set the DEFINEDIR environment variable to the working directory (*directoryname*) that contains the new formatted definition file. If you do not want to test the local file, skip to step 6. For the Korn and standard shells, set the variable as follows:

   ```
   DEFINEDIR=directoryname export DEFINEDIR
   ```

   For the C shell, set the variable as follows:

   ```
   setenv DEFINEDIR directoryname
   ```

4. Change directories by entering the following command line:

   ```
   cp sitedefs sitedefs_k directoryname/
   ```

5. Test the new file by using the define utility on selected terms.

6. Install the *sitedefs* and *sitedefs*_k files in the /usr/lib/define directory. The define utility reads the files in this directory and searches them in sequence for search string matches.

7. If necessary, reset the DEFINEDIR environment variable. For the Korn and standard shells, reset the variable, as follows:

   ```
   unset DEFINEDIR
   ```

   For the C shell, reset the variable as follows:

   ```
   unsetenv DEFINEDIR
   ```

9.1.3 Glossary Keywording Rules

The builddefs(1) utility reads a definitions file that has embedded keywords to produce a keyword file and a definitions file without embedded keywords. A pair of keywords in the form *++term* marks each definition in the input file. Synonyms for the term, if any, also begin with ++ and follow the keyword line. (Do not mark synonyms in pairs.) You should enter all keywords and synonyms into the definitions file and use the correct capitalization conventions. The keywords and synonyms are recorded in the keyword file in lowercase characters.

Example:

```
++access control list
++ACL
  The access control lists (ACLs) are an extension to the normal
  UNICOS file discretionary access control. ACLs support the ability
  to grant or deny access to a file on any user and/or group
  basis. For more information on ACLs, see the acl(1) man page.
++access control list
```

The `builddefs` utility checks all keywords for keywording errors. The following rules apply to keywording:

- The two ++ symbols that appear in columns 1 and 2 of the intermediate definitions file identify a keyword. The keyword immediately follows the ++ symbols, with no intervening blank spaces and tabs. Empty keywords (that is, ++ with no following text) are not allowed.
- A keyword can consist of up to 48 characters. If a keyword is longer than 48 characters, it will be truncated.
- Each definition must have two keywords (a matching pair). The first keyword indicates the start of the definition. The second keyword indicates the end of the definition.
- Synonyms for a keyword are in the form ++*synonym* and are limited to 48 characters. Do not mark synonyms in pairs.

9.2 Cray Message System

The Cray message system (formerly called the UNICOS message system) consists of tools and procedures for issuing error messages to users from program code and delivering documentation on those messages. The message system is based on the X/Open Native Language System specification.

This section contains information that administrators need to install, maintain, and update message system files under the UNICOS operating system.

Warning: Sites using the Cray ML-Safe configuration of the operating system can use the information and procedures outlined in the following sections to change or add messages. However, for changed messages, you must not alter the original, underlying meaning of the message.

Message system files are easy to install. They are shipped in both source and catalog format, and they are ready to use after they are loaded in the proper

directories. This section focuses on message system terminology, the location of message system files, and procedures for rebuilding message catalogs when message information changes.

9.2.1 Overview

The message system includes the following features that aid in improving error reporting and problem resolution:

- Published guidelines for writing good messages and good message documentation
- Message catalogs, located separately from the program code, that contain the text of the messages issued at run time
- Explanation catalogs that contain a discussion of the error and suggest solutions
- Online user access to message documentation by using the `explain`(1) command
- User control of the message format through the `MSG_FORMAT` environment variable
- Message text source files distributed with the release

These features create the following advantages for products that use the message system:

- Messages are more informative and usable.
- Online and printed explanations are readily available to users and administrators.
- Messages are easier to trace to their source because they contain a unique identifier that includes their product of origin.
- Messages and explanations are centralized in catalogs. The text of both is readily accessible for update and translation.
- Users can change the message format by using the `MSG_FORMAT` variable.

These advantages are achieved through a design that removes error messages from program code and places them in a *message text file*, which also includes explanations for each message.

The message text file is processed into a catalog of messages and a catalog of explanations. Library calls in the program code access the *message catalog* at run time. An accompanying *explanation catalog* contains explanations of the messages in the message catalog. To access these explanations, use the `explain` command.

The system administrator is responsible for installing, maintaining, and updating the message catalogs. The following sections explain how to work with message text files and message catalogs. They describe how to install the message system files so that UNICOS programs can access them. They also describe how you can update your message catalogs if your site wants to add or change message or explanation text.

See the *Cray Message System Programmer's Guide* for a complete description of the message system from a programmer's perspective.

The man pages for the message system routines contain descriptions and examples of the commands, routines, and environment variables that compose the message system. See the following man pages:

- `caterr`(1)
- `catxt`(1)
- `explain`(1)
- `gencat`(1)
- `whichcat`(1)
- `catgetmsg`(3)
- `catgets`(3)
- `catmsgfmt`(3)
- `catopen`(3) and `catclose`(3)
- `nl_types`(5)
- `msg`(7d)

9.2.2 Message System Files

The message system uses the following three kinds of files:

- Message text file
- Message catalog

- Explanation catalog

The message text file is the source file for message system text. The message and explanation catalogs are binary files produced from the message text file by using the `caterr`(1) command. The release tape includes both the source and binary forms of the message system files.

9.2.2.1 File Names

Each type of message system file has a name in the form *group.suffix*. The group code (*group*) identifies the product, and the suffix (*suffix*) identifies the file type.

The group code is any string that relates to the product or products that the file supports. For example, the `segldr`(1) and `ld`(1) loader commands use the `ldr` group code. The `explain`(1) man page lists all of the group codes used by Cray software.

Each type of message file has a different suffix after the group code. The message text file has the suffix `.msg`, the message catalog has the suffix `.cat`, and the explanation catalog has the suffix `.exp`.

