UNICOS® Basic Administration Guide for CRAY J90™ and CRAY EL™ Series

SG–2416 8.0.3.2

Cray Research, Inc.

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

Requests for copies of Cray Research, Inc. publications should be sent to the following address:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120
USA

Order desk (612) 683–5907
Fax number (612) 452–0141

Cray Research Software Documentation Map

The illustration on the following pages highlights the major body of documentation available for Cray Research (CRI) customers. The illustration is organized into categories by audience designation:

Audience	Description
End users	Those who use the UNICOS operating system, products, applications, or networking software
Application and system programmers	Those who write or modify system or application code on a CRI system for the purpose of solving computer system, scientific, or engineering problems
System administrators	Those who perform system administration tasks, such as installation, configuration, and basic troubleshooting
System analysts	Those who perform advanced troubleshooting, tuning, and customization
Operators	Those who perform operational functions, such as performing system dumps, and those who administer an operator workstation

To use the map, find the audience designation closest to your specific needs or role as a CRI system user. Note that manuals under other audiences may also be of interest to you; manuals are listed only once, underneath the audience to which they most directly apply. Some manual titles are abbreviated. The date in the map's footer tells you when the information was last revised.

For more information

In addition to the illustration, you can use the following publications to find documentation specific to your needs:

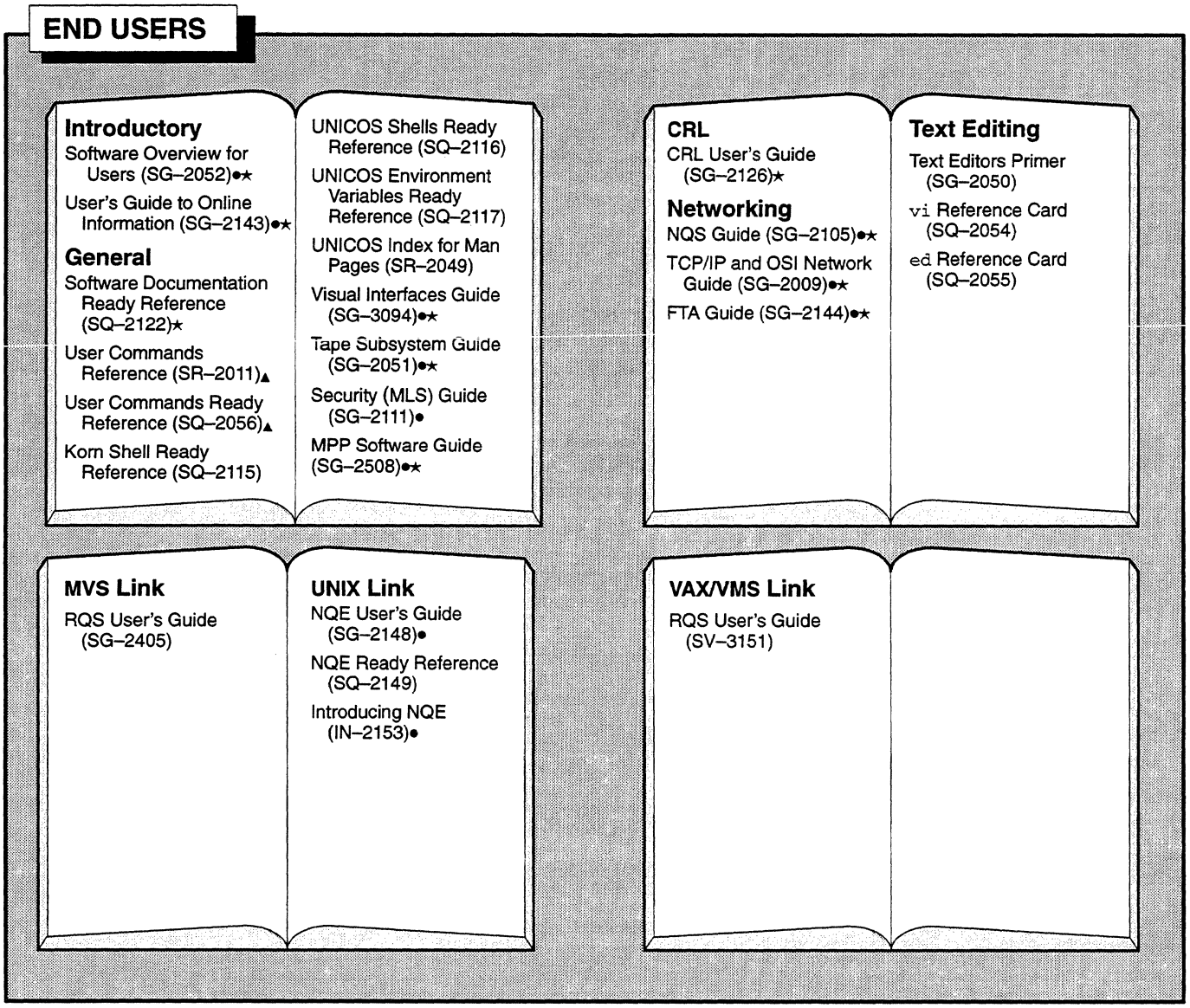
- *Software Documentation Ready Reference,* publication SQ–2122, serves as a general index to the CRI documentation set. The booklet lists documents and man pages according to topic.
- *Software Overview for Users*, publication SG–2052, introduces the UNICOS operating system, its features, and its related products. It directs you to documentation containing user-level information.
- *User Publications Catalog*, publication CP–0099, briefly describes all CRI manuals available to you, including some not shown on the map, such as training workbooks and other supplementary documentation.

Ordering

To obtain CRI publications, order them by publication number from the Distribution Center:

Cray Research, Inc.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120
USA

Order desk (612) 683-5907
Fax number (612) 452-0141



● Available online with CrayDoc
★ Available online with Docview
▲ Man pages available with the `man` command

10/94

APPLICATION AND SYSTEM PROGRAMMERS

Ada
Cray Ada Reference (SR–3014)★

Cray Ada Programming Guide (SR–3082)★

C
Cray Standard C Reference (SR–2074)●★

Cray Standard C Ready Reference (SQ–2076)

Cray Standard C for MPP (SR–2506)●★

CAL for CRAY Y-MP and CRAY Y-MP C90
Reference (SR–3108)●

Symbolic Machine Instructions (SR–3109)

Ready Reference (SQ–3110)

UNICOS Macros and Opdefs (SR–2403)▲

Cray Assembler for MPP
CAM Reference (SR–2510)●★

FORTRAN 77
CF77 Ready Reference (SQ–3770)

CF77 Commands and Directives (SG–3771)●★

CF77 Fortran Reference (SR–3772)●★

CF77 Optimization Guide (SG–3773)★

CF77 Message Manual (SR–3774)

Cray MPP Fortran Reference (SR–2504)●★

Fortran 90
CF90 Commands and Directives (SR–3901)●★

CF90 Fortran Language Reference (SR–3902)●★

CF90 Ready Reference (SQ–3900)

Introducing CF90 SPARC Prog. Env. (IN–2155)●

Introducing DPE (IN–2163)●

Libraries
System Libraries (SR–2080)▲

System Libraries Ready Ref. (SQ–2147)▲

Scientific Libraries (SR–2081)▲

Math Library (SR–2138)▲

Application Programmer's I/O Guide (SG–2168)▲

Application Programmer Library Ref. Manual (SG–2165)▲

Introducing CrayLibs (IN–2167)●

PVM and HeNCE Ref. (SR–2501)●★

PVM Reference Card (SQ–2512)

Loaders
Loader Reference (SR–0066)●★

SEGLDR Ready Reference (SQ–0303)

Loader for MPP
Cray MPP Loader Guide (SG–2514)●

Networking
RPC Reference (SR–2089)●★

Kerberos User's Guide (SG–2409)●★

Programming Tools
UNICOS Message System Programmer's Guide (SG–2121)●★

Compiler Information File (CIF) Reference (SR–2401)●★

CDBX Debugger Reference (SR–2091)●★

CDBX Debugger User's Guide (SG–2094)●★

CDBX Reference Card (SQ–2110)

Program Browser (`xbrowse`) (IN–2140)●

Tuning Guide to Parallel Vector Applications (SG–2182)●

MPP Apprentice Tool (IN–2511)●

Introducing Cray TotalView Debugger (IN–2502)●

Simulators
Cray MPP Simulator Guide (SG–2503)●

Source Control
USM User's Guide (SG–2097)●★

System Calls
System Calls (SR–2012)▲

X Window System
Reference (SR–2101)●★

Ready Reference (SQ–2123)

OPERATORS

OWS-E/IOS-E
OWS-E/IOS-E Reference (SR–3077)▲

OWS-E/IOS-E Ready Reference (SQ–3080)

OWS-E/IOS-E Operator's Guide (SG–3078)

OWS-E/IOS-E Administrator's Guide (SG–3079)

● Available online with CrayDoc
★ Available online with Docview
▲ Man pages available with the `man` command

10/94

SYSTEM ADMINISTRATORS AND ANALYSTS

UNICOS

UNICOS Installation Guide (SG–2112)

Installation Ref. Card (SQ–2411)

UNICOS Installation Tool Menus and Help Files (SG–2412)

UNICOS System Administration (SG–2113)●★

Administrator Commands Reference (SR–2022)▲

Administrator Commands Ready Ref. (SQ–2413)▲

CRL

CRL Administrator's Guide (SG–2127)★

DMF

DMF Administrator's Guide (SG–2135)★

Security and Licensing

UNICOS System Security Overview (SG–2141)★

FLEXlm Guide (SG–2181)●★

UNICOS under UNICOS

UuU Administrator's Guide (SG–2156)●★

CRAY EL Series

IOS Commands Reference (SR–2408)▲

IOS Commands Ready Ref. (SQ–2162)

UNICOS Basic Administration Guide (SG–2416)●★

UNICOS Installation Guide for CRAY Y-MP EL Systems (SG–5201)

IOS Messages (SQ–2402)

Networking

fy Driver Administrator's Guide (SG–2132)

MPP

CRAY T3D Administrator's Guide (SG–2507)●

MVS Link

RQS Administrator's Guide (SG–2406)

VAX/VMS Link

RQS Administrator's Guide (SV–3152)

UNIX Link

RQS Administrator's Guide (SG–2120)

NQE Administration (SG–2150)●

NQE Installation (SG–5236)●

Analysts

File Formats and Special Files Reference (SR–2014)▲

Data Migration MSP Writer's Guide (SN–2098)★

UNICOS Tuning Guide (SR–2099)●

UNICOS `nmake` Card (SQ–2146)

Installation and Configuration Tool Reference (SR–3090)

USCP

Front-end Protocol Internals (SM–0042)★

USCP Optimization (SN–2103)

- ● Available online with CrayDoc
- ★ Available online with Docview
- ▲ Man pages available with the `man` command

10/94

New Features

UNICOS Basic Administration Guide for CRAY J90 and CRAY EL Series SG–2416 8.0.3.2

This manual was revised to add support for CRAY J90 systems and is provided with the UNICOS 8.0.3.2 update. In addition, the following significant changes were made:

- In section 4, an additional sample submenu screen was added to the "Starting and stopping UNICOS system daemons" procedure to clarify the path of submenus traversed to change a system daemon.
- In section 5, the "Disk striping" subsection text was revised, and the sample `/sys/param` configuration file was replaced, which includes multiple IOSs and striped devices.
- Appendix H, "CRAY J90/CRAY EL Software Differences," was added, and other appendixes were reorganized.

This reprint also includes changes that were previously provided in only the online version with the UNICOS 8.0.3 revision and UNICOS 8.0.2.3 update (described below).

The online version of the *UNICOS Basic Administration Guide for the CRAY EL Series* is provided with the UNICOS 8.0.3 revision. The following corrections were made for this online version:

- The Warning at the beginning of each section was revised to reflect that this release is not a B1 evaluated system.
- The "Starting up the system" and "Shutting down UNICOS and the IOS" procedures in section 3 were corrected, and information about the length of time it takes to complete a system shutdown was added to the "Shutting down UNICOS and the IOS" procedure.
- The "Starting and stopping UNICOS system daemons" procedure in section 4 was corrected, and information about determining which daemons are running (not just enabled) was added.
- Section 5 changed as follows: The `diskusg` command description was added to subsection 5.3; HIPPI disk keyword identifiers were added to subsection 5.8.1; the physical device definition `cluster` *number* description was corrected; the "Identifying devices defined on your system and their file system allocation" and the "Modifying your configuration file" procedures were corrected; the `-q` option was added to example 2 of the "Building the file system" procedure; and a CAUTION was added to the "Unmounting file systems" procedure stating that a file system must be idle before you unmount it.
- A Caution was added to the "Restoring a file system without `cpdaemon`" procedure in section 6.
- Minor corrections were made to the `/etc/nu` examples for adding a user and removing a user's login in section 7.
- A note was added to the "Using the menu system to configure your CRAY EL system as an NIS slave server" and the "Configuring your CRAY EL system as an NIS slave server without using the menu system" procedures in section 12 and to the "Configuring the CRAY EL system as an NFS client" procedure in section 13 stating that the `portmap` daemon must be running.

- Minor changes and NQE references were added to section 14, and the "Configuring and starting your local host NQS system" procedure was corrected.
- Support for the new character special tapes interface using `tpdaemon`, SI-3 SCSI controllers, and EMASS, Inc. ER90 devices was added to section 15, as well as the following: STK 9914 9-track tape device type `m` was added, the minor number type was corrected to be 0–31, a `type LOADER` statement parameter value was corrected to be `STKACS`, the `IOP` statement `cluster` parameter description was corrected, the `BANK` statement `number` parameter value and description was corrected, the `SLAVE` statement is now supported only for ER90 devices, three new `DEVICE` statement parameters were added to support ER90 devices, the `user_exit_mask OPTIONS` statement parameter was added, and the "Installing an autoloader" procedure was revised.
- A summary of the content of appendix A was added at the beginning of the appendix, and the subsection on restoring a configuration was revised.
- The `/sys/config.uni` command was changed to be `/sys/param` in appendix B.
- DD-5I specifications were corrected in appendix E, and specifications were added for DD-5S, DD-6S, and DD-7S disk drives.
- Appendix H was added, which provides a strategy for file version numbering.

The UNICOS 8.0.2.3 update includes the online version of the *UNICOS Basic Administration Guide for the CRAY EL Series*. The following minor corrections were made for this online version:

- References to `/etc/setdev` were removed (SPR 78095); options `-r in` and `-n` were added to the backup and restore procedures notes describing the use of the `tpmnt` command (SPR 77271).
- Description of the `-A` *altfile* `dump` command option was added to subsection 6.4.
- Description of the `-v restore` command option was added to subsection 6.6.
- Subsection 5.2.1 was revised as follows: definition of a partition was revised to be one slice on one physical device, additional information about minor device number ranges was added, and reference to `/dev/mkdev.sh` was removed.
- Information about `setfs` was added to subsection 5.2.2.
- Reference to `/ce/bin/olhpa` was added to subsection 5.3.
- Subsection 5.8.2.5.3 text was revised to state that a slice is the name of a logical device, and a note was added to the "Procedure: Identifying devices on your system and their file system allocation."
- A note was added to appendix A stating that you can modify your `.profile` or `.cshrc PATH` statement for easier access to the menu system.

Record of Revision

The date of printing or software version number is indicated in the footer. Changes in rewrites are noted by revision bars along the margin of the page.

Version	***Description***
8.0	March 1994. Original printing. This basic administration guide supports the CRAY EL series of systems that run the UNICOS 8.0 release.
8.0.2.3	June 1994. The online version of this guide is provided; minor corrections are included with the UNICOS 8.0.2.3 update. For details about changes within this manual, see the New Features page.
8.0.3	October 1994. The online version of this guide is provided to support the UNICOS 8.0.3 revision release. For details about changes within this manual, see the New Features page.
8.0.3.2	February 1995. Reprint of this guide is provided to support the CRAY J90 series of systems for the UNICOS 8.0.3.2 update release. For details about changes within this manual since the original printing, see the New Features page.