Thus, the following three message system files are associated with the SEGLDR product:

File name	Description
`ldr.msg`	Message text file
`ldr.cat`	Message catalog
`ldr.exp`	Explanation catalog

9.2.2.2 File Location

The location of a message system file is determined by its type, text or catalog, and its product.

The message text file (*group*`.msg`) is located in the source tree with other files in the product's program library (for example, the message text file used by the loaders is located in the `/usr/src/prod/segldr/ldr.msg` file).

Most message and explanation catalogs (*group*`.cat` and *group*`.exp`) are installed in the `/usr/lib/nls/En` directory. Some products must be available when the `/usr/lib` file system is not mounted; the catalogs for these products are installed in the `/lib/nls/En` directory.

9.2.3 Installing Message System Files

The release media includes the message text file, message catalog, and explanation catalog for each product that uses the message system. Cray recommends that you install the message files initially without changes. The UNICOS installation process creates the `/usr/lib/nls/En` and `/lib/nls/En` directories on your system and copies the message system files to these directories. With these files in place, the message system functions correctly and issues messages from UNICOS software.

9.2.4 Changing the Message Text File

The message system lets you update the message and explanation catalogs with site-specific information. Situations in which you may want to add site-specific information to existing catalogs include the following:

- A local modification to the code has created the need to add a new message or to change an existing message.
- A particular error condition has a site-specific remedy that you want to describe in the explanation.
- You want to add names or phone numbers for persons or groups to be notified if certain errors occur.
- You are creating a new program and want to use the message system to issue the error messages.
- The site wants to translate the messages into a different native language; catalogs of messages in the target language must be created and installed.

Warning: Local modifications to message and explanation catalogs will be overwritten during the installation of the next UNICOS revision or update that contains those catalogs. Sites must back up their local modifications, install the new catalogs and message text files, and reapply their local modifications.

If catalogs for a product are mistakenly deleted from the system, you may have to rebuild them.

The following sections describe how to edit and rebuild message system files.

9.2.5 Editing the Message Text File

The message text file is the source file for messages and explanations. If you make changes to a product that have an impact on that product's messages, this is the file that you must change.

The message text file contains the following four types of information:

- Message text, preceded by the `$msg` tag
- Explanation text that contains `nroff`(1) formatting codes, preceded by the `$nexp` tag
- Plain ASCII explanation text, preceded by the `$exp` tag
- Comments, consisting of `$<space><`*text*`>`, `$<tab><`*text*`>`, or `$<newline>`

Blank lines are also acceptable within a message text file, but they are ignored during text-to-catalog processing.

Edit the message text file to include new information. Ensure that the resulting file conforms to the format specified in the *Cray Message System Programmer's Guide*.

9.2.6 Rebuilding Catalogs

After new information is incorporated into a message text file, you must rebuild the related catalogs. You can build catalogs in two ways:

- Use the makefile for the product to remake the catalogs
- Use message system commands to remake the catalogs

The first method, using the makefile, is simpler. It requires fewer steps and less intervention on your part. However, it is less flexible because it calls the message system commands in a specific way.

The second method, using message system commands to remake the catalogs, gives you more options, but it also requires that you understand more about the message system commands and how to use them.

The following sections describe the two methods of rebuilding catalogs.

9.2.6.1 Rebuilding with `nmake`

The makefile for each product builds a message and explanation catalog from the message text file. It places these catalogs in the current directory (usually

within the source tree). The `nmake install` command places the catalogs in the proper subdirectory. (It also reinstalls other software in that directory.)

The makefile varies from product to product, but, basically, `nmake` calls the `caterr` command twice. The first call to `caterr` creates a message catalog from the `$msg`-tagged information in the message text file. This message catalog is placed in the message system directory. For a discussion of where catalogs are located in the directory structure, see Section 9.2.2.2, page 355.

The second call to `caterr` creates an explanation catalog from the `$nexp`- and `$exp`-tagged information in the message text file. The explanation catalog is placed in the same directory as the message catalog.

To create an explanation catalog from source material tagged with `$nexp`, `caterr` calls `nroff` and a file of message macros. `nroff` processes the formatting codes embedded in the explanations and passes the formatted text back to `caterr`. `caterr` then completes its catalog creation process.

9.2.6.2 Rebuilding with Message System Commands

The `caterr` command rebuilds message system catalogs from the message text file. You can use `caterr` to rebuild a message catalog or an explanation catalog. Rebuilding both catalogs for a product requires that you invoke `caterr` twice. See the `caterr`(1) man page for details of the syntax.

For example, if changes were made only to a product's messages (not to the explanations), use the `caterr` command to process the messages into an updated message catalog. Use the `-c` option to call `gencat`(1).

The following command rebuilds the `ldr.cat` message catalog from the `ldr.msg` message text file:

```
caterr -c ldr.cat ldr.msg
```

The `caterr` command processes the text file, then calls `gencat`, which creates the new message catalog.

If changes were made only to a product's explanations (not to the messages), use `caterr` to remake the explanation catalog. Use the `-e` option to produce an explanation catalog instead of a message catalog.

The following command rebuilds the `ldr.exp` explanation catalog from the `ldr.msg` message text file:

```
caterr -e -c ldr.exp ldr.msg
```

The `caterr` command processes the text file, then calls `gencat`, which creates the new explanation catalog.

9.2.7 Printing Messages

The messages for any group code can be printed as a document. You might want to do this in either of the following cases:

- Local changes are made to the message file and the site wants to provide an updated message document to users
- Cray has provided the messages only online, but the site wants to provide a printed message document

Follow the steps below to print a message document locally. Throughout this procedure replace *group* with the group code for the product whose messages you want to print.

1. Locate the message text file in the source tree for the product. If you are not familiar with the structure of the source tree, use the following `find` command syntax to locate the message text file. This invocation of the command displays the path name of the file *group*`.msg`.

   ```
   find /usr/src -name group.msg -print
   ```

2. Locate or create a header file for use in printing. Check in the directory where the message text file was found for a file named *group*`.head`. If this file exists, proceed to the next step. If it does not, create a header file that contains at least the following macros. (The `msg`(7d) man page describes the text processing macros used in this header file.)

   ```
   .GC group
   .ST "group Messages"
   .2S
   ```

3. Extract the explanations from the message text file. Use the `catxt` command to perform this step. The following invocation of the `catxt` command extracts the explanations from the file *group*`.msg` and places them in the file *group*`.nexp`. Invoke this command from a directory in which you have write permission. This may require that you copy the message text file (*group*`.msg`) to a directory outside of `/usr/src`.

   ```
   catxt -n group.nexp group.msg
   ```

   If the *group*`.msg` file contains `#include` directives, the files specified in those directives must also be available in your working directory. If

#include directives appear in the file, the messages use symbolic names instead of literal message numbers. Use the following form of the catxt command (instead of the command shown previously) to extract the explanations. The -s option resolves the symbolic names into message numbers.

```
catxt -s -n group.nexp group.msg
```

4. Process the *group*.nexp file with a version of the troff(1) text processor and print the resulting file. The commands to perform this step depend on the target printer. (This procedure assumes that the target printer is connected to a UNIX system networked to the Cray system.)

   If the target printer is capable of printing files output from the device-independent troff (sometimes called ditroff), go to step a. If the target printer is not capable of printing these files, go to step b.

   If you are unsure of the capability of the target printer to accept device-independent troff input, check the lpr(1) command man page for the UNIX system to which the printer is connected (not the lpr man page on the Cray system). If the lpr command accepts the -n option, the printer is capable of handling output from device-independent troff.

   a. Device-independent troff procedure

      Cray systems include device-independent troff as part of the base software release. On your Cray system, use the following command line to process the header and explanation files with device-independent troff:

      ```
      troff -msg group.head group.nexp | lpr -n
      ```

   b. troff procedure

      Many UNIX systems other than Cray systems include troff (not device-independent troff) as part of the base software release. Copy the Cray message system macro file /usr/lib/tmac.sg to the UNIX system connected to the target printer. Also copy the *group*.head and *group*.nexp files from the Cray system to the UNIX system.

      On this UNIX system, use the following command line to process the header and explanation files with troff:

      ```
      troff -t tmac.sg group.head group.nexp > outfile
      ```

Use the following command to print the `troff` output (*outfile*):

```
lpr -t outfile
```

9.3 Local Man Pages

The man(1) command displays online man pages. The `man` command used in the UNICOS release is compatible with the UNIX 4.3BSD `man` command. The following two differences affect how you can implement local man pages:

- The `man` command references unformatted man pages in a set of `man`*x* directories, and formatted man pages in a set of `cat`*x* directories (the source, or unformatted, man pages are not released with the UNICOS system). Directories follow the BSD organization.

Directory	Description
`cat1`	User commands
`cat2`	System calls
`cat3`	Library routines
`cat4`	Special files (devices)
`cat5`	File formats
`cat7`	Miscellaneous information and DWB macro descriptions
`cat8`	Administrator commands
`catl`	(letter l) Local man pages

- The `MANPATH` environment variable lets you maintain local man pages in the directory of your choice. The X Window System `xman` command also uses the `MANPATH` environment variable.

The `MANPATH` environment variable lists the directories that the `man` command should search for man pages. When the `MANPATH` environment variable is not set, the `man` command, by default, searches the `/usr/man` subdirectories for man pages. Users can set the `MANPATH` environment variable to find man pages in directories other than `/usr/man`.

You can install local man pages as either source (unformatted) man pages (in `man`*x* directories) or as formatted man pages (in `cat`*x* directories). When the source page is newer than the formatted page, or when the formatted page is not available, the man command uses the man page macros defined in `/usr/lib/tmac/tmac.uc` to format source man pages.

If you install your local man pages in directories that are not affected by UNICOS upgrades, you will not have to reinstall them after future upgrades; however, users must set the MANPATH environment variable to include the local man page directory.

Man pages installed in the /usr/man manl or catl subdirectories will have to be reinstalled after each UNICOS upgrade. Save your local man pages before beginning the upgrade, then restore those packages to the appropriate location.

Only when man page searches fail on the first specified path does the man command search the next path. For example, by setting the MANPATH environment variable to include the local man page directory before the Cray man page directory, the man command will display local man pages, rather than Cray man pages that have the same name.

For more information about the man command and the MANPATH environment variable, see the *UNICOS User Commands Reference Manual.*

9.3.1 Examples

The following two examples illustrate ways of using the MANPATH environment variable to implement local man pages on your system.

9.3.1.1 Example 1

You can install local man pages in the same directory structure as the associated binary files, such as /usr/local. If /usr/local/man is used for local man pages, you would follow these steps:

1. Create the /usr/local/man directory, then create the cat*x* or man*x* directories under /usr/local/man.
2. Have users add the MANPATH environment variable to their .cshrc or .profile file, as follows:

   In .cshrc:

   ```
   setenv MANPATH /usr/man:/usr/local/man
   ```

   In .profile:

   ```
   MANPATH=/usr/man:/usr/local/man
   export MANPATH
   ```

3. Add the system-wide definition of the MANPATH environment variable to /etc/cshrc and /etc/profile, as follows:

In /etc/cshrc:

```
setenv MANPATH /usr/man:/usr/local/man
```

In /etc/profile:

```
MANPATH=/usr/man:/usr/local/man
export MANPATH
```

4. Install the local man pages in the /usr/local/man/cat*x* or /usr/local/man/man*x* directories. Alternative man directories must follow the subdirectory structure and the naming convention as used in /usr/man. The man page file name convention allows an extension that corresponds to the number of the directory, such as l for catl and manl, plus a one-letter extension, as follows:

   - *file* .1 (b, c, g, m, or X)
   - *file* .3 (c, f, g, i, l, m, n, r, s, u, x, or X)
   - *file* .4 (f, n, or p)
   - *file* .7 (d or X)
   - *file* .8 (c, v, or e)

9.3.1.2 Example 2

Users can have private man pages, or man pages restricted to a specific group of users. To include private man pages in $HOME/man or restricted man pages in /*restricted_man_dir*/man, follow these steps:

1. Add $HOME/man or /*restricted_man_dir*/man to the MANPATH environment variable in the .cshrc or .profile file, as follows:

   In .cshrc:

```
setenv MANPATH /usr/man:/usr/local/man:$HOME/man:/restricted_man_dir/man
```

   In .profile:

```
MANPATH=/usr/man:/usr/local/man:$HOME/man:/restricted_man_dir/man
export MANPATH
```

2. Install the private man pages in the alternative man directories specified in the MANPATH environment variable; those directories must follow the subdirectory structure and the naming convention as used in /usr/man (see step 4 in example 1).

9.3.2 Display Order for Same-name Man Pages

If a local man page has the same name as a Cray man page, the man(1) command displays the man pages in the order found. The man command searches the subdirectories in numerical order, 1-8, and searches the l (local) subdirectory last. For example, man1/cat1 are searched before man2/cat2, and man8/cat8 are searched before manl/catl.

When the MANPATH environment variable is set, the man command searches the next path only when a search fails on the first specified path. The man command will not display all same-name man pages installed in separate search paths.

If you want to display both man pages, do the following:

- To display the Cray man page first, followed by the local man page, install the local man pages in /usr/man/catl.
- To display the local man page first, followed by the Cray page, do the following:

  1. Install the local man page in /usr/man/cat*x* by using the appropriate file extension or suffix for cat*x* with contents clearly marked "Local."
  2. Move the Cray original page to /usr/man/catl (or /usr/man/manl source pages), with the file extension .l, then modify it to mark its contents "Cray original."

User-defined Locales [A]

In order to provide more flexible support for multicultural software interfaces, the UNICOS system supports the concept of a locale. A *locale* is a collection of culture-dependent information used by an application to interact with a user. The information in a locale includes information on the following:

- Sorting or collation (`LC_COLLATE`)
- Character classes and case mapping (`LC_CTYPE`)
- Basic interaction messages (`LC_MESSAGES`)
- Monetary formats (`LC_MONETARY`)
- Numeric formats (`LC_NUMERIC`)
- Time and date formats (`LC_TIME`)

To make use of these features in an application, that application must be written to use the information in a locale or interfaces that access the locale implicitly. Use of this information from a locale can help an application be free of cultural dependencies. Such an application is said to be *internationalized.* The goal is that such an application can then be run by a user and, through the manipulation of environment variables, interact with that user in a more natural manner.

A.1 The `localedef` Utility

Locales are defined using the `localedef`(1) utility. The input to `localedef` is a text file (known as a *locale definition file*) that describes all the attributes of the desired locale. Using this information, `localedef` creates files that can be loaded by application (using the `setlocale`(3) library routine) to establish that locale in the environment of the application.

The `localedef` utility is invoked in the following manner:

```
localedef [ -c ] [ -i localefile ] [ -f charmap ] locale
```

The optional arguments are used as follows:

`-c` Creates the locale even if warning messages are issued. By default, the locale will not be created if any warnings occur.

`-i` *localedef*	Specifies the name of the locale definition file. If not specified, the locale definition will be read from the standard input.
`-f` *charmap*	Indicates the character encoding to be used. Currently two are supported: 646 (which supports the ISO 646 or ASCII character encoding) and 8859 (which supports the ISO 8859-1 character encoding). The default is 646.

The specified locale is the name of the locale to be created. If *locale* contains a slash character, it is interpreted as a path name of a directory to put the locale files in (if the directory does not exist, it will be created). Otherwise, the locale will be created in `/usr/lib/locale`, making it generally available for users.

A.1.1 Character Specifications

In a locale definition file, characters can be specified symbolically or as literal values. The use of symbolic values is preferred, because this allows the locale definition file to be independent of the particular character encoding.

Literal values can either be the specific character itself (assuming that the target locale will have the same encoding for that character as the 646 locale) or a numeric value. Numeric values can be of the following forms (assuming that the backslash (\) is the current escape character):

`\xNN` *For hexadecimal byte values*
`\dNNN` *For decimal byte values*
`\NNN` *For octal byte values*

A multibyte numeric value can be specified by concatenating byte specifications of the above form.

Characters can also be specified symbolically in a locale definition file. For example, the encoding of the characters 'a', '5', or the bell character can be specified as < `a` >, < `five` >, and < `alert` >. The symbolic names for characters in the 646 and 8859 charmap are specified in the following list:

Name	Value
<NUL>	\x00
<SOH>	\x01
<STX>	\x02
<ETX>	\x03

```
<EOT>                    \x04
<ENQ>                    \x05
<ACK>                    \x06
<BEL>                    \x07
<alert>                  \x07
<backspace>              \x08
<tab>                    \x09
<newline>                \x0a
<vertical-tab>           \x0b
<form-feed>              \x0c
<carriage-return>        \x0d
<SO>                     \x0e
<SI>                     \x0f
<DLE>                    \x10
<DC1>                    \x11
<DC2>                    \x12
<DC3>                    \x13
<DC4>                    \x14
<NAK>                    \x15
<SYN>                    \x16
<ETB>                    \x17
<CAN>                    \x18
<EM>                     \x19
<SUB>                    \x1a
<ESC>                    \x1b
<IS4>                    \x1c
<IS3>                    \x1d
<IS2>                    \x1e
<IS1>                    \x1f
<SP>                     \x20
<space>                  \x20
<exclamation-mark>       !
```