Preface

This guide describes procedures that system administrators of CRAY J90 and CRAY EL systems running UNICOS frequently perform.

Note

For detailed information about specific topics described in this guide, see "Related Documentation" in each section.

Recommended prerequisite skills

To get the most out of this guide, you should have the following skills:

- User experience with UNICOS or UNIX file structure and utilities and using man pages (see "Online information").
- Ability to read and understand UNICOS or UNIX shell scripts.
- Familiarity with a UNIX text editor, such as `vi` or `ed`.
- Experience with the administration of at least one operating system.

Online information

The following types of online information products are available to Cray Research customers:

- CrayDoc online documentation reader, which lets you see the text and graphics of a manual online. The CrayDoc reader is available on workstations. To start the CrayDoc reader at your workstation, use the `cdoc`(1) command.

- Docview text-viewer system, which lets you see the text of a manual online. The Docview system is available on the Cray Research mainframe. To start the Docview system, use the `docview`(1) command.
- Man pages, which describe a particular element of the UNICOS operating system or a compatible product. To see a detailed description of a particular command or routine, use the `man`(1) command.
- UNICOS message system, which provides explanations of error messages. To see an explanation of a message, use the `explain`(1) command.
- Cray Research online glossary, which explains the terms used in a manual. To get a definition, use the `define`(1) command.
- `xhelp` help facility. This online help system is available within tools such as the Program Browser (`xbrowse`) and the MPP Apprentice tool.

For detailed information on these topics, see the *User's Guide to Online Information*, publication SG–2143.

Conventions

The following conventions are used throughout this manual:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.

Convention	Meaning
`manpage(x)`	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: 1 User commands 1B User commands ported from BSD 2 System calls 3 Library routines, macros, and opdefs 4 Devices (special files) 4P Protocols 5 File formats 7 Miscellaneous topics 7D DWB-related information 8 Administrator commands
`routine()`	Routine names followed by an empty set of parentheses designate a library or kernel routine; for example, `ddcntl()`. Kernel routines do not have man pages associated with them.
variable	Italic typeface denotes variable entries, words or concepts being defined, and explanatory comments in examples and screens.
`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.
`abbreviation`	Underlining indicates the shortest possible abbreviation for a command.
[]	Brackets enclose optional portions of a command line.
...	Ellipses indicate that a preceding command-line element can be repeated.
KEY	This convention indicates a key on the keyboard.

Convention	Meaning
<KEY>	This convention indicates a key on the keyboard.
Command lines using I/O redirection	All command lines you enter that include I/O redirection, such as **2>**. require that you use either the `ksh` or `sh` shell; the `csh` shell will not accept the same I/O redirection command syntax as the `ksh` and `sh` shells.

It is the objective of Cray Research to become compliant with IEEE Std 1003.1–1990 (POSIX.1) and IEEE Std 1003.2–1992 (POSIX.2). This manual reflects those ongoing efforts.

POSIX.2 uses *utility* to refer to executable programs that Cray Research documentation usually refers to as *commands*. Both terms appear in this document.

Ordering publications

The *User Publications Catalog*, publication CP–0099, lists all Cray Research hardware and software manuals that are available to customers.

To order a manual, use one of the following methods:

- Call the Distribution Center in Mendota Heights, Minnesota, at (612) 683–5907.
- Use the CRInform system (CRInform is an online, menu-driven information and problem-reporting service for Cray Research customers).
- Send a facsimile of your request to fax number (612) 452–0141.
- Contact your CRAY J90 or CRAY EL service representative.
- Cray Research employees may choose to send electronic mail to `orderdsk` (UNIX system users).

Reader comments

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

- Send us electronic mail from a UNICOS or UNIX system, using the following UUCP address:

  `uunet!cray!publications`

- Send us electronic mail from any system connected to Internet, using the following Internet addresses:

  `pubs2416@timbuk.cray.com` (comments specific to this manual)

  `publications@timbuk.cray.com` (general comments)

- Contact your Cray Research representative and ask that a Software Problem Report (SPR) be filed. Use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.

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

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

  (612) 683–5600

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

- Use the postage-paid Reader's Comment Form at the back of this manual.

We value your comments and will respond to them promptly.

Contents

List of Procedures

Figures

Tables

Introduction [1]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

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

The role of a system administrator

1.1

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

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

Daily administrative duties may consist of the following functions:

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

Create and maintain a log book

1.2

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

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

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

Major characteristics of UNICOS
1.3

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

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

The following subsections describe the major characteristics of UNICOS.

High-performance I/O
1.3.1

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

File systems
1.3.2

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

Disk devices
1.3.3

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

File system quotas
1.3.4

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

User database (UDB)
1.3.5

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

Resource control
1.3.6

Resource control was added to UNICOS to permit a system administrator to set limits on CPU, memory, tapes, and file allocation. User limits are applied to processes or jobs, and they establish the maximum amount of a resource that can be consumed. You can specify limits for interactive and batch workloads, as well as for per process and per job. This lets a system provide restricted resources for interactive use, without limiting a user's batch resources to the same degree.

Fair-share scheduler
1.3.7

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

System accounting
1.3.8

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

TCP/IP
1.3.9

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

Network Queuing System (NQS)
1.3.10

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

Menu system
1.3.11

UNICOS contains a set of shell scripts, parameter files, and a user interface written in menu specification language (MSL). You may use the menu system to perform configuration changes after you have installed UNICOS. For information about using the menu system, see appendix A, page 415.

Data migration
1.3.12

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

How this guide will help you
1.4

After you boot your IOS and UNICOS software and bring it to multiuser mode by following your UNICOS installation guide (*UNICOS Installation Guide for the CRAY J90 Series*, publication SG–5271, or *UNICOS Installation Guide for the CRAY EL Series*, publication SG–5201) or your *Open First* documentation, this guide will enable you to perform each of the following tasks:

- Establish and maintain basic system security; see section 2.
- Start up and shut down the IOS and UNICOS; see section 3.

- Verify and change date and time of both the IOS and UNICOS; see section 3.
- Start and stop UNICOS system daemons; see section 4.
- Determine existing file systems; see section 5.
- Plan and configure file systems; see section 5.
- Create, label, mount, and check the integrity of a file system; see section 5.
- Monitor disk usage; see section 5.
- Back up and restore a file system; see section 6.
- Create and maintain user accounts; see section 7.
- Communicate with your system users; see section 8.
- Interpret system logs and determine when to "clean up" logs; see section 9.
- Set up Cray system accounting (CSA) and monitor accounting functions; see section 10.
- Add your CRAY J90 or CRAY EL system to an existing network; see section 11.
- Configure NIS; see section 12.
- Configure NFS; see section 13.
- Create and maintain the Network Queuing System (NQS), a batch facility; see section 14.
- Initiate and monitor the tape subsystem; see section 15.

Note

Each procedure that you must follow begins on a right-hand (odd-numbered) page. You can remove procedures from the manual without disturbing additional explanatory text.

This guide also contains several appendixes that may be of interest to you, including basic information about how to use the UNICOS menu system and a summary of software differences between CRAY J90 and CRAY EL systems.

This guide also refers you to other publications for additional information you may need to perform more advanced system administration tasks, such as setting file system quotas.

Where to look for more information

1.5

The following publications contain additional information you will need to administer a CRAY J90 or CRAY EL system:

- *CRAY IOS-V Commands Reference Manual,* publication SR–2170
- *CRAY IOS-V Messages,* publication SQ–2172
- *CRAY EL Series IOS Commands Reference Manual,* publication SR–2408
- *CRAY EL Series IOS Messages,* publication SQ–2402
- *UNICOS System Administration,* publication SG–2113
- *UNICOS Administrator Commands Reference Manual,* publication SR–2022
- *UNICOS User Commands Reference Manual,* publication SR–2011
- *UNICOS File Formats and Special Files Reference Manual,* publication SR–2014
- *Flexible License Manager (FLEXlm) Administrator's Guide,* publication SG–2181
- *UNICOS Installation Guide for the CRAY J90 Series,* publication SG–5271
- *UNICOS Installation Guide for the CRAY EL Series,* publication SG–5201

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

Note

Where UNICOS publications provide information for Cray Research systems that have an I/O subsystem (Model E (IOS-E), this information also applies to CRAY J90 and CRAY EL systems, unless noted otherwise.

For information about	Read
File system space monitoring	SG–2113; df(1) and du(1) man pages
File system quotas	SG–2113
System activity monitoring	SG–2113; sag(1), sar(8), sdc(8), tsar(8),and timex(1) man pages
Cray Research system activity monitor (SAM)	csam(8), sam(8), samdaemon(8), and xsam(8) man pages
Automated incident reporting (AIR)	SG–2113; aird(8), airdet(8), airprconf(8), airsum(8), airtsum(8) man pages
Job and process recovery	SG–2113; chkpnt(1), chkpnt(2), and crash(8) man pages
Reinstalling your system software	SG–5201
Updating your system software	SG–5201
Using the cron and at utilities	SG–2113; at(1) and cron(8) man pages
Configuring network interfaces	SG–2113
Monitoring networks	SG–2113

For information about	Read
Unified Resource Manager (URM) centralizes resource allocation with a formal method of communication	SG–2113
Fair-share scheduler	SG–2113; `shradmin`(8) and `shrdist`(8) man pages
Memory scheduling	SG–2113; `mschedv`(8) man page
Multilevel security (MLS)	SG–2113
UNICOS message system	SG–2113
Data migration facility (DMF)	SG–2113

Accessing online documentation by using CrayDoc

1.6

CrayDoc is a workstation-based electronic documentation reader that includes graphics, hypertext links, and quick information retrieval searches. You can use CrayDoc to view documentation at your workstation, to print selected sections of a book, to print an entire book, to print to a PostScript file for viewing by using a PostScript viewer, or to export the Standard Generalized Markup Language (SGML)-tagged information for further filtering or viewing.

CrayDoc is delivered on a CD-ROM disk that contains the reader and the associated documentation.

The `cdoc`(1) man page provides the information you need to set up your workstation so that you can use CrayDoc. After you have set up your workstation appropriately, enter one of the following commands to start the reader:

For Motif:

```
$ cdoc
```

For OpenWindows on a Sun workstation:

```
$ cdoc.o1
```

You do not have to be in any particular directory to run the reader.

After you enter the `cdoc` or `cdoc.o1` command, the CrayDoc copyright window appears briefly on your screen. After the copyright window disappears, the cursor changes to a watch icon to indicate that the reader and the online books are being loaded. After a few seconds the CrayDoc Library window appears. You are now ready to use CrayDoc.

For more information about CrayDoc, see the following documents:

- *CrayDoc Reference Card*, publication SQ–6101
- *CrayDoc Installation Guide*, publication SG–6103
- *Reader Guide to UNIX* (available online in CrayDoc)
- *User's Guide to Online Information*, publication SG–2143 (available online in CrayDoc)
- `cdoc`(1) man page

Accessing online documentation by using Docview

1.7

Most of the documentation you need to administer and use your system is available online as either man pages (for an explanation of man page, use the `man` command) or as Docview files. The Docview utility provides online access to an ASCII version of Cray Research documents. The *User's Guide to Online Information*, publication SG–2143, and Docview help screens will guide you through the process of viewing a document online. To enter the Docview program, enter the following command at the system prompt:

```
$ docview
```

The following command menu will appear:

```
                    D O C V I E W
          On-line Documentation System Command Menu

Please enter a command at the menu> prompt.

a[list]                          List docnames in alphabetical order
c[list]                          List docnames by subject category
d[list]                          List docnames by date last submitted
f[ind]  [string]                 Find keywords and corresponding docnames
                                 associated with "string"
p[revious]                       Return to the previous command mode
v[iew]  [docname] [keyword]      View passage "keyword" in document "docname"
w[rite] [docname] [keywords]     Write passages specified by "docname"
                                 and "keywords"
h[elp]  [topic]                  Display help for the current screen
                                 or a Docview topic or command
m[enu]                           Display this menu
q[uit]                           Quit from Docview

Enter "help quick" for a quick look at how to use Docview menu>
```

The a command lists the manuals (*docnames*) available under Docview. The `view` *docname* `oindex` command lists the keywords of the selected manual (*docname*).

UNICOS online glossary

1.8

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

```
$ define stripe

striped disk slice

  A logical disk device composed of two or more
physical disk slices (also known as members).

striped group

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

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

Basic System Security [2]

Note

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

This section does **not** apply to systems that use the UNICOS multilevel security feature (MLS) or to Trusted UNICOS systems. For information on using the UNICOS MLS system and Trusted UNICOS, see *UNICOS System Administration*, publication SG–2113.

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. UNICOS includes programs that provide the necessary tools for the creation of a set of procedures that lets you automate much of the daily work of monitoring system security. This section discusses security issues in four areas: system security (ensuring that the superuser privileges are safe), user security, partition security, and tape device access.

Related basic system security documentation

2.1

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

- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `diskusg`(8) man page
- *UNICOS System Administration*, publication SG–2113
- *UNICOS User Commands Reference Manual,* publication SR–2011: `chown`(1), `du`(1), `find`(1), `login`(1) `su`(1), and `umask`(1) man pages

Superuser privileges
2.2

As in standard UNIX systems, in the UNICOS operating system, the user identification number (user ID) of 0, associated with the account named `root`, has special privileges and may override the security features that govern the activity of normal users. Such a user is referred to as a *superuser*, and the superuser'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 superuser privileges remains solely in the hands of the administrator and the administrator's staff. Failure to guard this access allows unauthorized users to acquire superuser 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.

Password security for superuser
2.2.1

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

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

- Ensure that the `root` password is not obvious and is very difficult to guess. Do not use a normal word in any language that might be known to a majority of the system's users. Additionally, capitalizing a random letter or two (not the first letter of the password), or including a punctuation character or a numeral in the password, or both, helps to keep superuser privileges safe from an intruder who is trying to guess the `root` password.
- Change the `root` password frequently, at least once a month.
- Do not write down the `root` password.
- Ensure that the `root` password is known to as few people as possible; generally, these should be the system administrator and the administrator's staff.

You can monitor the use of the `root` password, and catch potential security breaches, by checking the `/usr/adm/sulog` file. (For information about system logs, see section 9, page 219.) You can compare the log entries against the names of

users known to have valid authorization, alerting the administrator to unauthorized superusers (a security breach) or users who are repeatedly trying to gain superuser privileges (a security risk).

Physical security 2.2.2

A person who has access to the IOS console and a knowledge of how to halt and reboot the system could do so and thus acquire unauthorized superuser privileges.

To guard against this possibility, your IOS console and your system itself should be physically accessible only to those persons who genuinely need that access, and your IOS console should not be left unattended while you are logged into the system. If this is not possible, your IOS console should at least be monitored to prevent unauthorized persons from trying to enter commands on the system console.

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

`setuid` *programs* 2.2.3

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

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

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

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

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

As administrator, you should check the list of known `setuid` or `setgid` 0 programs regularly to ensure that none have been modified since the last check and that any modifications that have been made are known (that is, were made by you or a member of your staff). Unknown modification of a `setuid` or `setgid` 0 program may indicate a security breach. To generate a checksum and block count list of a file, you can use the `sum` command; to do this, you can pipe the preceding `find` command line through the `sum` command as follows and then compare any changes in sizes:

```
| sum > filename
```

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

Because checking the entire system for `setuid` or `setgid` 0 programs uses a lot of I/O and CPU time, you should perform this check during off-peak hours. To make the task less obtrusive, use the `cron` or `at` command to perform the check automatically.

root PATH 2.2.4

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

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

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

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

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

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

For example, suppose a knowledgeable user creates a program that mimics a commonly used system utility, such as `ls`. In addition to performing the expected system function (listing the files in the current directory), the new `ls` utility makes a copy of a program such as `sh` and turns on the `setuid` bit on the copy. An unsuspecting superuser who has the current directory in `PATH`, having changed directories to a user's directory and inadvertently run the bogus `ls`, then creates a `setuid` 0 shell, which gives anyone executing it complete control over the system.

User security
2.3

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

The `umask` *command*
2.3.1

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

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

You should choose a umask value that restricts default access permissions to a level appropriate to the desired security of the system. However, because users can override the default value by using the `umask` command themselves, do not make the default umask value too stringent, because users may find that the default value interferes with their work. For instance, if two users are working on a joint project, and each needs access to the other's files, they may want to change their umask value to open their files. As an alternative, they may want to use the groups mechanism; see subsection 2.3.3, page 18.

Default PATH *variable* 2.3.2

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

The users expect to be able to execute programs in the current directory without preceding the program name with `./` to indicate the current directory explicitly. However, many UNICOS systems traditionally place the current directory first in the PATH, which can make the users vulnerable to a security breach. The current directory should thus be the last entry in the default PATH, after the normal system directories.

User groups 2.3.3

You can enhance user security by the careful placement of users into groups. Generally, when deciding on the placement of users into groups, you should use factors external to the system. Some examples might be the following:

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

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

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

File-owner fraud
2.3.4

Neither the listed owner ID of a file nor its location in the directory tree always leads to the actual creator and owner of the file. That is, users tend to think of the files residing in their home directory as their only files, even though they may own files in another home directory, such as those being used for a project that involves several other users. Also, files that reside in one user's home directory tree may be owned by another user.

Users may become confused by this situation and then use the `chown` command to change the ownership of some of their files to another user (most likely one who will cooperate and give the file back when requested). To get a general idea of the users who trade ownership of files, you can use the `diskusg` and `du` commands together.

Login attempts
2.3.5

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

Partition security
2.4

When administered properly, the UNICOS file system should provide adequate protection for user and system files. To enhance system security, however, mount partitions only when they are needed. (A *partition* is one slice on one physical device.) In particular, if there are users who will be allowed dedicated time on your system, you can provide extra protection for those

accounts by not mounting their files during nondedicated time or by not mounting the file systems that contain other users' accounts during dedicated time. (For more information about file systems, see section 5, page 57.)

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

Example:

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

Tape device access
2.5

For CRAY EL systems, you should not configure up a given tape drive in the tape daemon if it also has a UNIX character special file defined for it in the `/dev` directory. To avoid interference between the tape daemon and the standard UNIX tape access commands, you should define each tape device for **only one** of these two access paths:

- Tape daemon: define a tape device for tape daemon usage by including it in your `/etc/config/tapeconfig` file (see subsection 15.4, page 368).
- Standard UNIX commands: define a tape device for standard UNIX tape command access by making a UNIX character special file for it in the `/dev` directory using the `/etc/mknod` command (see subsection 15.3, page 362).

Caution

Interference between the tape daemon and the standard UNIX tape commands for a given device may cause severe problems, including data corruption, system hangs, or system panics.

Startup and Shutdown [3]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

This section includes procedures to do the following:

- Start both the CRAY IOS-V and CRAY J90 mainframe or start both the CRAY EL IOS and CRAY EL mainframe and bring UNICOS up to a multiuser run state mode (startup; also called booting).
- Bring UNICOS back to single-user mode (shutdown).

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

To start and stop UNICOS system daemons, see section 4, page 49.

Related startup and shutdown documentation

3.1

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

- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `bcheckrc`(8), `brc`(8), `rc`(8), `dmdstop`(8), `fuser`(8), `init`(8), `msgdstop`(8), `sdaemon`(8), and `shutdown`(8) man pages
- *CRAY IOS-V Commands Reference Manual,* publication SR–2170
- *CRAY IOS-V Messages,* publication SQ–2172

- *UNICOS Installation Guide for the CRAY J90 Series,* publication SG–5271
- *CRAY EL Series IOS Commands Reference Manual,* publication SR–2408
- *CRAY EL Series IOS Messages*, publication SQ–2402
- *UNICOS System Installation Guide for the CRAY EL Series,* publication SG–5201

Procedure: Starting up the system

Note

A CRAY J90 IOS-V is case sensitive; enter all lowercase characters on the system console. A CRAY EL IOS system converts all characters to upper case; enter either uppercase characters or lowercase characters on an EL IOS console.

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

1. To start the IOS by loading the appropriate device strategies and drivers and by loading and executing the IOS kernel, enter the `load` command at the IOS boot prompt, which is `BOOT[`*snxxxx*`-ios`*x*`]>` on CRAY J90 systems (*snxxxx* is the mainframe serial number, and `ios`*x* is the IOS number) and `BOOT>` on CRAY EL systems.

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

   The IOS `load` command produces output on your terminal and returns the IOS prompt when complete, as follows (for CRAY EL systems, the `IOS>` prompt will be returned):

```
snxxxx-iosx>
```

Note

When using the system console, CONTROL-a toggles between the IOS and UNICOS prompts. When going from the UNICOS prompt, the prompt changes to the IOS prompt after you press CONTROL-a. When going from the IOS prompt, the prompt does not change until you press RETURN.

2. Step two is optional for CRAY J90 IOS-V systems because the date and time are obtained from the system console at IOS load time and are updated at 3:00 a.m. each morning.

   Enter the following command to report the time that will be used when UNICOS is booted:

```
IOS> time
```

Be sure the time is correct for your location to avoid changing it later for UNICOS, which would cause the UNICOS and IOS clocks to be different. To change the time on IOS0, enter the following (*dd/mm/yy* is the day, month, and year, and *hh:mm:ss* is the hour, minute, and second, respectively):

```
IOS> time dd/mm/yy hh:mm:ss
```

To change the time on any slave IOS on your system, execute the following command once for each slave IOS, substituting the number of the IOS (1, 2, 3, and so on) for the *n* symbol:

```
IOS> rcmd iosn time dd/mm/yy hh:mm:ss
```

Important

If you want to reload the IOS, type **reload** at the IOS prompt, and then start UNICOS by following step 3.

3. To start UNICOS, enter the `/bin/boot` command at the IOS prompt:

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

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

```
sn1234#
```

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

After booting UNICOS to single-user mode, you should run the `gencat` command to check the file systems for inconsistencies. Enter the following commands:

```
snxxxx-iosx> <CONTROL-a><RETURN> (toggles to the UNICOS console)
# /etc/gencat
```

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

When you are in single-user mode with only the `root (/)` file system available, you must do all editing by using the `ed` editor, because the `vi` editor is located in the `/usr` file system. If you want to use the `vi` editor before going to multiuser mode (step 4), you must mount the `/usr` file system (the `vi` editor is located in the `/usr` file system). Before going to multiuser mode, if you intend to work in single-user mode, it is advisable to check the `root (/)` file system by using `fsck`; see the "File Systems" section of this guide for the procedure.

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

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

If you are using the WYSE terminal, the solution is to type the following. Use the DELETE key to backspace.

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

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

If you are using the CRAY J90 IOS-V system console, the solution is to type the following:

```
:wq
# resize
# TERM=sun (or sun-cmd, if you are using the command tool)
# export TERM
# echo $TERM
# echo $SHELL
```

Note

Before going to multiuser mode, or if you intend to work in single-user mode, it is advisable to run `fsck` on the `root (/)` file system (see subsection 3.4.4, page 37, for more information).

4. To bring the system to multiuser mode, you must signal the `/etc/init` process to change to a new run level by entering the following command:

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

   Multiuser mode is usually run level 2. Although you can configure a system to run in multiuser mode at any level between 0 and 6, you may want to reserve some states for the future. For additional information about run-level configuration, see subsections 3.4.1, page 32, and subsection 3.5, page 41.

   As the system boots into multiuser mode, output is produced on your terminal. The `Creating new DD devices` message eventually will appear. This message indicates that the system is moving into multiuser mode properly. You will be prompted for the system date, which is an optional entry. When the system boot is complete, you see the following prompt:

```
Console Login:
```

For a sample of what appears on your console screen during the start-up procedure, see your UNICOS installation documentation.

Procedure: Shutting down UNICOS and the IOS

1. You must be logged in as `root` and be in the `root` (`/`), `/etc`, or `/ce` directory; to change to the `root` (`/`) directory, enter the following command:

```
# cd /
```

2. The `/etc/shutdown` script is designed to return the UNICOS system to single-user run state in a clean, orderly manner. The `/etc/shutdown` script prompts you for a message that will be sent to all users; if you want to include a message, use the `wall`(8) command to provide the message (see section 8, page 211). Before executing `/etc/shutdown`, you can use the `ps -eaf` command to see processes that are running, and the `who -u` command to see whether people are actively using the system. You may want to send active users a special message about when the system will be shut down. The `shutdown` command uses the following format:

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

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

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

The time it takes for the shutdown to complete depends on the number of processes that must terminate and file systems that must be unmounted; however, the shutdown process usually takes no more than 2 to 3 minutes.

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

```
Message: INIT: SINGLE-USER MODE.# /bin/sync
# /bin/sync
# /bin/sync
# /etc/ldsync (if you are using ldcache)
# df (to verify that all file systems have been unmounted cleanly)
```

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

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

```
# <CONTROL-a>
snxxxx-iosx> mc
```

Note

A CRAY J90 IOS-V is case sensitive; enter all lowercase characters on the system console. A CRAY EL IOS system converts all characters to upper case; enter either uppercase characters or lowercase characters on an EL IOS console.

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

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

5. Optional step. Power off your system if you choose to do so (see appendix I, page 463, or see your hardware installation manual for procedures to power off your system).

Shutdown information

3.2

The `/etc/shutdown` script terminates all user processes and system daemons, releases all logical device cache, and unmounts all UNICOS file systems (except for `root`). Unlike the `/etc/rc` start-up script, the operation of the `/etc/shutdown` script is not altered by any UNICOS control files. For CRAY J90 and CRAY EL systems, you do not have to modify the `/etc/shutdown` script directly.

If you install and start any local processes or daemons during the execution of the `/etc/rc` script, to stop them within the `/etc/shutdown` script, you can add control information into the `/etc/config/daemons` file and insert `/etc/sdaemon` commands into the `/etc/shutdown` script. You should make any changes either in `/etc/config/rcoptions` or in one of the user exits; **do not** modify `/etc/rc`. For information on starting and stopping UNICOS system daemons, see section 4, page 49.

The process of a UNICOS system shutdown is as follows:

1. The `/etc/shutdown` script announces that UNICOS is being stopped and processes killed at the end of a specified grace period. A system administrator can set this grace period by passing an argument to `/etc/shutdown`; otherwise, the grace period defaults to 60 seconds.
2. All running processes are terminated using the `/etc/killall` script; the following signals are sent during the execution of the script:

   `SIGSHUTDN` (signal 27)

   `SIGHUP` (signal 1)

   `SIGKILL` (signal 9, the default; kill with extreme malice)
3. The `sync`(1) utility flushes all previously unwritten system buffers to disk; the `ldsync`(8) utility flushes all logical device cache.
4. UNICOS system accounting, NQS, and other daemons are shut down.
5. All logical device cache is released.
6. All file systems (except the `root` file system) are unmounted in the reverse order from which they were mounted.
7. UNICOS is now in single-user mode (UNICOS is still running after shutdown).

8. At this point, UNICOS is in a state in which a reboot should not corrupt any files. (To reboot the system, toggle to the IOS console and follow the procedure on page 23.)
9. If you want to stop UNICOS from running, toggle to the IOS console and enter an `mc` (Master Clear) command to the mainframe (follow the procedure on page 27).
10. At this point, UNICOS has been shut down (or halted following step 8) but the IOS is still executing. If you want to stop the IOS software from running and revert to running the PROM program, enter the `reset` command. This step puts the system as close as possible to the state it is in after being powered up.

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

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

Note

A CRAY J90 IOS-V is case sensitive, so the following file names must be entered in all lowercase characters on the system console.

A CRAY EL IOS converts all characters to upper case, so the following file names may be entered in either upper case or lower case on an EL IOS console.

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

File	Description
`/autoboot`	If the file exists, the IOS automatically attempts to load itself and boot UNICOS after any reset or power cycle. This file may contain the absolute path to an alternate IOS kernel to be loaded; otherwise, `/ios/ios` will be loaded followed by `/bin/boot`.
`/bin/boot`	UNICOS boot script.

File	Description
`/config`	IOS-V and EL IOS configuration file.
`/sys/param`	Default IOS-parameter file that contains configuration specification language (CSL) statements defining physical, logical, and system disk devices.
`/sys/unicos.ymp`	Default UNICOS kernel.

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

File	Description
`/etc/config/daemons`	File listing and daemons to be started during multiuser startup; used by `/etc/sdaemon`. See section 4, page 49.
`/etc/config/rcoptions`	Sets environment variables that control `/etc/rc`.
`/etc/inittab`	Read by `/etc/init` at system boot.

The following are UNICOS shell scripts:

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

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

File	Description
`/CONFIGURATION`	Contains processed CSL definitions of disk devices and special system devices. `/CONFIGURATION` has the slice and logical device size comments corrected, but it loses the early CSL directives in `/IOS-param`.
`/IOS-param`	A copy of the `/sys/param` file.
`/unicos`	A copy of the running UNICOS kernel. This file is **not** an exact copy of the bootable image that resides on the IOS disk in `/sys/unicos.ymp`.

Start-up scripts
3.4

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

The `/etc/init` *command*
3.4.1

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

At any moment, `init` considers the system to be in one of eight different run levels: 0 through 6, or S (s) (a run level of `S` or `s` refers to single-user mode). When you specify S, `init` operates in single-user mode with the additional result that `/dev/syscon` is linked to the user's terminal line, thus making it the virtual system console). For more information about run-level configuration, see subsection 3.5, page 41.

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

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

The `/etc/inittab` *file* 3.4.2

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

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

Field	Description
`ID`	A label to identify the entry. The label can consist of a maximum of four characters and must uniquely identify the entry.
`run state`	Run level in which an entry should be processed. A null entry (two colons) indicates that the entry must be executed when changing to any numbered (0 through 6) run state. Numbered run states signify varying levels of UNICOS functionality and rely on the `/etc/init` process starting or stopping system processes as required. See the `init`(8) man page. The run state field is the "when" portion for the entry.
`action`	The action field specifies "how" to start the command or program specified in the process field for this entry.
`process`	The command or the name of the program to execute. This action field is the "what" portion of the entry. You can insert comments into this field by prefixing a line with a # symbol.

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

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

Value	Description
`wait`	Starts process and waits for its termination before proceeding.

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

- The initial run level (specified by an entry with the action `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. By convention, the `bcheckrc` (see `brc`(8)) program is called by an entry with the action `bootwait` to perform boot-time-only actions, and the `rc` (see `brc`(8)) program is called by an entry with the action `wait` to perform 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 following command (see `consoled`(8)) to allow logins on the system console:

  ```
  /etc/consoled
  ```

- Any run levels that accept logins from users on front-end systems need an entry with an action of `generic`. This entry instructs `init`(8) to accept login requests from daemons through the `/etc/initreq` FIFO special file (named pipe). This is true even when the run level is intended for use by just one dedicated user; restricting access to the system is accomplished by the `rc`(8) script (see subsection 3.4.6, page 39), not by limiting logins to specific devices, as is often done on traditional UNIX systems.

A sample `/etc/inittab` file follows:

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

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

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

Note

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

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

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

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

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

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

`/etc/bcheckrc` *script*
3.4.4

The `/etc/bcheckrc` script is one of the start-up scripts that `/etc/init` invokes when it reads through the `/etc/inittab` file.

The `/etc/bcheckrc` script performs two major activities. It resets the system date, if necessary, and checks all file systems that will be mounted during the start-up process.

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

Note

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

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

Caution

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

`/etc/brc` *script* 3.4.5

The `/etc/brc` script is invoked by `/etc/init` through `/etc/inittab` to perform the following tasks:

- Conditionally remove and re-create the system mount table (`/etc/mnttab`).

  The system mount table records which file systems are mounted. When you execute commands, such as `/etc/mount`(8), the contents of `/etc/mnttab` are displayed. When you execute the `/bin/df`(1) command to check the level of free disk space, the `/etc/mnttab` file is used. All mounted file systems have one entry recorded into the file. When you unmount a file system, its entry is removed from `/etc/mnttab`.

  Do not modify the `/etc/bcheckrc` or `/etc/brc` scripts to start processes. If the system crashes without allowing for the orderly unmounting of file systems, the `/etc/mnttab` file will no longer accurately reflect the state of your mounted file systems or correctly show what files must be mounted after rebooting the system. After `/etc/brc` resets the contents of `/etc/mnttab` (the system mount table), future `/etc/mount` commands will work properly.

- Copy system dumps into a separate file system by executing the `/etc/coredd`(8) script.

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

The multiuser start-up script `/etc/rc` 3.4.6

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

Note

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

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

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

For more information about the `/etc/rc` script, see *UNICOS System Administration,* publication SG–2113.

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

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

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

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

You may use `rcoptions` to set the device name for the `/usr`, `/usr/tmp`, and `/tmp` file systems; the path name of the `/etc/rc` log file; specify whether to `mkfs /tmp` or `/usr/tmp` at boot time; mount `user` file system; whether to activate `ldcache`; and whether to start accounting, `sadc`, or network.

For additional information, see *UNICOS System Administration*, publication SG–2113.

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

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

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

Run-level configuration
3.5

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

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

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

Changing run level
3.5.1

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