<quotation-mark>	“
<number-sign>	#
<dollar-sign>	$
<percent-sign>	%
<ampersand>	&
<apostrophe>	’
<left-parenthesis>	(
<right-parenthesis>	)
<asterisk>	*
<plus-sign>	+
<comma>	,
<hyphen>	-
<hyphen-minus>	-
<period>	.
<full-stop>	.
<slash>	/
<solidus>	/
<0> or <zero>	0
<1> or <one>	1
<2> or <two>	2
<3> or <three>	3
<4> or <four>	4
<5> or <five>	5
<6> or <six>	6
<7> or <seven>	7
<8> or <eight>	8
<9> or <nine>	9
<colon>	:
<semicolon>	;
<less-than-sign>	<
<equals-sign>	=
<greater-than-sign>	>

<question-mark>	?
<commercial-at>	@
<A>...<Z>	A...Z
<left-square-bracket>	[
<backslash>	\
<reverse-solidus>	\
<right-square-bracket>	]
<circumflex>	^
<circumflex-accent>	^
<underscore>	_
<low-line>	_
<grave-accent>	‘
<a>...<z>	a...z
<left-brace>	{
<left-curly-bracket>	{
<vertical-line>	|
<right-brace>	}
<right-curly-bracket>	}
<tilde>	~
<DEL>	\x7f
<PAD>	\x80
<HOP>	\x81
<BPH>	\x82
<NBH>	\x83
<IND>	\x84
<NEL>	\x85
<SSA>	\x86
<ESA>	\x87
<HTS>	\x88
<HTJ>	\x89
<VTS>	\x8a
<PLD>	\x8b

<PLU>	\x8c
<RI>	\x8d
<SS2>	\x8e
<SS3>	\x8f
<DCS>	\x90
<PU1>	\x91
<PU2>	\x92
<STS>	\x93
<CCH>	\x94
<MW>	\x95
<SPS>	\x96
<EPA>	\x97
<SOS>	\x98
<SGCI>	\x99
<SCI>	\x9a
<CSI>	\x9b
<ST>	\x9c
<OSC>	\x9d
<PM>	\x9e
<APC>	\x9f
<nobreakspace>	\xa0

A.1.2 General Syntax of the Locale Definition File

The format of the locale definition file is a list of category specifications. Each category corresponds to the basic groups of locale information: LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME. The general format for these categories is the following:

```
category_name
keyword      value
keyword      value...
END category_name
```

category_name is either LC_MESSAGES, LC_MONETARY, LC_NUMERIC, or LC_TIME. The specific keywords and the valid associated values are detailed below. The possible values can be strings (enclosed in quotes) or integers or

lists of either of these. If the value is a list, then the list elements are separated by semicolons.

Note that the format of the remaining categories, LC_COLLATE and LC_CTYPE, is quite different and is unique for each of these categories. The details of all the category specifications is described in the following sections.

In addition to the list of category specifications, the locale definition file can have the following global statements:

```
escape_char     value
comment_char    value
```

These define the character used to precede comments, and escape the usual meaning of a character. The default values for these are the following:

```
escape_char
comment_char    #
```

The comment character must appear as the first character of a line. It causes localedef to ignore the rest of that line. The escape character is used to specify numeric character constants and to do line continuation. The latter is necessary since each localedef directive must appear on a single line. The escape character allows the breaking up of long lines while allowing localedef to consider such a set of lines as a single line.

The following example shows specification of the LC_MONETARY category:

```
LC_MONETARY
int_curr_symbol      "<U><S><D><space>"
currency_symbol      "<dollar-sign>"
mon_decimal_point    "<period>"
mon_thousands_sep    "<comma>"
mon_grouping         3
positive_sign        "<plus-sign>"
negative_sign        "<hyphen-minus>"
int_frac_digits      2
frac_digits          2
p_cs_precedes        1
p_sep_by_space       0
n_cs_precedes        1
n_sep_by_space       0
p_sign_posn          4
n_sign_posn          4
END LC_MONETARY
```

A.1.3 The LC_MONETARY Category

The LC_MONETARY category describes monetary formatting conventions. The following keywords are recognized by localedef in the category:

int_curr_symbol type: string

The international currency symbol. The value is the three character international currency symbol defined by the ISO 4217:1987 standard followed by a character, such as a space, to separate the currency symbol from the value.

currency_symbol type: string

The local currency symbol.

mon_decimal_point type: string

The symbol used as a decimal point for monetary values.

mon_thousands_sep type: string

The symbol used to separate groups of digits for monetary values.

mon_grouping type: list of integers

Describes how mon_thousands_sep is used to separate digit groups. For the nonfractional part of a monetary value, the digits are separated by mon_thousands_sep into groups of the sizes specified in this list beginning from the least significant digits. All groups should be greater than zero other than the last value which may be -1. If the last value is -1, no further grouping of digits will be done; otherwise, the last grouping value will be used to determine the size of all subsequent groups. As an example, if the value was 1;2;-1 then the value 123456789 would be formatted as 123456,78,9. A typical use would be to separate thousands of digits for an entire value regardless of length. A value of 3 would produce the desired result in this case.

positive_sign type: string

The symbol used to indicate positive monetary values.

negative_sign type: string

The symbol used to indicate negative monetary values.

`int_frac_digits` type: integer

The number of fractional digits printed for values formatted with an international currency symbol.

`frac_digits` type: integer

The number of fractional digits printed for values formatted with a local currency symbol.

`p_cs_precedes` type: integer

This value (indicated in parentheses) indicates whether the international and local currency symbols precede (1) or succeed (0) a positive monetary value.

`p_sep_by_space` type: integer

This value indicates that no space separates the international or local currency symbol from a positive monetary value (0), or if a space separates the symbol from the value (1), or if a space separates the symbol and the sign string if adjacent (2).