```
/etc/init level
```

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

Strategies for using run levels
3.5.2

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

All systems have a single-user mode (for system work that must be performed unencumbered by the presence of other users on the system) and at least one multiuser mode. If the system is restricted at various times to dedicated use by one or more users,

you should devote one or more run levels to initializing the system for this dedicated use. In all cases except for single-user mode (which requires little or no initialization), initialization is performed by the `rc` (see the `brc`(8) man page) script.

Single-user mode
3.5.2.1

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

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

Typically, the system is brought into single-user mode either following a system boot or by `shutdown`(8). 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.

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

Multiuser mode
3.5.2.2

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

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

Typical tasks you can perform while in multiuser mode
3.5.2.3

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

- Checking which file systems are mounted by using the `/etc/mount` command (see section 5, page 57).
- Using the `/bin/df` command or the `/etc/fsmon` file system monitor to check all mounted file systems to ensure that no mounted file system consumes all available free disk blocks.
- Checking the number of system users using the `who` command. To identify idle users, enter `who -u`. To determine the number of users, enter `who | wc -l`. To generate the number of users and a list of their names, enter `who -q`.
- Informing users of system changes by using `/etc/wall` (see section 8, page 211).
- Monitoring how your UNICOS system is running by using the `/usr/bin/sar` utility. The `/usr/bin/sar`(1) utility has many options used to gain information about disk performance, character list buffers, CPU performance, and IOS throughput. The most useful options for a system administrator include `-d` (disk), `-x` (IOS), and `-v` (critical internal system table sizes). For more information, see the `/usr/bin/sar`(1) man page.
- Checking all running processes by using the `ps`(1) command to determine whether a process is using an abnormally large amount of CPU time. The `-eaf` options generate a full listing for all running processes.
- Checking the contents and size of your UNICOS error logs. Usually, error logs are found in the `/usr/adm` directory. Also, ensure that the error logging daemon is executing and that IOS disk errors are being logged into the `/usr/adm/errfile` file. For log information, see section 9, page 219. For details on disk error reporting, see the `/etc/errpt`(8) man page.
- Checking `mail` by using the `/bin/mail` command while logged on to `root`, or the login that receives requests to restore files. If a problem occurs, the system itself sometimes sends mail to `root`.

Dedicated system
3.5.2.4

It is sometimes necessary to provide dedicated system time so that a particularly large or time-critical job can run unencumbered by other user processes. There also will be times at which system development work requires that the system be brought up as though it were running in multiuser mode, when access to the machine is actually restricted to the system staff. To lock out all users except yourself, use `/etc/udbrstrict -r -L` *YOUR_USERID*. Do not use just `/etc/udbrstrict -r`, because this limits logins to only root, which can then be done only on the console device. For more information about the UDB `ue_permbits` field, see *UNICOS System Administration*, publication SG–2113.

IOS prompts, and permissible actions
3.6

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

Note

When using the CRAY J90 or CRAY EL IOS master console, CONTROL-a toggles between the IOS and UNICOS prompts.

For CRAY J90 systems, when going from the UNICOS prompt, after you press CONTROL-a, the prompt changes to sn*xxxx*-ios*x*>. For CRAY EL systems, when going from the UNICOS prompt, after you press CONTROL-a, the prompt changes to IOS>.

When going from the IOS prompt, the UNICOS prompt is not displayed until you press RETURN.

IOS boot prompt 3.6.1

The IOS boot prompt is as follows:

`BOOT[sn`*xxxx*`-ios`*x*`]>` (on CRAY J90 systems)

`BOOT>` (on CRAY EL systems)

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

- The IOS is down; it is running in PROM; no strategies or drivers are loaded.
- The CRAY J90 or CRAY EL mainframe is down; UNICOS is not running.

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

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

- On CRAY EL systems, unmount, mount, or format the IOS SCSI disk by using `mount`, `umount`, or `dformat`. For CRAY J90 systems, this action is not necessary because the IOS file system resides on the SPARCstation system console.
- Take an IOS dump (not a UNICOS dump) by typing `iosdump`.
- By using the `tar` command, transfer files between the CRAY J90 system console disk or the CRAY EL IOS disk and the IOS DAT (rpd03) tape drive (or the QIC (rpq01) tape drive on CRAY EL systems).
- Load the IOS kernel, strategies, and drivers into memory by typing `load`. This also starts the execution of all IOSs defined in the `/config` file (`/CONFIG` file on CRAY EL systems).
  - The IOS kernel resides on the CRAY J90 system console disk or the CRAY EL IOS disk and has the path name `/ios/ios`.
  - The IOS strategies and drivers reside on the CRAY J90 system console disk or the CRAY EL IOS disk in the `/dev` directory.
  - The IOS `load` command uses the IOS configuration file `/config` (`/CONFIG` file on CRAY EL systems) to determine which strategies and drivers to load into IOS memory.

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

For CRAY J90 systems:

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

For CRAY EL systems:

```
BOOT> load
```

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

IOS prompt 3.6.2

The IOS prompt is as follows:

snxxxx-iosx> (on CRAY J90 systems)

IOS> (on CRAY EL systems)

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

- The IOS is up; it is running the IOS kernel, strategies, and drivers. Any slave IOP in the IOS may or may not be running. Check the `/adm/syslog` file on the CRAY J90 system console disk or the CRAY EL IOS disk for messages indicating that a slave IOS has panicked if this is suspected.
- The IOS and CRAY J90 or CRAY EL mainframe are both up; CONTROL-a was pressed to change from the mainframe prompt to the IOS prompt.
- The CRAY J90 or CRAY EL mainframe is down; UNICOS is not running. A mainframe system panic has occurred. The IOS is still running, however.

If a mainframe system panic occurs, the IOS may still be running, but it will be in an undefined state. Taking an IOS dump at this point may be helpful; use the `iosdump` command. See the *CRAY IOS-V Messages,* publication SQ–2172, or the *CRAY EL Series IOS Messages*, publication SQ–2402.

From this state, you can perform the following actions:

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

UNICOS System Daemons [4]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

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

Related UNICOS system daemons documentation

4.1

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

Procedure: Starting and stopping UNICOS system daemons

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

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

Note

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

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

```
Configure System
...System Daemons Configuration
......System Daemons Table
```

System Daemons Table

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

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

```
                        System Daemons Table

S-> Group                                          NFS
    Name                                           exportfs
    Start up at boot time?                         YES
    Kill action                                    *
    Executable pathname                            /etc/exportfs
    Command-line arguments                         -av
    Additional command-line arguments
    Additional command-line arguments
```

If you are not using the menu system, edit the `/etc/config/daemons` configuration file to set which daemons to start or stop. You can modify this file by using your preferred UNICOS editor (for a sample `/etc/config/daemons` file, see page 55).

Note

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

To Start One Daemon

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

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

```
/etc/sdaemon -s daemon
```

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

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

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

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

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

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

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