`n_cs_precedes` type: integer

This value indicates if the international and local currency symbol precedes the value for a negative monetary value (1) or if the symbol succeeds the value (0).

`n_sep_by_space` type: integer

This value indicates that no space separates the international or local currency symbol from a negative monetary value (0), or if a space separates the symbol from the value (1), or if a space separates the symbol and the sign string if adjacent (2).

`p_sign_posn` type: integer

This value indicates the relative position of the positive sign and a positive monetary value.

Value	Description
0	Parentheses enclose the value and the currency symbol (local or international).
1	The sign precedes the value and the currency symbol.
2	The sign succeeds the value and currency symbol.
3	The sign precedes the currency symbol.

4 The sign succeeds the currency symbol.

`n_sign_posn` type: integer

This value indicates the relative position of the negative sign and a negative monetary quantity. The values are the same as for `p_sign_posn` above.

`copy` type: string

Causes the copying of the `LC_MONETARY` specification from the locale specified as the keyword value. This keyword cannot be combined with any of the other keywords in the category.

This category affects the operation of the `strfmon` () library routine. This information is also available directly from the `localeconv` () library routine.

A.1.4 The `LC_MESSAGES` Category

The `LC_MESSAGES` category describes messages for user interaction. Currently this is limited to the format of simple acknowledgment (yes or no) requests. The following keywords are recognized by `localedef` in the category:

`yesexpr` type: string

An extended regular expression defining the possible value for an affirmative response.

`noexpr` type: string

An extended regular expression defining the possible value for a negative response.

`yesstr` type: string

A string defining an affirmative response.

`nostr` type: string

A string defining a negative response.

`copy` type: string

Causes the copying of the `LC_MESSAGES` specification from the locale specified as the keyword value. This keyword cannot be combined with any of the other keywords in the category.

This information is available directly from the `nl_langinfo` () library routine.

A.1.5 The LC_NUMERIC Category

The LC_NUMERIC category describes numeric formatting conventions. The following keywords are recognized by localedef in the category:

decimal_point type: string

The symbol used as a decimal point for numeric values.

thousands_sep type: string

The symbol used to separate groups of digits for numeric values.

grouping type: list of integers

Describes how thousands_sep is used to separate digit groups.

copy type: string

Causes the copying of the LC_NUMERIC specification from the locale specified as the keyword value. This keyword cannot be combined with any of the other keywords in the category.

This category affects the operation of the printf () and scanf () family of library routines. This information is also available directly from the localeconv () and nl_langinfo () library routines.

A.1.6 The LC_TIME Category

The LC_TIME category describes time and date formatting conventions. The following keywords are recognized by localedef in the category:

day type: list of strings

A list, which must have seven entries, of the days of the week. Example: "Monday";"Tuesday";... (most of the following examples will not use symbolic format for string, for example, "<M><o><n><d><a><y>", for clarity even though this is, strictly, bad form)

abday type: list of strings

A list, which must have seven entries, of the abbreviated names of the days of the week. Example: "Mon";"Tue";...

mon type: list of strings

A list, which must have twelve entries, of the months of the year. Example: "January";"February";...

`abmon` type: list of strings

A list, which must have twelve entries consisting of the abbreviated names of the months of the year. Example: `"Jan";"Feb";...`

`d_t_fmt` type: string

The format of a date and time specification. See the description of the `strftime`(3) library interface for the syntax of time / date format strings.

`d_fmt` type: string

The format of a date specification. See the description of the `strftime`(3) library interface for the syntax of time / date format strings.

`t_fmt` type: string

The format of a time specification. See a description of the `strftime` () library interface for the syntax of time / date format strings.

`am_pm` type: list of strings

A list, which must have two entries, of the names of antemeridian and postmeridian periods of the day. Example: `"AM";"PM"`

`t_fmt_ampm` type: string

The form of a date and time specification using a 12-hour clock qualified by the appropriate entry from the `am_pm` list. See a description of the `strftime` () library interface for the syntax of time / date format strings.

`era` type: list of strings

Defines how years are counted and displayed for each era in a locale. Each element of the list indicates how a specific time range will be displayed. Each list entry identifies the range of dates that it corresponds to and the format that should be applied for that era. More specifically, each entry has the following form:

direction:*offset*:*start_date*:*end_date*:*era_name*:*era_format*

The details of each of these fields is detailed below. Using a simple example, the following describes the BC/AD era convention:

```
"+:1:-0001/12/31:-*:BC:%Ey %EC" ; "+:0:0000/01/01:+*:AD:%EC %Ey"
```

This example will be used to clarify the meaning of the individual fields in an era description.

Field	Description
`direction`	Either a + or - to indicate whether the year of the `start_date` has lower or higher numeric values than the year of the `end_date`.
`offset`	The number of the year closest to `start_date`.
`start_date`	The year, month, and day of the beginning of the era. The year, month, and day should be specified in the format *yyyy/mm/dd*, respectively.
`end_date`	The ending date of the era. This can either be specified in the same format as `start_date` or as either -* (beginning-of-time) or +* (end-of-time).
`era_name`	The name of the era. In the above example, either BC or AD.
`era_format`	The format for the printing days in the era.
`era_d_fmt` type: string	The format of the date in alternative era notation.

`era_t_fmt` type: string

The format of the time in alternative era notation.

`era_d_t_fmt` type: string

The format of the date and time in alternative era notation.

`alt_digits` type: list of strings

The alternate names for digits in a date specification. Example:

```
"1st";"2nd";"3rd";"4th";"5th";"6th" ...
```

Up to 100 alternate names can be specified.

`copy` type: string

Causes the copying of the `LC_TIME` specification from the locale specified as the keyword value. This keyword cannot be combined with any of the other keywords in the category.