```
/etc/sdaemon -k daemon
```

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

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

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

Note

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

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

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

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

File Systems [5]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

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

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

No single configuration of available disk drives into file systems and logical devices will prove best for all purposes. Optimizing file system layout is usually an iterative process; make your best attempt, then run it for awhile and monitor it, for disk use monitoring information (see subsection 5.3, page 63). You will make adjustments to your configuration based on information you gather about your users' needs. As the needs of your users change, you will reconfigure your file systems to have a well-balanced configuration. In the absence of a set of absolute rules, the facts and guidelines presented in this section will prove useful when you decide on a file system plan for your system.

Note

Although all UNICOS file systems have some common aspects, file system creation and organization varies between file systems on CRAY J90 and CRAY EL systems and on some CRAY Y-MP systems. If you also have other Cray Research systems that are not CRAY J90 or CRAY EL systems, see *UNICOS System Administration*, publication SG–2113, to determine differences in file systems and how to configure them.

Related file systems documentation 5.1

The following Cray Research publications contain information discussed in this section:

- *UNICOS System Administration*, publication SG–2113, sections on File Systems and Special Files and Configuring UNICOS
- *UNICOS Installation Guide for the CRAY J90 Series*, publication SG–5271
- *UNICOS Installation Guide for the CRAY EL Series*, publication SG–5201
- *UNICOS User Commands Reference Manual*, publication SR–2011: `df`(1), `du`(1), `mkdir`(1), and `rm`(1) man pages
- *UNICOS File Formats and Special Files Manual*, publication SR–2014: `dir`(5), `dsk`(4), `fs`(5), `fstab`(5), `inode`(5), `ldd`(4), `mnttab`(5), and `pdd`(4), man pages
- *UNICOS Administrator's Commands Reference Manual*, publication SR–2022: `ddstat`(8), `disk`(8), `diskusg`(8), `dmap`(8), `econfig`(8), `fsck`(8), `fsmap`(8), `fuser`(8), `labelit`(8), `mkfs`(8), `mknod`(8), `mount`(8), `stor`(8), and `umount`(8) man pages

Note

Where UNICOS publications provide information for Cray Research systems that have I/O subsystem model E (IOS-E), this information also applies to CRAY J90 and CRAY EL systems, unless noted otherwise.

An overview of file systems
5.2

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

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

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

Note

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

Terminology 5.2.1

This subsection provides terminology associated with file systems. Everything is viewed by UNICOS as a file, whether it is an ASCII file of user data or a physical disk device. UNICOS supports five types of files: regular, directory, block special (such as a disk drive), character special (such as a tape drive), and FIFO special. *Regular files* hold user data of various formats. *Directory files* contain the names of "regular" files and other directories, along with their corresponding inode numbers. When block or character special files are accessed, device drivers are invoked that communicate with peripheral devices, such as terminals, printers, and disk drives. FIFO special files, also called *named pipes*, allow unrelated programs to exchange information.

A *physical device* is a tape or disk device. Physical disk devices are read from and written to in units of 512-word (4096-byte) blocks. The smallest unit of I/O disk devices can perform is one block. UNICOS file systems are defined in regions of contiguous blocks called *slices*. File systems can be built on many different slices.

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

One or more slices create a *logical device*. Although a logical device appears to be one device, its slices can be located across several physical devices. Logical devices become file systems when the disk is initialized with a file system structure by using the `/etc/mkfs` command.

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

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

Special files are not composed of readable data. Instead, they serve as a connection between a path name (such as `/dev/dsk/root`) and the device handling routines in the UNICOS kernel to control I/O to the device. Block special files are used to communicate with file systems.

- *Block special files*: The drivers for these files process data in blocks. Block devices have a minimum transfer unit size of one block (4096 bytes or 512 words). All I/O for CRAY J90 and CRAY EL file systems use block special files. You can address block special files and their related devices by using various I/O techniques.

- *Character special files*: The drivers for these files process "raw" data, bypassing UNICOS kernel buffering. Data is transferred directly between the user's memory area and the physical device. UNICOS character special files are used to support tape and tty connections, among others. You can use character special files and their related devices only for sequential I/O.

All special files have a major and a minor device number associated with it. A *major device number* refers to the type of device. Major device numbers are used as an index into a table of device drivers appropriate for that kind of physical device. These routines open, close, read, write, and control a physical device. A *minor device number* is used by the appropriate driver (determined by the major number) to specify a particular logical disk device, tape drive, or physical device. Minor device numbers range from 0 to 255; however, numbers 250 through 255 are reserved for use by the operating system. For example, minor number 253 is used for the `ce` partition. For additional information, see *UNICOS System Administration*, publication SG–2113.

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

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

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

File	Description
`/dev`	Directory of special files and subdirectories of other special files.
`/dev/dsk/`	Directory that contains all block special files that represent logical disk devices for current file system configuration. The major device number is always 0 for disk devices. A `b` in the directory permissions field (`ls -l` output) indicates a block special file.

UNICOS file system structure 5.2.2

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

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

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

Element	Description
Super block:	Used to store file system size and the number of inodes in the file system, as well as internal parameters such as allocation strategy. It is updated when the `mkfs` or `setfs` command is run. Several copies of the super block are kept for robustness (redundant copies make it easier to recover information if a catastrophic failure occurs). The super block is read into memory when the file system is mounted, and it is flushed to disk when it is modified or when the file system is unmounted.
Inode region:	Each file in a mounted file system is identified with a unique pointer called an inode number. The *inode* itself contains file information such as permissions, file size, whether the file is a directory, and so on. The inode region contains a maximum of 32,768 inodes. You can have a maximum of 4 inode regions per partition.
Dynamic block:	A block that contains the file system information that changes during system operation. The dynamic block contains block counts for a specific partition. This information is flushed to disk when the file system is modified, when the file system is unmounted, or when `sync`(2) is executed.

Element	Description
`setfs`(8) command:	Changes dynamic information in the file system super block without remaking the file system.
Block allocation bit map:	Controls block allocation across the entire file system.
Map blocks:	A bit map of the disk sectors.
Partition data blocks:	The disk area for directories and user data.

Commands for examining files and file systems

5.3

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

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

Command	Description
`/usr/lib/acct/diskusg`	Summarizes the disk usage on the file system you specify by file ownership and identifies users who are using most of the space on a file system. The `/usr/lib/acct/diskusg -h` command is the preferred command for summarizing disk use; the `-h` option provides headings.
`/etc/dmap`	Displays information about the configuration of a disk subsystem.
`/etc/bmap`	Displays which file is using the block on a given file system.
`/etc/fsmap`	Displays file system free block layout.
`/bin/df`	Displays the number and percentage of free blocks available for mounted file systems; the `-p` option is particularly useful.

Command	Description
`/bin/du`	Summarizes the disk usage on a file system, by directory structure. The `-s` option provides the total number of disk blocks used under each directory (or file) specified.
`/etc/errpt`	Processes errors report generated by `errdaemon`. This UNICOS command is for disk hardware errors; `errpt -a` produces a detailed list of errors; `errpt -d` *device-type* produces list of errors for the specified device type.
`/etc/mount`	Displays the list of all currently mounted disk files and their mount points when issued without arguments.
`/ce/bin/olhpa`	Displays hardware errors by reading the `/usr/adm/errfile` file. The `-d` option lets you view disk errors.

File system planning
5.4

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

The first step a UNICOS system administrator must perform is to plan which slices of a physical device will be used to make up each file system, as well as which file systems should be striped, if any, and which should be banded. (For information about disk striping, see subsection 5.6, page 68, and for information about disk banding, see subsection 5.7, page 68.) You must consider disk capacity and transfer rate, as well as file system size and usage, along with the number of users and types of applications your Cray Research representative installed on your system.

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

Note

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

For minimum file system and amount of free blocks needed to install other Cray Research software products, see the *UNICOS Installation Guide for the CRAY J90 Series,* publication SG–5271, or the *UNICOS Installation Guide for the CRAY EL Series Systems*, publication SG–5201.

The `root` *(/) file system* 5.4.1

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

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

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

The `/usr` *file system* 5.4.2

Size recommendations: You should define a **minimum** region of 190,000 blocks. The contents of the `/usr/adm` subdirectory tend to grow very large because the UNICOS accounting data is kept here.

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

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

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

The `/usr/src` *file system* 5.4.3

Size recommendations: The recommended minimum default value is 50,000 blocks. This size is sufficient to hold all of the files necessary to relink the UNICOS kernel. Also, you must allow enough space in your default value to handle additional Cray Research asynchronous products you will load and use (for this information see your UNICOS installation guide and related errata). This file system is best handled by allocating slices from several different disks to compose the logical file system. This disk allocation strategy is called *banding*.

The `/tmp` *file system* 5.4.4

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

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

The `swap` *device* 5.4.5

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

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

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

If your system's job mix swaps frequently, you may want to configure your swap device as a striped device. If possible, stripe `swap` across two DD-4, DD-5I, or DD-5S disks or across disks attached to separate ESDI controllers. For more information about striping see subsection 5.6 and *UNICOS System Administration*, publication SG–2113.

The dump *device*
5.4.6

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

Caution

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

The back-up root *(/) and back-up* /usr *file systems*
5.4.7

Size recommendations: The backup root (/) (traditionally called bk_root) and back-up /usr (traditionally called bk_usr) file systems are equal in size to the original root(/) and /usr file systems.

Device recommendations: Keep bk_root and /usr file systems on different disk drives, controllers, and IOSs if possible from root (/) and bk_usr file systems. You also should keep the back-up version of a file system on a different drive (and controller and IOS if possible) from your original file system.

The /home *file system*
5.4.8

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

Disk device characteristics
5.5

You may define and mount one or more file systems on disk devices. For a table of disk device characteristics see appendix E, page 437.

Disk striping
5.6

A striped device can be made up of two or more of the same type of disk drives or can be logically the same type. The number of blocks must be the same in each slice. Several drives are combined together in one logical unit (known by the name of the first slice name), which makes simultaneous I/O operations possible. Slice members of a stripe group must be previously defined in the physical device statement of the configuration specification language (CSL). In addition to physical CSL definition statements in the IOS `/sys/param` file, a special stripe device definition statement also is required to configure a stripe group. For information about using CSL, see subsection 5.8.

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

Note

You should not ldcache a swap file.

Disk banding
5.7

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

Configuring your devices and their file system allocation
5.8

The system configuration file that configures disks is the `/sys/param` file on the IOS disk drive. The configuration specification language (CSL) is used to define the configuration and parameter settings that are used at boot time. CSL defines the following:

- Number of IOSs
- Mapping IOS channels to specific CPUs

- Physical device attributes and slice layout
- Logical grouping of physical disk slices
- System-defined devices
- Network configuration

Note

If you use the menu system to configure these settings, it will automatically generate the CSL statements that describe your system configuration. The following subsections include the name of the menu on which the parameters described are set.

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

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

CSL syntax
5.8.1

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

- An identifier is a sequence of digits and letters that specify either special keywords (such as `CONFIG`) or specific objects (such as a physical device). You can enclose the digits and characters in double quotation marks. The underscore (_) and dash (–) are interpreted as letters. Identifiers consist of letters, numbers, and the – and _ symbols. CSL identifiers are case sensitive. The first character may be any of the valid identifier characters.

– Reserved disk type identifiers in CSL (descriptive comments are within brackets). DD3, DDESDI, RD1, and DD4 are not supported disk type identifiers on CRAY J90 systems:

DD3 (new ESDI)	DD4 (IPI-2 Sabre-7)
DDESDI (old ESDI)	RD1 (removable DD-3 ESDI)
DDRAM (RAM device)	DD5I (buffered IPI)
DDSTR (striped device)	DD5S (SCSI IPI)
HD16 (HIPPI disk, 16-Kbyte sector); deferred implementation for CRAY J90 systems	HD32 (HIPPI disk, 32-Kbyte sector); deferred implementation for CRAY J90 systems
HD64 (HIPPI disk, 64-Kbyte sector); deferred implementation for CRAY J90 systems	

– For a list of additional keyword identifiers that have special meaning, see "Configuring UNICOS," in *System Administration*, publication SG–2113.

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

Placement of CSL statements
5.8.2

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

- All CSL statements must be terminated by a semicolon, and all section definitions must be placed within one pair of braces { } or ("curly brackets").
- UNICOS processes CSL statements in order of appearance in the IOS parameter file.

- Your IOS parameter file must begin with the keywords `revision sn`*xxxx*; *xxxx* is your machine's serial number.
- The first section (ios_e section) of CSL statements defines IOS information (as of the UNICOS 8.0 release, reference to ios_e in the configuration file includes CRAY J90 IOS-Vs and CRAY EL IOSs).
- The second section of CSL statements defines the mainframe information.
- The third section of CSL statements defines the UNICOS configuration information.
- The fourth section of CSL statements defines the file systems information, which includes configuration statements for the physical devices, logical devices, and special system devices.
- The fifth section of CSL statements defines the network configuration for your system.
- The file must end with the closing brace.

The following subsections describe the parameter file sections; a sample file is included beginning on page 78.

Revision section
5.8.2.1

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

ios_e section
5.8.2.2

This statement section sets the number of I/O subsystems configured. You specify the characteristics on the `Configure System ==> IOS Configuration` submenu.

This section should include the following:

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

Statement	Description
`eiop 0`	Required keywords for the cluster
`eiop` *xx*	Designates the controller numbers for the disks associated with the specific IOS.

Mainframe section
5.8.2.3

The mainframe section defines the number of CPUs, size of the mainframe memory, and channel information. You specify the characteristics on the `Configure System ==> Mainframe Hardware Characteristics` submenu.

This section should include the following:

Statement	Description
value `cpus`	The number of CPUs.
value units `memory`	The *units* may be either words or Mwords; *value* is typically set to the physical amount of memory in the machine.
`channel` *value*	The *value* of the master IOS is always 20 or 020 (octal). The channel numbers of the slave IOSs depend on the CPU to which they are connected. IOS0 will always be on CPU0, channel 020. The other possible channels for slave IOSs on CPU0 are 024 and 026. CPU1 can have IOSs on channels 040, 044, and 046. CPU2 can have IOSs on channels 060, 064, and 066. CPU3 can have IOSs on channels 0100, 0104, and 0106. To define the channels that connect to the slave IOSs, use the `channel` keyword. Additional CPUs beyond CPU3 are on daughter CPU boards and cannot have IOSs attached directly to them.

UNICOS section
5.8.2.4

The UNICOS section sets certain tunable parameters in the `var` structure. You set these parameters in the `Configure System ==> UNICOS Kernel Configuration` submenu. Because this is an advanced topic, you should read *UNICOS System Administration*, publication SG–2113, to determine what parameters you might want to change.

File system section
5.8.2.5

The file system section includes the following:

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

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

Physical device definition
5.8.2.5.1

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

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

You must define each field:

Field	Description
`disk` *name*	The *name* of each device is site-configurable, but it must be unique among all devices. By convention, the format is composed of the disk type, IOS number, controller number, and unit (for example, `S_0200`).
`type` *type*	Defines the disk type (see Table 1).
`cluster` *number*	Defines the IOS. Specifies the IOS number; no default.
`eiop` *number*	Defines the controller (see Table 1).
`channel` *number*	Defines the channel number.

Field	Description
`unit` *number*	Defines the unit attached to the controller (see Table 1).
`pdd` *slice*	Each *slice* name also is site-configurable, but it must be unique among all devices. Use a meaningful naming scheme for your file system slices. You may want to name the slices with a combination of a reference to the file system of which they are a part, and a unique number. Names of slices must be unique and consist of 8 characters or fewer.
`minor` *number*	Must be unique; you should use numbers in ascending order.
`sector` *measure*	For all disks used with CRAY J90 or CRAY EL systems, *measure* is the starting block number of the slice.
`length` *number units*	For all disks used with CRAY J90 or CRAY EL systems, *number units* is the number of blocks for the slice.

Table 1. Disk device types and their values

Disk type keyword	`eiop` value range	Total blocks per disk unit	Maximum number of units
DDESDI	0 – 7	170,100	4 (0 – 3)
DDLDAS	8	1,269,114	–
DD3	0–7	334,200	4 (0 – 3)
DDAS2	8	2,502,000	–
DD4	10–17	653,000	2 (0 or 1)
DD5S	20–27	781,000	4 (0 – 3)
DD5I	30–37	723,000	4 (0 – 3)
RD1	0–7	334,200	–
DDRAM			

Note

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

Logical device definition
5.8.2.5.2

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

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

The fields are defined as follows:

Field	Description
`ldd` *name*	Consists of up to eight characters. By convention, the name is lowercase and reflects the name of the special file that will be created automatically during UNICOS multiuser startup. Each `ldd` *name* shows up as a file name in the UNICOS `/dev/dsk` directory. The `ldd` *name* portion must be unique for each logical device.
`minor` *number*	Must be unique; you should use numbers in ascending order.
device slice	The previously defined physical slice name that describes this logical device.

Special system devices
5.8.2.5.3

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

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

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

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

Network section 5.8.2.5.4

The network section defines network devices and network parameters. You can configure low- and high-speed network communication devices in the `Configure System ==> UNICOS Kernel Configuration ==> Communication Channel Configuration` menu. The network section includes the following information:

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

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

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

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

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

Checking your disk configuration parameter file
5.9

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

To verify the configuration, manually use the `/etc/econfig` command to check the syntax of CSL, using the following syntax:

```
# /etc/econfig your_parameter_file_name
```

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

A sample `/sys/param` configuration file follows:

```
/*
 * Update information for "ios_e" section:
 *
 * IOS INFORMATION:
 */
ios_e {
   /*
 * BEGIN SECTION:  IOS INFORMATION
   */
cluster 0 {
      muxiop; eiop 0; eiop 30;
    }
        cluster 1 {
                muxiop; eiop 0; eiop 20; eiop 30;
        }
cluster 2 {
      muxiop; eiop 0; eiop 20; eiop 30;
    }
   /*
 * END SECTION:  IOS INFORMATION
   */
}

/*
 * Update information for "mainframe" section:
 *
 * HARDWARE INFORMATION:
 */
mainframe {
   /*
 * BEGIN SECTION:  HARDWARE INFORMATION
   */
channel 020 is lowspeed to cluster 0;
        channel 040 is lowspeed to cluster 1;
channel 024 is lowspeed to cluster 2;
channel 0112 is lowspeed to pseudo TCP;
   /*
 * END SECTION:  HARDWARE INFORMATION
   */
}
/*
 * UNICOS configuration
 */
unicos {
   /*
```

```
 * BEGIN SECTION:  KERNEL PARAMETERS
 */
2048  NLDCH;    /* ldcache headers */
16384 LDCHCORE;            /* ldcache main memory allocation */
1024  NBUF;   /* system buffers */
117   PDDSLMAX;            /* maximum number of physical slices */
114   LDDMAX;   /* maximum number of logical devices */
100   PDDMAX;   /* maximum number of physical devices */
10    SDDSLMAX;            /* maximum number of striped devices */
8     MDDSLMAX;            /* maximum number of mirrored devices */
12    TAPE_MAX_CONF_UP;
65536 TAPE_MAX_PER_DEV;
   /*
 * END SECTION:  KERNEL PARAMETERS
   */
}

/*
 * Update information for "filesystem" section:
 *
 * DISK CONFIGURATION:
 *
 * SPECIAL SYSTEM DEVICES:
 */
filesystem {
   /*
 * BEGIN SECTION:  DISK CONFIGURATION
   */
/*
 *Physical device configuration
 */
disk B_000 {type DD5I; iopath{cluster 0;eiop 30;channel 020;} unit 0;
      pdd root_000 {minor 1; sector 0; length 90000 sectors;}
      pdd bkusr_000 {minor 2; sector 90000; length 150000 sectors;}
      pdd src_000 {minor 3; sector 240000; length 50000 sectors;}
      pdd dump_000 {minor 4;sector 290000;length 65536 sectors;}
      pdd core_000 {minor 5;sector 355536;length 90000 sectors;}
      pdd tmp_000 {minor 6; sector 445536; length 277463 sectors;}
    }
disk B_001 {type DD5I; iopath{cluster 0;eiop 30;channel 020;} unit 1;
      pdd bkroot_001 {minor 7; sector 0; length 90000 sectors;}
      pdd usr_001 {minor 8; sector 90000; length 150000 sectors;}
      pdd bksrc_001 {minor 9; sector 240000; length 50000 sectors;}
      pdd tmp_001 {minor 10; sector 290000; length 335112 sectors;}
      pdd bs_001 {minor 11; sector 625112; length 97187 sectors;}
    }
```

```
disk B_002 {type DD5I; iopath{cluster 0; eiop 30; channel 020;} unit 2;
      pdd swap_002 {minor 12;sector 0;length 55576 sectors;}
      pdd tmp_002 {minor 13; sector 55576; length 207464 sectors;}
      pdd bs_002 {minor 14; sector 263040; length 257424 sectors;}
    }
disk B_003 {type DD5I;iopath{cluster 0;eiop 30;channel 020;} unit 3;
      pdd swap_003 {minor 15; sector 0;length 653000 sectors;}
    }
disk S_100 {type DD5S; iopath{cluster 1; eiop 20; channel 040;} unit 0;
      pdd s_100 {minor 18; sector 0; length 781000 sectors;}
    }
disk S_101 {type DD5S; iopath{cluster 1; eiop 20; channel 040;} unit 1;
      pdd s_101 {minor 19; sector 0; length 781000 sectors;}
    }
disk B_100 {type DD5I; iopath{cluster 1; eiop 30; channel 040;} unit 0;
      pdd b_100 {minor 20; sector 0; length 720000 sectors;}
    }
disk B_101 {type DD5I; iopath{cluster 1; eiop 30; channel 040;} unit 1;
      pdd b_101 {minor 21; sector 0; length 720000 sectors;}
    }
disk S_200 {type DD5S; iopath{cluster 2; eiop 20; channel 024;} unit 0;
      pdd s_200 {minor 22; sector 0; length 781000 sectors;}
    }
disk S_201 {type DD5S; iopath{cluster 2; eiop 20; channel 024;} unit 1;
      pdd s_201 {minor 23; sector 0; length 781000 sectors;}
    }
disk B_200 {type DD5I; iopath{cluster 2; eiop 30; channel 024;} unit 0;
      pdd b_200 {minor 24; sector 0; length 720000 sectors;}
    }
disk B_201 {type DD5I; iopath{cluster 2; eiop 30; channel 024;} unit 1;
      pdd b_201 {minor 25; sector 0; length 720000 sectors;}
    }
/*
 *Logical device configuration
 */
ldd swap  { minor 1;
      pdd swap_002;
      pdd swap_003;
    }
ldd dump  { minor 2;
      pdd dump_000;
    }
ldd core  { minor 3;
      pdd core_000;
    }
```

```
ldd tmp   { minor 4;
      pdd tmp_000;
      pdd tmp_001;
      pdd tmp_002;
    }
ldd root  { minor 5;
      pdd root_000;
    }
ldd usr   { minor 6;
      pdd usr_000;
    }
ldd src   { minor 7;
      pdd src_000;
    }
ldd bkroot  { minor 8;
      pdd bkroot_000;
    }
ldd bkusr { minor 9;
      pdd bkusr_000;
    }
ldd bksrc {minor 10;
      pdd bksrc_000;
    }
/*
 *Stripe device configuration
 */

sdd s0 {minor 1;
      pdd s_100;
      pdd s_101;
      pdd s_200;
      pdd s_201;
    }
sdd b0 {minor 2;
      pdd b_100;
      pdd b_101;
      pdd b_200;
      pdd b_201;
    }
ldd sdev0 {minor 26;
      sdd s0;
    }
ldd sdev1 {minor 27;
      sdd b0;
    }

   /*
```

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

 * BEGIN SECTION:  SPECIAL SYSTEM DEVICES
   */
swapdev is ldd swap;
rootdev is ldd root;
   /*
 * END SECTION:  SPECIAL SYSTEM DEVICES
   */
}
network {
            0 npmaxdevs;
            0 npmaxpaths;
            2 enmaxdevs;
           2 fdmaxdevs;

        0700 npdirmode;
        0600 npfilemode;
        0700 hidirmode;
        0600 hifilemode;

        fddev 0 {
                iopath {
                        cluster 0;
                        eiop 0;
                        channel 040;
                }
        }
        endev 0 {
                iopath {
                        cluster 0;
                        eiop 0;
                        channel 020;
                }
        }
}
```

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

Note

To do this procedure you must be super user; you will see the snxxxx# prompt.

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

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

1. Enter the menu system:

Note

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

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

2. Select the following menu:

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

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

4. Subsection 5.8.1, page 69, describes the sections of the `/sys/param` file.

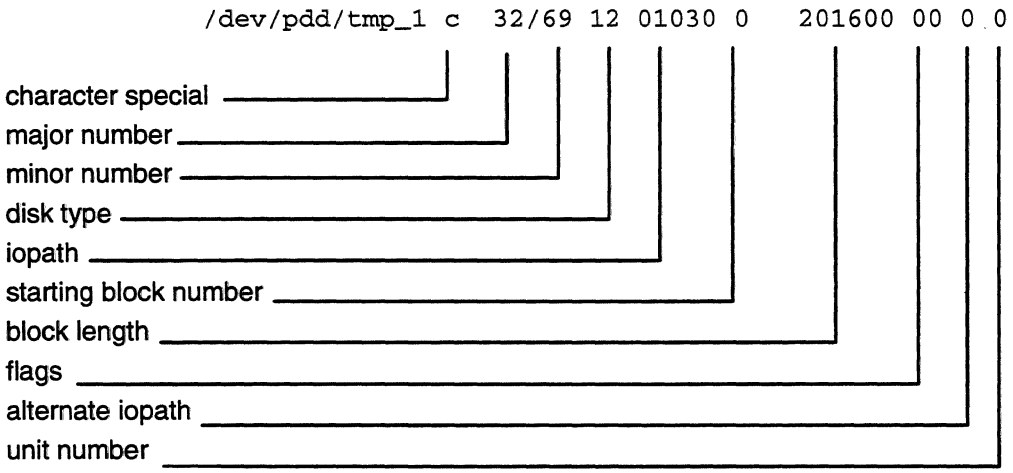
A sample menu screen follows:

```
                        Disk Configuration

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

If you are not using the menu system, and if you have not changed your configuration since the system was last booted, look at the UNICOS `/CONFIGURATION` file. If you have changed your configuration since the system was last booted, the following commands will display information so that you can identify the devices on your system and their file system allocation:

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

```
$ ddstat /dev/dsk/tmp

/dev/dsk/tmp b 34/69 0 0 /dev/ldd/tmp
       /dev/pdd/tmp_1 c 32/69 12 01030 0 201600 00 0 0 0
       /dev/pdd/tmp_2 c 32/96 12 01032 0 201600 00 0 0 0
       /dev/pdd/tmp_3 c 32/97 12 01036 0 201600 00 0 0 0
```


/dev/pdd/tmp_1 c 32/69 12 01030 0 201600 00 0 0
character special
major number
minor number
disk type
iopath
starting block number
block length
flags
alternate iopath
unit number

Procedure: Modifying your configuration file

Note

To do this procedure you must be super user; you will see the snxxxx# prompt.

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

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

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

Note

A CRAY J90 IOS-V is case sensitive; enter all lowercase characters for IOS-V commands. A CRAY EL IOS system converts all characters to upper case; enter either uppercase characters or lowercase characters for EL IOS commands.

1. Make a back-up of any file system that will be changed in your revised configuration file (`sys/param`) by using the `dump` command. See section 6, page 109.

2. Make a back-up copy of your current configuration file (the CRAY EL IOS prompt is `IOS>`):

```
snxxxx# <CONTROL-A> (toggles to the IOS)
snxxxx-iosx> cp /sys/param old.param
snxxxx-iosx> <CONTROL-A><RETURN> (toggles to UNICOS)
```

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

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

4. Edit your copy of the parameter file on UNICOS (see subsection 5.8.1, page 69).

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

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

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

6. When you are done making your configuration changes, copy your new version of the system configuration (`new.param`) on top of the old original version of the system configuration (`sys/param` on the IOS disk), using the `/bin/exdf`) command. The following command specifies that the `new.param` file will be written to the IOS system disk (the `-o` output option and < redirection) and be named `sys/param`:

Caution

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

```
snxxxx# exdf -o sys/param < new.param -r
```

7. At this point, the next time UNICOS is booted, it will come up with the new system configuration that you specified, and the system will copy the IOS `sys/param` file to two UNICOS files: `/IOS-param` and `/CONFIGURATION`.

8. After the system is booted to single-user mode, you must make, label, check, and mount any file system (old or new) that differs in any way from the way it was previously defined in the original version of the IOS `sys/param` file you changed. (subsection 5.10 describes these additional steps.) You then must restore altered file systems from the back-up tapes you created in step 1 of this procedure by using the `restore` command.

Creating file systems
5.10

After you have planned the configuration of your physical and logical devices and defined them using CSL, you must follow the steps described in this subsection to create file systems on your logical devices. (To determine the devices provided with your system and how they are allocated to file systems, see subsection 5.8.2.5, page 73.)

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

The remainder of this section describes the following:

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

Note

UNICOS System Administration, publication SG–2113, includes information on other aspects of file system maintenance. For example, the file system space monitoring capability can improve the usability and reliability of the system. Space monitoring observes the amount of free space on mounted file systems and takes remedial action if warning or critical thresholds are reached. Also, the file system quota enforcement feature (also called *disk quotas*) lets you control the amount of file system space in blocks and the number of files used by each account, group, and user on an individual basis. You may apply controls to some or all of the configured file systems, except for the `root` file system. Attempts to exceed quota limits cause an error similar to the error that occurs if the file system is out of free space. Optional warning levels also are available for informing users when usage gets close to a quota limit.

Step 1: Building the file system

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

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

`-q` Specifies quick mode; bypasses surface check.

`-n` *blocks* (Optional) Specifies number of blocks you want the file system to contain.

`-a` *strategy* (Optional) Specifies an allocation strategy. This option can take one of the following values:

`rrf` Round-robin all files (default)
`rrdl` Round-robin first-level directories
`rrda` Round-robin all directories

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

`-A` *blocks* (Optional) Specifies the minimum number of 4-Kbyte blocks allocated for a file whose size is greater than or equal to `BIGFILE` (see the `-B` option). The default is 21 sectors (blocks) and is defined by the `BIGUNIT` argument in `/usr/src/uts/sys/param.h`; you cannot change the definition. The default might be the value you want to use at your site.

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

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

device (Required) Full path name of the block special file (`/dev/dsk/`*filename*). When the disk configuration is activated at system startup, block special files are created for each logical device in your configuration. They are placed in the `/dev/dsk` directory and take on the same name as the logical device. You must know the full path name.

A basic example follows:

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

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

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

Example 1:

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

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

Example 2:

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

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

Example 3:

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

Continue with step 2.

Step 2: Labeling the file system

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

```
/etc/labelit device filesystemname volumename
```

device (Required) The name of the logical device that you want to label.

label The actual label consists of the following two required fields:

filesystemname The name you want to assign to the file system.

volumename The name you want to assign to the volume.

Note

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

Example 1:

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

The syntax is `/etc/labelit` *device filesystemname volumename*

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

Example 2:

The following command uses the old syntax, which has been preserved for compatibility with previous versions of the `labelit`(8) command:

The syntax is `/etc/labelit -f` [*filesystemname* `-v` *volumename*] *device*

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

Example 3:

If you do not specify a label, `labelit` displays current label information about a file system, as shown in example 3, which specifies only the file system name:

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

Continue with step 3.

Step 3: Checking a file system

Note

You must check a file system **before** it is mounted; otherwise, the file system will not be mounted. Before mounting a file system, always perform a consistency check on it to ensure that a reliable environment exists for file storage. When the system is brought to multiuser mode the `/etc/bcheckrc` multiuser level start-up script automatically checks any file systems listed in the `/etc/fstab` file. The `/etc/fstab` file also has an option that can cause its files to be mounted automatically at multiuser start-up time (see subsection 5.11.2, page 103). Because of the multipass nature of the `/etc/fsck` command, the file systems must be in an inactive state while being checked. You must ensure that all file systems to be checked are unmounted.

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

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

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

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

Phase	Description
1: Check blocks and sizes	Examines the file system's inode list for duplicate blocks, incorrect block numbers, or incorrect format.
2: Check path names	Removes directory entries that were modified in phase 1.
3: Check connectivity	Checks the connectivity of the file system, verifying that each inode has at least one directory entry and creating error messages for unreferenced directories.

Phase	Description
4: Check reference counts	Lists errors from missing or lost directories, incorrect link counts, or unreferenced files.
5: Check free list	Checks the relationship between the number of allocated blocks in the file system, the number of blocks in use, and the difference between the two (the *free block list*). If the current free block count (immediately calculated) is not the same as the free block list, an error is reported.
6: Salvaging	Occurs only if an error occurred in phase 5 and you answered YES to the SALVAGE? prompt.

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

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

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

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

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

Example:

```
/etc/fsck /dev/dsk/elssa
```

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

Note

A file system can become corrupted in a variety of ways, the most common of which are hardware failures and improper shutdown procedures. If proper startup procedures are not followed, a corrupted file system will become further corrupted.

A hardware failure can occur because of the following:

- Disk pack error
- Controller failure
- Power failure

An improper system shutdown can occur because of the following:

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

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

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

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

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

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

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

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

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

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

Continue with step 4.

Step 4: Creating a mount point for the file system

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

Example:

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

Continue with step 5.

Step 5: Mounting the file system

A file system is a sequential array of data until it is mounted. When the file system is mounted, the UNICOS kernel interprets the data as a UNICOS file system that is available as part of the system's complete directory structure. To be accessible to the UNICOS system, all file systems except `root` (`/`) must first be explicitly mounted by using the `mount`(8) command. The file system is mounted on an existing directory. The directory may have to be created, using the `mkdir`(1) command (see step 4). By convention, the name of the directory corresponds to the name of the logical device. The fourth field of the `/etc/fstab` file controls the automatic mounting of user file systems when going to multiuser mode. (For steps to configure a file system to be mounted automatically at initialization of multiuser mode, see page 105.)

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

Note

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

Example:

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

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

Important

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

/etc/mnttab and /etc/fstab files
5.11

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

/etc/mnttab
5.11.1

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

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

/etc/fstab
5.11.2

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

Note

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

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

```
/etc/mount /dev/dsk/elssa /elssa
```

You can type one of the following commands instead:

```
/etc/mount /elssa

/etc/mount /dev/dsk/elssa
```

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

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

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

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

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

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

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

```
#ls -la /etc/config/rcoptions
-r--r--r-- 1 root root 1914 Mar  8 11:29 /etc/config/rcoptions
# chmod 644 /etc/config/rcoptions
-rw-r--r-- 1 root root 1956 Mar  8 17:28 rcoptions
# vi rcoptions
   (make changes)
# chmod 444 /etc/config/rcoptions
-r--r--r-- 1 root root 1914 Mar  8 11:29 /etc/config/rcoptions
```

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

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

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

3. Edit the fourth field for the desired file system entry to read `CRI_RC=YES`.

4. Save the changes you have made to the `/etc/fstab` file.

Procedure: Unmounting file systems

At shutdown, all file systems are unmounted by the `/etc/shutdown` script; however, you may want to make a file system unavailable during normal operation for maintenance purposes. By convention, the `/mnt` directory is used to mount a file system that needs maintenance. To make a file system unavailable to users, unmount it by using the `umount` command.

Caution

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

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

Examples:

```
# /etc/umount /elssa
```

```
# /etc/umount /dev/dsk/elssa
```

Backing Up and Restoring File Systems [6]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

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

Note

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

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

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

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

Related backup and restore documentation

6.1

The following documentation contains information covered in this section:

- *UNICOS System Administration*, publication SG–2113, sections on file systems planning and configuring UNICOS
- *UNICOS User Commands Reference Manual*, publication SR–2011: `cpio`(1), `dd`(1), `rls`(1), `rsv`(1), `tar`(1), and `tpmnt`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*, publication SR–2014: `dump`(5) and `fstab`(5) man pages
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `dump`(8), `rdump`(8), `restore`(8), and `rrestore`(8) man pages
- *CRAY IOS-V Messages*, publication SQ–2172: procedure to dump the IOS-V.
- *CRAY EL Series IOS Messages*, publication SQ–2402: procedure to dump the CRAY EL IOS.

CRAY EL tape devices referenced in /dev
6.2

For CRAY EL systems, to use the various UNICOS utilities to back up and restore files not using the `tpdaemon`, the tape devices referenced are in the `/dev` directory, not the `/dev/tape` directory in which the `tpdaemon` addressable devices reside. Use the "no rewind" version of these devices for writing multiple saved datasets to a single tape or when you must span across more than one tape. The following are examples of tape devices that can be configured in the `/dev` directory (for more information on naming tape devices, see section 15, page 357):

Tape device	Description
`/dev/[n]rmt00`	Nine-track PERTEC round tape; no rewind on close is `/dev/nrmt00`.
`/dev/[n]rsm00`	STK 9914 round (SCSI) tape; no rewind on close is `/dev/nrsm00`.
`/dev/[n]rpq01`	Quarter-inch tape QIC cartridge; no rewind on close is `/dev/nrpq01`.
`/dev/[n]rpe02`	8-mm EXABYTE; no rewind on close is `/dev/nrpe02`.
`/dev/[n]rpd03`	Digital Audio Tape (DAT) device; no rewind on close is `/dev/nrpd03`.
`/dev/[n]rss000`	Eighteen-track IBM-compatible square tape; no rewind on close is `/dev/nrss000`.

Backup and restore utilities
6.3

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

dump *and* restore *utilities*
6.3.1

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

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

rdump *and* rrestore *utilities*
6.3.2

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

dd *utility*
6.3.3

The dd utility is recommended for copying data directly from a disk partition. dd is a good tool for creating absolute block-by-block copies of entire file systems (for example, copying your root file system to a back-up root file system). The dd utility converts and copies a file to the specified output device (disk-to-disk backup).

tar *and* cpio *utilities*
6.3.4

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

/etc/dump utility
6.4

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

Option	Description
-A *altfile*	Specifies the name of a file to contain a second copy of the output from the beginning of dump.
-c	Writes to cartridge tape.
-f *file*	Places the dump in a disk file, rather than on tape.
-t *dump_level*	Specifies the dump level. Default is level 0, which is a full system dump.
-u	Writes the date and time of the beginning of dump in the /etc/dumpdates file. A separate date is maintained for each file system and dump level.
-v *vsn list*	Specifies a list of volume serial numbers to use for output. If you omit this option, dump prompts the operator for a list of VSNs.
-w	Prints the file systems that must be dumped. This information is gathered from the dump frequency field in the /etc/fstab and /etc/dumpdates files.

Note

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

Routine backup (dump) strategy
6.5

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

The dump utility dumps all file system files that have been modified since the most recent dump that was performed at a lower level. For example, if a level 0 dump was performed on Sunday, a level 9 dump was performed on Monday, a level 8 dump was performed on Tuesday, and a level 9 dump was performed on Wednesday, then Monday's level 9 dump tapes would contain all changes since Sunday's level 0 dump. Tuesday's level 8 dump tapes also would contain all changes since Sunday's level 0 dump (Sunday's level 0 dump is the most recent dump with a dump level value less than 8). Wednesday's level 9 dump tapes would contain all changes since Tuesday (Tuesday's level 8 dump is the most recent dump with a dump level less than 9).

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

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

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

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

Restoring file systems
6.6

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

Options	Description
`-c`	Reads from cartridge tape.
`-f` *file*	Reads the dump from a disk file, rather than tape.
`-i`	Initiates interactive restoration. A shell-like interface is provided that lets the user traverse through the directory tree and select files to be restored.
`-r`	Reads entire tape and loads into current directory. Do this only if you `mkfs` the file system first. You usually should do a full `dump` after a full `restore`.

Options	Description
`-t`	Lists the specified file names if the files are on the tape. If no file names are specified, all files on the tape are displayed.
`-v` *vsn*	Causes `restore` to type the name of each file it treats, preceded by its file type.
`-x`	Extracts specified files from the tape (creates subdirectories as necessary).

Note

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

Increasing and decreasing file system space

6.7

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

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

Not all file systems suffer from fragmentation. For example, `/root` and `/usr` contain many directories and commands that never change; but the user and spooling (if separated) file systems are in constant change.

When you want to decrease file system fragmentation, perform a full dump and restore of that file system.

Procedures included in this section

6.8

This section includes the following procedures:

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

Note

To back up and restore in batch requires you to set limits on the UDB account being used and on the NQS queue. You also must use the `qsub -lU` command.

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

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

Note

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

---

1. If a `/bin/file` command shows the UNIX tape logical device to which you want to dump is **not** currently configured in the `/dev` directory, you must create it by using `mknod` commands (see section 15, page 357). Typically these devices have names such as `/dev/rss00` (`CART`), `/dev/rmt00` (`TAPE`), `/dev/rpe02` (`EXB`), `/dev/rpq01` (`QIC`), `/dev/rpd03` (`DAT`), and so on.

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

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

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

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

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

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

3. Physically mount a tape in the tape drive that matches the logical tape device you want to use (for this example, a square (`CART`) tape was mounted in a drive that corresponds to the logical device `/dev/rss00`).

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

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

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

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

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

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

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

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

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

```
sn511# /etc/dump -t 0 -u -f /dev/rss00 /dev/dsk/src
```

Note

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

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

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

Your file system backup (dump) is now complete.

Procedure: Restoring a file system without tpdaemon

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

1. If a /bin/file command shows the UNIX tape logical device you want to access is not currently configured in the /dev directory, you must create it by using mknod commands (see section 15, page 357). Typically these devices, have names such as /dev/rss00 (CART), /dev/rmt00 (TAPE), /dev/rpe02 (EXB), /dev/rpq01 (QIC), /dev/rpd03 (DAT), and so on.

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

Caution

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

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

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

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

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

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

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

```
sn5111# /etc/mount

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

Warning

Step 3 deletes all information on this file system.