This category affects the operation of the `strftime` () and `strptime` () library routines. This information is also available directly from the `nl_langinfo` () library routine.

A.1.7 The LC_CTYPE Category

The `LC_CTYPE` category can be used to:

- Define membership of character classes
- Specify case conversion

The members of character classes (such as `alpha`, `digit`, `xdigit` `punct`, `space`) can be defined as in the following example. This specifies that the `alpha` class contains a-z and A-Z.

```
alpha   <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
        <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>;\
        <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
        <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
```

Alternatively, range specifications can be used. The following is equivalent to the previous example:

```
alpha   <a>;...;<z>;<A>;...;<Z>
```

The possible character classes are the following:

```
alpha   print     phonogram
blank   punct     ideogram
cntrl   space     english
digit   xdigit    number
graph   special
```

Additionally, user-defined character classes can be created. For example, the following defines a character class named `xalpha` that includes all alphabetic characters that are used as hexadecimal digits:

```
charclass      xalpha
xalpha        <a>;...;<f>;<A>;...;<F>
```

It is necessary to declare all user-defined character classes with the *charclass* keyword before the members of that class are specified.

Character class mapping may also be defined via the `toupper` and `tolower` keywords. The following example illustrates this:

```
toupper (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
        (<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
        (<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
        (<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
        (<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
        (<z>,<Z>)
```

Unfortunately, the current version of `localedef` does not support ranges for case mapping, so all of the mapping pairs must be specified explicitly.

A.1.7.1 Character Class and Case Mappings

There are implicit rules and restrictions for building character classes and case mappings, so that all relationships do not need to be specified explicitly.

Membership in a class can implicitly add a character to other classes as well.

- Members of the `upper` or `lower` classes are added to the `alpha` class.
- Members of the `alpha` class are added to the `alnum` class.

- Members of the `digit` classes are added to the `xdigit` and `alnum` classes.
- Members of the `blank` class are added to the `space` class.
- Members of the `alpha`, `digit`, `xdigit`, and `punct` classes are added to the `graph` and `print` classes.

Restrictions on character class membership:

- Members of the `digit` class cannot be in the `upper` or `lower` classes.
- Members of the `alpha` and `xdigit` classes cannot be members of the `space`, `cntrl`, or `punct` classes.
- Members of the `space` character class cannot be members of the `graph` class.
- Members of the `graph` or print classes cannot be members of the `cntrl` class.

Note that the relationships need to be considered in conjunction. For example, since members of the `xdigit` class cannot be members or the `cntrl` class, then neither can members of the `digit` class, since membership in the `digit` class implies members in the `xdigit` class. This is a bit complicated but should not be a problem in defining actual locales, because these relationships simply enforce the logical relationships between classes.

For case conversion, the following actions and restrictions are imposed:

- Each member of the conversion must be members of the upper or lower classes, as appropriate.
- If no conversions are specified, the traditional a-z to A-Z conversion will be done.
- If a `toupper` conversion is specified without a `tolower` conversion, then the `tolower` conversion will be the inverse of the `toupper` conversion.

Note that the a-z to A-Z conversions are not included implicitly in a conversion if that conversion is explicitly defined.

A.1.8 The `LC_COLLATE` Category

Collation controls the relative order of characters and of strings of characters. In general, the ordering of strings and individual characters is independent. However, they are typically closely related.

A.1.8.1 Collation Sequence

The relative order of characters is referred to as the *collation sequence.* It defines the characters referred to by a range in regular expressions, such a A-Z or 0-9. The collation sequence is defined by a simple listing of the characters in order, one per line.

Additionally, it is possible to define a multicharacter sequence as having a unique position in the collation sequence. Such a sequence is called a *multicharacter collating element,* whereas the simpler term *collating element* refers to either a character or a multicharacter collating element. For example, the two-character sequence ch could be treated as a single character for the purposes of the collation sequence (and for string sorting).

The following is a simple example of a collation sequence:

```
LC_COLLATE
collating-element <ch> from "<c><h>"
collating-element <CH> from "<C><H>"
order_start
<a>
<b>
<c>
<ch>
<d>
<z>
<A>
<B>
<C>
<CH>
<D>
<Z>
<one>
<nine>
order_end
END LC_COLLATE
```

This collation sequence reverses the convention of lowercase preceding uppercase characters. Additionally, it defines uppercase and lowercase forms of the multicharacter collating element ch. Also, all digits will succeed alphabetic characters in the collation sequence.

The use of ellipses to indicate ranges of characters is allowed syntax in the locale definition file and is not just a convention for simplifying this example. An ellipsis can also be used before and after the other characters in the collation

sequence to indicate, respectively, all characters earlier or later in the order of the current character encoding, not including the smallest (typically 0) or the largest value.

The keyword `UNDEFINED` can be inserted into the collation sequence. This results in all characters which are not explicitly in the collation sequence being put into the sequence at the point of the `UNDEFINED` statement in the order of their encoded values.

A.1.8.2 String Ordering

Character string ordering can also be specified by extending the syntax described in the preceding example. In general, the locale definition file can describe a multipass ordering of strings with pass-specific ordering rules. Passes can scan strings in forward or reverse order.

Multipass sorting works in the following manner. Two strings are compared on the first pass. If they are not equal, the ordering for the first pass defines the ordering of the strings. If, however, they are equal on the first pass, a second comparison pass will be done. This continues until a pass compares the strings as unequal or the maximum number of passes have been executed.

String sorting is defined by the weights of the collating elements being compared. These are specified by putting the weights to the right of the specification of an element in the collation sequence. There may be up to `COLL_WEIGHTS_MAX` (currently 8) weights specified, each separated by a semicolon. A weight can be any of the following:

1. A character. In this case, the order is indicated by the position of that character in the collation sequence.
2. A multicharacter collating element. The order is indicated by the collating elements' position in the collation sequence.
3. A collating symbol. A collation symbol is a symbol that marks a position in the collation sequence. Once defined, the only purpose for a collation symbol is to define weights for collating elements.
4. An ellipsis. In this case, it refers to the collation value of the character or collation element. It is only valid to use this on a line that begins with an ellipsis or in an `UNDEFINED` statement.
5. The keyword `IGNORE`. In this case the collating element is ignored for the purposes of sorting. One exception to this is if the `position` parameter is specified for the associated collation pass.

The following is an example of a specification of a collation sequence with explicit string ordering information:

```
LC_COLLATE
collating-symbol <LOW>
order_start     forward;backward
UNDEFINED
<LOW>
<a>       <a>;<a>
<b>       <b>;<b>
<c>       <c>;<c>
<d>       <d>;<d>
<z>       <z>;<z>
<A>       <a>;<A>
<B>       <b>;<B>
<C>       <c>;<C>
<D>       <d>;<D>
<E>       <e>;<E>
<F>       <f>;<F>
<G>       <g>;<G>
<H>       <h>;<H>
<I>       <i>;<I>
<J>       <j>;<J>
<K>       <k>;<K>
<L>       <l>;<L>
<M>       <m>;<M>
<N>       <n>;<N>
<O>       <o>;<O>
<P>       <p>;<P>
<Q>       <q>;<Q>
<R>       <r>;<R>
<S>       <s>;<S>
<T>       <t>;<T>
<U>       <u>;<U>
<V>       <v>;<V>
<W>       <w>;<W>
<X>       <x>;<X>
<Y>       <y>;<Y>
<Z>       <z>;<Z>
<one>     <one>;<LOW>
...       ...;<LOW>
<nine>    <nine>;<LOW>
END LC_COLLATE
```

The preceding example is case-insensitive on the first pass but considers case on the second pass. For digits they are considered to be higher than alphabetic characters in the first pass and are sorted according to their numeric value. However, in the second pass they will sort after all the alphabetic characters and will be considered equivalent to each other.

The specification of the weights of the lowercase letters is unnecessary since the default for unspecified weights is to use the location of the collating element in the collation sequence.

The previous example is not very useful for any real-world collation. A more typical use of multipass and multidirection sorting would be in the processing of accents or other diacriticals. The first pass would compare two strings in a forward direction without considering the diacriticals. If the strings were equal, the second pass would compare the strings backward considering the diacriticals significant, as in the following example:

```
LC_COLLATE
order_start     forward;backward,position
<a>               <a>;<a>
<a-acute> <a>;<a-acute>
<a-grave> <a>;<a-grave>
<a-circumflex>    <a>;<a-circumflex>
<a-diaeresis>     <a>;<a-diaeresis>
order_end
LC_COLLATE
```

The use of the keyword `position` in describing the second pass is not significant in this example and is added to give an example of the general format of an `order_start` directive. That format is of a semicolon-separated list of pass-specific parameters. When multiple parameters refer to the same pass, they are separated by commas. For example:

```
order_start     forward;backward;forward,position;backward,position
```

The only valid parameters for a pass are the following:

`forward`	The pass shall scan the string from beginning to end.
`backward`	The pass shall scan the string from the end to the beginning.
`position`	The position of ignored weights will be considered significant. The string with the first mismatched ignored element shall succeed the other string.

Glossary

cylinder

A group of tracks, one from each platter surface, that is under the read and write heads of a disk drive during one rotation. The number of surfaces within the disk device determine the number of tracks per cylinder. Usually, a disk drive has a set of arms, on which the read / write heads are mounted, that can be moved along the platter surface. All of these arms move together as a group, and the tracks under these read / write heads, as a group, are what makes up a cylinder. See also **track**.

explanation catalog

A binary file, produced by the `gencat`(1) command, that contains the text of UNICOS error message explanations. The user accesses and displays these explanations by using the `explain`(1) command. For more information, see also the `explain`(1) man page.

fixed

As in `RECFM=F`, `F` indicates that all records, both logical and physical, in an MVS dataset are the same length.

logical device

One or more physical device slices that the operating system treats as a single device.

logical disk device

A collection of blocks on one or more physical disk or other logical disk devices.

message catalog

A binary file produced by the `gencat`(1) command that contains the text of error messages as they are called from the software at run time.

message text file

The file that contains the source form of the messages and explanations. A message text file can contain messages, formatted and unformatted explanations, and comments.

partition

(1) On Cray MPP systems, a group of processing elements (PEs) and a portion of the barrier synchronization resources that are assigned to one application. (2) A logical or physical grouping of memory and CPUs or processing elements in a computer system so that all process one application; it is a contiguous set of blocks on a logical device that holds a file system.

A partition of a logical device corresponds to a slice on a physical device. In file allocation, partitions permit the distribution of files across the physical devices underlying the logical device on which a file system is mounted. (3) A whole or partial disk unit that consists of an arbitrary number of consecutive tracks on a physical disk device. See **hardware partition** and **IOS partition**.

run level

A software configuration of the system, controlled by the contents of the `/etc/inittab` file (see the `inittab`(5) man page). The two most common run levels are **single-user mode** and **multiuser mode**.

throughput

The rate of data transfer through a computer system. Throughput is an important method of measuring the real work that a system performs; it is limited by the slowest function of the system.

Usually, **throughput** is measured as a function of data measurement from initial input into the system to the completion of output from the system. Throughput is limited (this is a basic application of von Neumann's Law) by the slowest function of the computer system.

track

The area under one read/write head on a platter surface in a disk storage unit. These platter surfaces are usually stacked on top of each other to create a disk pack. The tracks on these platters, looking vertically through the disk pack are composed of groups called **cylinders**. Typically, tracks are divided into records, which are sometimes also called **disk blocks**. The most common disk block size used in UNICOS is 512 Cray words, or 4096 bytes. See also **cylinder**.

Index

M