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

   The `-q` option on the `mkfs` command shown in the next example is optional syntax. Using this option bypasses the disk surface check and speeds the `mkfs` process, but it is not the most thorough way to prepare the disk for a file system structure. The first time this command is used to format a logical disk area for a file system structure, the `-q` option probably should not be used. For more information about the `mkfs` command, see section 5, page 57, and the `mkfs` man page.

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

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

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

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

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

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

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

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

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

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

```
sn5111# cd /mnt
```

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

9. Rewind the tape. Round-type tape drives rewind the tape automatically when you push the button to select the tape to be loaded. You should be cautious and rewind other types of tapes when they are used. However, to rewind other types of tapes, enter the `mt -f /dev/`*tapename* `rew` command and specify the tape device you are rewinding (`-f /dev/`*tapename* specifies the raw tape device to be activated). For example, to rewind an EXABYTE (`/dev/rpe02`) tape, type the following command line:

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

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

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

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

    You can invoke the interactive method at this point by typing the following command line:

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

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

    To do a full file system restore:

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

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

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

    To do a partial file system restore:

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

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

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

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

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

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

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

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

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

12. To remount the restored file system on its normal mount point, check the file system, for example (if the file system was unmounted cleanly, this step is optional):

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

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

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

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

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

For this procedure, it is assumed that the tpdaemon is up and that all tape hardware (devices, controllers, and so on) are configured to be up and available to the user. In the following example, a new tape is used, and it will be specified as unlabeled. The volume name used is arbitrary.

Note

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

Caution

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

If you are in single-user mode (no file systems mounted other than root and no daemons started) and you want to back up the /dev/dsk/usr file system, you cannot invoke an operator window (needed to answer tpdaemon tape-related questions during the backup process), because that command is in /usr (/usr/lib/msg/oper). If you mount /dev/dsk/usr so that you can use the commands and daemons that reside in the /dev/dsk/usr file system to do the backup and restore, you can successfully perform the backup. You must also start any daemons you need (such as tpdaemon and msgdaemon) if you are in single-user mode. While in single-user mode and working from a nonwindow environment (such as the WYSE terminal master console), you should run the dump command with an & symbol appended so that the command runs as a background process. This enables you to keep your window free to invoke the operator message window (/usr/lib/msg/oper) to answer the operator messages your backup procedure will generate.

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

   A tpgstat display shows all possible available tape types for each user, how many of each type that user has reserved, and for how long. In the following sample tpgstat display, the

output shows that no user has any tapes reserved. For all tape types (device groups) that are currently configured UP (through the `tpconfig` command), it will appear as if the `tpdaemon` has reserved that tape type:

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

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

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

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

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

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

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

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

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

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

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

   The dump command and its output follow:

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

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

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

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

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

Note

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

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

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

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

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

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

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

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

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

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

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

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

  1    17:07   TM122 - mount tape JON1(nl) on a CART device for jones
                  14,  () or reply cancel / device name

:
: display truncated
:
Enter '?' for help.
>
```

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

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

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

6. Respond to the next operator message by specifying the volume serial number of the tape.

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

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

All messages related to your tape job have now disappeared.

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

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

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

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

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

Note

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

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

Procedure: Restoring a full file system by using tpdaemon

Assumptions

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

A full file system restore is being performed. This means that the entire contents of the specified file system are read in from tape and **overwrite** the current disk version of that file system.

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

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

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

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

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

Procedures

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

1. Determine which tape device group you plan to use for your restore. This will be the same as the preceding backup (see step 1 of the backup procedure).
2. Verify that the tape(s) written from the preceding backup are set to prevent tape contents from being overwritten (see step 2 of the backup procedure).
3. If it was not already unmounted, unmount the file system to be restored by using the `/etc/umount` command. The `/dev/dsk/src` file system is used in the rest of this example.

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

```
sn5111# /etc/mount
/ on /dev/dsk/root read/write on Mon Oct 11 10:38:15 1993
/tmp on /dev/dsk/tmp read/write on Mon Oct 11 10:40:56 1993
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Mon Oct 11 10:40:59
1993
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Mon Oct 11
10:41:02 1993
/usr/src on /dev/dsk/src read/write,rw,CRI_RC="YES" on Mon Oct 11 10:41:04
1993
```

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

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

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

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

```
sn5111# /etc/mount
/ on /dev/dsk/root read/write on Mon Oct 11 10:38:15 1993
/tmp on /dev/dsk/tmp read/write on Mon Oct 11 10:40:56 1993
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Mon Oct 11 10:40:59
1993
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Mon Oct 11
10:41:02 1993
```

Warning

Step 4 **deletes all information** on this file system.

4. Remake the file system structure on the `/dev/dsk/src` logical device by using the `/etc/mkfs` command.

The -q option on the mkfs command that follows is optional syntax. You always should use the -q option for CRAY EL systems. For more about the mkfs command, see the mkfs man page.

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

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

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

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

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

```
sn5111# /etc/mount
/ on /dev/dsk/root read/write on Tue Oct 12 19:09:25 1993
/tmp on /dev/dsk/tmp read/write on Tue Oct 12 19:10:01 1993
/usr on /dev/dsk/usr read/write,rw,CRI_RC="NO" on Tue Oct 12 19:10:04
1993
/usr/home on /dev/dsk/home read/write,rw,CRI_RC="YES" on Tue Oct 12
19:10:07 1993
/usr/src on /dev/dsk/src read/write,rw,CRI_RC="YES" on Tue Oct 12 19:10:09
1993
```

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

7. Mount the file system on the mount point you have selected by using the /etc/mount command. In this example, the mount point /mnt is used. If any user is in the directory or any of its subdirectories, the mount command will not be successful (to remove users forcibly from a file system, see the fuser man page). If you are the one in the directory or a

subdirectory, change to a directory that is not part of the directory tree, including the mount point directory or any of its subdirectories, as follows:

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

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

```
sn5111# cd /mnt
```

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

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

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

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

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

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

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

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

Note

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

---

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

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

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

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

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

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

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

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

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

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

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

  1     17:07    TM122 - mount tape JON1(nl) on a CART device for jones
                    14,  () or reply cancel / device name

:
: display truncated
:
Enter '?' for help.
>
```

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

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

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

12. Respond to the next operator message by specifying the volume serial number of the tape.

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

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

All messages related to your tape job have now disappeared.

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

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

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

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

```
sn5111# /etc/restore -r -V INC1 -l nl -g CART
```

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

    Example:

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

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

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

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

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

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

Note

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

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

Procedure: Restoring a partial file system by using tpdaemon

Assumptions

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

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

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

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

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

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

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

Procedures

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  1     17:07    TM122 - mount tape JON1(nl) on a CART device for jones
                    14,  () or reply cancel / device name

:
: display truncated
:
Enter '?' for help.
>
```

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

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

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

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

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

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

All messages related to your tape job have now disappeared.

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

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

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

Note

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

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

Maintaining Users [7]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

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

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

For information on ways to communicate with users, see section 8, page 211.

Related user accounts documentation
7.1

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

- *UNICOS User Commands Reference Manual*, publication SR–2011: `chgrp`(1), `chown`(1), `passwd`(1), `su`(1), and `udbsee`(1) man pages
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `nu`(8), `udbgen`(8), and `udbpl`(8) man pages
- *UNICOS File Formats and Special Files Reference Manual*, publication SR–2014: `acid`(5), `cshrc`(5), `group`(5), `passwd`(5), `profile`(5), `shells`(5), and `udb`(5) man pages
- *UNICOS System Administration* publication SG–2113, the "Basic Administration" section

The user database (UDB)
7.2

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

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

You can modify the UDB in two ways:

- By using the `/etc/nu` utility (see subsection 7.5, page 165)

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

- By using the `/etc/udbgen` utility (see subsection 7.6, page 185)

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

Using the UDB
7.3

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

- Learn about the UDB fields and decide which values to assign to the UDB fields (more than 80 fields exist). The following subsection describes a suggested subset of UDB fields (see page 155). For a full listing and explanation of all possible fields in the UDB, see the `udbgen` man page. Some of the values you select will affect other factors on the system (for example, the login directory field determines in which file system the user is placed). You must make sure sufficient disk space is available to meet the user's needs in this file system.
- If the user will be placed in a new group that you will reference by name, add the new entry in `/etc/group`. (See page 161.)
- If the user will be placed in a new account group that you will reference by name, add the new entry in `/etc/acid`. (See page 163.)

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

- Follow the procedures in subsection 7.5, page 165, to make the new entry in `/etc/udb` by using the `/etc/nu -a` command. (Subsection 7.5 also includes procedures for modifying and deleting user records.)

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

- Follow the procedures in subsection 7.6, page 185, to make the desired entry in `/etc/udb` by using the `/etc/udbgen` command. (Subsection 7.6 also includes procedures for modifying and deleting user records.)

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

UDB files and commands
7.4

The following are the principal files related to the UDB:

File	Description
`/etc/acid`	Account name/ID map file for accounting billing group.
`/etc/group`	Group name/ID map and membership file. This file is common to all UNIX environments.
`/etc/nu.cf60`	The nu utility configuration file.
`/etc/passwd`	UNIX password file used for compatibility with existing commands. The * symbol replaced the encrypted password field. This file is common to all UNIX environments. Encrypted passwords are stored in the `/etc/udb` file.
`/etc/udb`	Primary user database file; binary file. It stores the user's password.
`/etc/udb.public`	Public version of `/etc/udb` with read permission for "world." All sensitive information has been set to 0.

Note

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

Script	Description
`/etc/nulib/nu1.sh`	The nu utility uses this script to create a user directory and to change the permissions on this directory.
`/etc/nulib/nu2.sh`	The nu utility uses this script to initialize the contents of the user's directory.
`/etc/nulib/nu3.sh`	The nu utility uses this script to purge a login account.
`/etc/nulib/nu4.sh`	The nu utility uses this script to purge a login without removing the account from the UDB. This action is performed to preserve accounting information.

The following are the principal commands related to the user database:

Command	Description
`/etc/nu`	Adds, deletes, and modifies login records. `/etc/nu` uses the following scripts:
`/bin/passwd`	Creates or changes a user's password.
`/bin/udbsee`	Converts information from the user database into an ASCII format.
`/etc/udbgen`	Generates and maintains the user database.
`/etc/udbpl`	Writes to `stdout` administrative information for designated users.

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

Procedure: Determining settings for UDB fields

The UDB (`/etc/udb`) contains information for each user who is allowed to log in and run jobs on your system. The UDB also contains many other fields that are specific to the UNICOS environment. Fields that you can specify for each user include settings that specify limits for batch processing, interactive processing, account security, the data migration facility, CPU access, disk quotas, the fair-share scheduler, and many others. You must provide the appropriate settings for the fields and resource limits in the UDB for each user record.

For a full listing and explanation of all possible fields in the UDB, see the `udbgen`(8) man page, which includes several examples.

Note

The following UDB fields are a suggested minimum subset of the UDB fields that you should define for each user. The "`keyword:`*value*`:`" syntax of each entry that follows reflects the format accepted by the UDB if you use the `/etc/udbgen` utility; however, when using the `/etc/nu` utility, you do not use this format (see subsection 7.5, page 165).

Basic user account definition fields

You should define the following UDB fields for each user (for all possible fields, see the `udbgen`(8) man page).

Note

The global default table contains entries for some of the UDB fields; for a list of these fields, see the `udbgen`(8) man page. The release defaults are applied by `udbgen` when it updates a UDB that has a default table that contains all zeros. To create a default table in an existing UDB, execute the `udbgen -c'#'` command. This command is an empty modification request, but it causes the default table to be created with the released defaults. To change one or more entries, write the appropriate directive line (see "Adding users to `/etc/udb` by using `/etc/udbgen`," page 187).

Login name field

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

Encrypted password field

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

Password aging field

pwage :force, superuser, *max, min, time*:
Manipulate password age control fields by using pwage. If you omit a keyword, also omit its separating comma.

The *max, min,* and *time* fields control how old a password can become (max), how long it has to exist before being changed (min), and when it was changed (time). Neither *max* nor *min* may exceed 64 weeks.

pwage :force, superuser, *max, min, +age*:
In the second form of pwage, a plus sign preceding the last numeric value causes *age* to be interpreted as the amount of time to subtract from the time "now" to result in a value of the time-of-day clock that is age units in the past and then store that value in the time field. Usually, this is intended to make it easy to set the current time in the field by using the value +0 as the age.

You must precede the time with two commas if you do not specify *max* and *min* ages, because this part of the directive is position dependent, reading from the left to determine the meaning of the value string.

pwage : *max, min*:
The third form of pwage alters the *max* and *min* age fields.

pwage ::
The fourth form of pwage removes only age control from a record. All age control fields are set to a 0 or null state, which removes age control totally. After this has been done, all historical information is lost from the record. When the YP permbit is set (see permbits) and the password is being accessed from the database, password aging is disabled.

User ID field

`uid :`*n*`:`
`uid :next:`

Unique number that represents the user ID. If the value is `next`, the next highest user ID from the UDB is assigned to this user. The value 0 indicates a superuser login. The highest value that you may use is defined in `sys/param.h` as `UID_MAX-1`.

Group ID field

`gids =|+|- :`*n1*`, `*n2*`, ..., `*nn*`:`
`gids =|+|- :`*g1*`, `*g2*`, ..., `*gn*`:`

Comma-separated list of numeric group IDs or group names to which the user belongs. The group limit is 64. If group names are used, they must be found in the `/etc/group` file before executing the `udbgen` command.

User comment field

`comment :`*text*`:`

Comments that consist of a maximum of 39 characters; white space is not removed. This field is often used for indicating the user's full name, although a site may have other uses for this optional field.

Login (`home`) directory

`dir :`*directory*`:`

Default login (`home`) directory for this user relative to the `root` directory. The `dir:directory:` consists of a string of up to 63 characters. Typically, root is `/`; therefore, `dir` is based on the `root` (`/`) directory, but this does not have to be true. If you do not specify a value, the user is logged in under the `/` directory.

User shell at login

`shell :`*sh_name*`:`

Default login shell. You can specify a maximum of 63 characters. Default value for *sh_name* is `/bin/sh`.

Account ID field

`acids =|+|- :`*n1*`, `*n2*`, ..., `*nn*`:`
`acids =|+|- :`*a1*`, `*a2*`, ..., `*an*`:`

Account IDs. This is a list of up to 64 numeric account IDs or account names separated by commas. If you use account names, they must be found in the `/etc/acid` file as it existed before `udbgen` was executed.

Login `root` directory

`root :`*directory*`:`

Login `root` directory. You can specify a string of up to 63 characters. `root` specifies the directory to which the base of the user's directory tree is set. (For further information, see the `chroot`(2) man page.)

Nice value

`nice[b] :n:`
`nice[i] :n:`

The `nice` value bias in the range 0 < *n* < 19 for batch (`[b]`) or interactive (`[i]`) processes. If you do not specify this field, the value from the default table or the released default value of 0 is used. This field is useful for getting different interactive versus batch and NQS scheduling priority.

User resource limit fields

You can specify user resource limits for both batch and interactive processing in the UDB. The following is a list of some user limits that you may want to set; for a complete list of available limits, see the udbgen(8) man page.

Note

A UDB field setting of 0 means "infinite," except for tape access, where 0 means the user has no tape privileges.

CPU limits

Job CPU time limit
`jcpulim[b] :n:`
`jcpulim[i] :n:`

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

Per-process CPU limit
`pcpulim[b] :n:`
`pcpulim[i] :n:`

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

Memory limits

Job memory limit
`jmemlim[b] :n:`
`jmemlim[i] :n:`

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

Per-process memory limit
`pmemlim[b] :n:`
`pmemlim[i] :n:`

Per-process memory limit in 4096-byte blocks for batch(`[b]`) or interactive (`[i]`) processes. The default is unlimited.

Process limits

Job process limit
`jproclim[b] :n:`
`jproclim[i] :n:`

Job process limit for batch (`[b]`) or interactive (`[i]`) jobs. If you do not specify a value for this field, the default is the value of `/MAXUP` in `sys/param.h`.

SDS limits

SDS is not supported on CRAY J90 or CRAY EL systems; CRAY J90 and CRAY EL system administrators should ignore these fields and use the default setting.

Tape limits

Job tape unit limit
`jtapelim[b][t]:`*n*:
`jtapelim[i][t]:`*n*:

Job tape unit limit for batch (`[b]`) or interactive(`[i]`) jobs. The integer value `t` represents the tape type. The default tape types are defined in the `DEVICE_GROUPS` section of the `/etc/config/tapeconfig` file. The first type defined in that section is represented by `t=0`, the second is `t=1`, and so on. If *n* is 0, the user is denied tape access.

File limits

Per-process file allocation limit
`pfilelim[b] :`*n*:
`pfilelim[i] :`*n*:

Per-process file allocation limit in 4096-byte blocks for batch (`[b]`) or interactive (`[i]`) processes. If *n* is 0, the user's file allocation is unlimited.

Procedure: Adding a group to `/etc/group`

Note

An important step in adding a user record to the UDB is to assign the user to a group or groups. You may have to add group definitions so that you can make group assignments when you add user records.

As system administrator, you maintain the `/etc/group` text file, which contains the names of groups to which users belong. Groups are created to gather together users who have common needs for accessing files or programs.

You may have to edit the `/etc/group` file to add new group names to the file. The `/etc/nu` command does not allow you to enter a group name in the `gids` field until it has been entered in the `/etc/group` file; however, you may use group ID numbers even if no entry line for that group ID number is in the `/etc/group` file. In this case, a group name is created with the form `G-`*nnnnn*; *nnnnn* is the group ID number. The `/etc/nu` utility updates this file by adding login names to the group login name field. The file contains one entry for each UNICOS group. To delimit an entry line for a group, use a new line character.

To add a group to your system, edit the `/etc/group` file by adding an entry with the following format; **fields must be separated by a colon**:

```
group_name:unused_password_field_string:group_id:
```

group_name Name that you choose to reflect the group of users. The group name consists of 1 to 8 alphanumeric characters. The first character must be alphabetic. By convention, lowercase characters are used for group names.

password This field is not implemented under UNICOS. Place an unmatchable character string, such as *, in this field.

group_id The values 0 to 99 are reserved, by convention, for system-related groups; therefore, you can use group ID values 100 to `UID_MAX-1` for user groups. You should select the next available group ID number for the new group. Ensure that your *group_id* field is followed by a colon.

Note

When you create a new group, this field remains blank; ensure that your *group_id* field is followed by a colon. The udbgen and nu utilities maintain this field. A comma-separated list of login names placed in this group through the gids UDB field are placed automatically in this /etc/group field.

Example:

```
ops:*:62:
```

To see all group names to which a specified user belongs, use the groups command.

To complete adding user records to the UDB, use either the /etc/nu or /etc/udbgen utility.

Subsection 7.5, page 165, describes how to use /etc/nu, and subsection 7.6, page 185, describes how to use /etc/udbgen. To determine which utility will work best for you in a given situation, you may want to read both subsections.

Procedure: Adding an accounting group to /etc/acid

As system administrator, you maintain the /etc/acid file, which contains the names of accounts associated with users. Accounting groups are implemented for the accounting subsystem, allowing reports to generate information through accounting groups.

Just like the /etc/group file, you may have to edit this file to add new account names to the file. The /etc/nu command does not allow the use of account names in the acids field until an entry has been made in the /etc/acid file; however, you may use account ID numbers even if an entry line for that account ID number is not in the /etc/acid file. In this case, an account name is created of the form A-*nnnnn*; *nnnnn* is the account ID number. The file contains one entry for each UNICOS accounting group. A new line character delimits an entry line for each account.

To add an accounting group to your system, edit the /etc/acids file by adding an entry with the following format; **fields must be separated by a colon**:

```
account_name:account id
```

account_name Name that you select to reflect the accounting group (for easy identification in accounting reports). The name must consist of 1 to 79 alphanumeric characters. The first character must be alphabetic. Typically, lowercase characters are used for account names.

account_id (Account identifier) You can use account ID values 0 to UID_MAX-1.

Example:

```
markting:93
```

To complete adding user records to the UDB, use either the /etc/nu or /etc/udbgen utility.

Subsection 7.5, page 165, describes how to use /etc/nu, and subsection 7.6, page 185, describes how to use /etc/udbgen. To determine which utility will work best for you in a given situation, you may want to read both subsections.

Using the /etc/nu utility
7.5

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

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

Procedure: Changing /etc/nu configuration parameters

Default values for the /etc/nu utility are in the /etc/nu.cf60 configuration file. To change several parameters in this configuration file, either edit the file or use the menu system. You also can turn off or "hide" prompts for UDB values that you want the /etc/nu program to accept automatically.

The following are common parameters you may want to change; for a complete list of changeable parameters and a description of each, see the nu(8) man page:

Parameter	Description
DefaultAcids	String that contains the default account IDs assigned to the UDB acids field.
DefaultDr	Default login root string assigned to the UDB root field.
DefaultGids	Default group IDs assigned to the UDB gids field.
DefaultHome	Default directory used to create the login directory for a new user if no GroupHome declaration exists for the user's assigned group ID. The directory created will be $DefaultHome/*username*; *username* is the user login name.
DefaultShell	String that contains the default login shell assigned to the UDB shell field.
GroupHome	Default directory used to create the login directory for a new user in the associated group if no DefaultHome declaration exists for the user. Multiple GroupHome lines should exist for each group ID number. For example, if the declaration is GroupHome = 100 "/user1", user john (who is assigned to group 100) will have, by default, a login directory of /user1/john.
Security feature variables	Security feature parameters are used only when you turn on the security feature or use the -p option of nu.

Note

If you have configured the nu utility to skip prompting for specific UDB fields, you must use udbgen to access these fields.

If you are using the menu system, select the Configure System ==> NU Configuration menu and its submenus. You can import the default configuration file by executing the Import

`nu configuration` line; then modify the parameter settings and activate your changes. A sample `NU Configuration` menu screen follows:

```
Configure System
...NU Configuration
```

```
                                NU Configuration

 M-> Group home directories ==>
     Password aging ==>
     Tape Limits ==>
     Miscellaneous Limits ==>
     Ask about setting up Cray Station            1
     Login shell                                  /bin/csh
     Maximum login id length                      8
     nu log file                                  /usr/adm/nu.log
     Directory creation script                    /etc/nulib/nu1.sh
     Directory initialization script              /etc/nulib/nu2.sh
     Account removal script                       /etc/nulib/nu3.sh
     Account disabled script                      /etc/nulib/nu4.sh
     Default account ids                          0
     Default group ids                            0
     Default permbits                             none
     Default security level                       0
     Default maximum security level               0
     Default minimum security level               0
     Default default integrity class              0
     Default maximum integrity class              0
     Default valid compartment name string        none
     Default active compartment name string       none
     Default category name string                 none
     Default valid category name string           none
     Default permission name string               none
     Default resource group UID                   0
     Default allocation shares                    100
     Default login root                           /
     Default home directory                       /u      Import nu
configuration ...
     Activate nu configuration ...
```

If you are not using the menu system, edit the `/etc/nu.cf60` configuration file.

Procedure: Creating a file system to use with /etc/nu

The /etc/nu command defaults, which are set in the /etc/nu.cf60 file, expect a default home directory path of /usr/home.

To use the /etc/nu command, you must do **one** of the following:

- Change the DefaultHome parameter in the /etc/nu.cf60 file to match what your site will use for a default /home path. (See page 167.)

or

- Invoke the mkdir command to create a new directory called home in the /usr directory. This means that home will not be a separate file system, but just a subdirectory with files in the /usr file system.

or

- Create /usr/home as a file system and ensure that it is mounted on /usr when in multiuser mode.

  If you are using the menu system, select the Configure System ==> File System (fstab) Configuration ==> Standard File Systems menu, add your entries and update the form file. Then activate your changes through the File Systems (fstab) Configuration menu. A sample Standard File Systems Configuration menu screen follows:

```
Configure System
...File System (fstab) Configuration
......Standard File Systems
```

```
                    Standard File System Configuration

    Device Name       Mount Point     FsType  Freq  Pass  R/O  Quota   User
    --------------  --------------  ------  ----  ----  ---  ------  ---- >
E-> /dev/dsk/home      /usr/home       NC1FS    1     2    rw                
```

If you are not using the menu system, enter the following commands to accomplish this:

1. /bin/mkdir /usr/home
2. /etc/mkfs -q /dev/dsk/home
3. /etc/labelit /dev/dsk/home home vol1 (optional step)
4. /etc/fsck /dev/dsk/home

5. `/etc/mount /dev/dsk/home /usr/home`

6. To ensure that `/home` is always both checked and then mounted on `/usr` when running at multiuser level, edit the `/etc/fstab` file to check and mount `/home` automatically. Such an entry would look like the following:

```
/dev/dsk/home     /usr/home      NC1FS  rw,CRI_RC="YES"  1     2
```

At this point, the `/etc/nu` command will work as intended.

Procedure: Adding a user record to /etc/udb by using /etc/nu

Important

Before you begin this procedure, make sure you have completed the following:

- Determined the UDB field settings for the user account.
- Added needed group(s) to /etc/group to which the user will belong.
- Added needed account(s) that will be associated with the user if you have accounting implementation on your system.
- Created a /home or site-specific file system for user with /etc/nu.

For details on these procedures, see pages 155 through 170.

To use the nu utility to add a user to the /etc/udb file, use the following form of the nu utility (see example, page 173):

```
/etc/nu  -a
```

When you use the nu utility, the /etc/udb, /etc/udb.public, /etc/group, /etc/acid, and /etc/passwd files are updated, or you can maintain private (testing) versions. When you maintain private versions, you can move or copy them to /etc to install the updates.

The nu utility queries you for values to UDB fields; it also lets you accept default values that have been specified in the program's configuration file (/etc/nu.cf60).

Note

A UDB field setting of 0 means "infinite," except for tape access, where 0 means the user has no tape privileges.

For more information about user account field settings, see the procedure about determining settings for UDB fields, page 155.

The user's login name is something you provide when the /etc/nu utility prompts you. The user's login name must be a unique 1 to 8 alphanumeric representation, in which the first character is alphabetic. Typically, the name is made up of lowercase alphabetic characters.

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

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

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

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

Example /etc/nu session that adds a user:

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

```
# /etc/nu -a
cmd/nu/nu.c
Login name? (1-8 characters) [quit] jones
Enter password:
Please re-enter password:
Enter actual user name: John R. Jones
User id number? (max = 60000)  [2] 624
Which groups? (names or numbers, use commas, ? for list) [0]   (See procedure for adding groups)
cray,test,trng,usrsrc
You selected groups:
100 0
cray , 100
104 1
test , 104
105 2
trng , 105
98 3
usrsrc , 98
Are these correct? (y or n) [y]
Login directory?  [/hot/u1/jones]     (This will be the user's home directory.)
Enter shell [/bin/csh]
Which accounts? (names or numbers, use commas, ? for list) [0]   (See procedure for acids)
You selected accounts:
root , 0
Are these correct? (y or n) [y]
User default login root?  [/]   (This will be the user's root directory; set to other than / to restrict access to file systems.)
Resource group ID? (name or number, ? for list) [0] Users
Which permissions? (names or numbers, use commas, ? for list) [none]
You selected permbits:
none
Are these correct? (y or n) [y]
Allocation shares? (min=0) [100]
DEFAULT security compartments? (name1,name2,... or none, ? for list)
[default]
VALID security compartments? (name1,name2,... or none, ? for list)
[default]
Security permissions? (name1,name2,... or none, ? for list)  [default]
Security levels? (default max min)  [0 0 0]
```

```
Integrity classes? (default max)  [0 0]
DEFAULT integrity categories? (name1,name2,... or none, ? for list)
[default]
VALID integrity categories? (name1,name2,... or none, ? for list)
[default]
Do you want this user locked? (y or n) [n]
Do you want this user trapped? (y or n) [n]
Per job process limit for batch? (min=0)  [100]
Per job process limit for inter? (min=0)  [100]
Per job MPP PE limit for batch?  [none]
Per job MPP PE limit for inter?  [none]
Per job MPP time limit for batch?  [none]
Per job MPP time limit for inter?  [none]
Per job MPP barrier limit for batch?  [none]
Per job MPP barrier limit for inter?  [none]
Per process MPP time limit for batch?  [none]
Per process MPP time limit for inter?  [none]
Will the user be using the Cray Station? (y or n) [y] n
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  pwage ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624
   6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/rnl/tmp/jones
   8)  shell ...... /bin/csh
   9)  acids ...... root (0)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... default
  18)  valcomps ... default
  19)  permits .... default
  20)  intcls...... 0            21)  maxcls...... 0
  22)  intcat...... default      23)  valcat...... default
  24)  disabled ... 0            25)  trap ....... 0
  26)  jproclim[b] ..      100       jproclim[i] ..      100
  27)  jcpulim[b] ...     none       jcpulim[i] ...     none
  28)  pcpulim[b] ...     none       pcpulim[i] ...     none
```

(No for CRAY J90 or CRAY EL systems.)

```
  29)  jmemlim[b] ...       none        jmemlim[i] ...       none
  30)  pmemlim[b] ...       none        pmemlim[i] ...       none
  31)  pfilelim[b] ..       none        pfilelim[i] ..       none
  32)  jsdslim[b] ...    1048576        jsdslim[i] ...    1048576
  33)  psdslim[b] ...    1048576        psdslim[i] ...    1048576
  34)  jtapelim[b][type0   ]      99      jtapelim[i][type0   ]     99
       jtapelim[b][type1   ]      99      jtapelim[i][type1   ]     99
       jtapelim[b][type2   ]      99      jtapelim[i][type2   ]     99
       jtapelim[b][type3   ]      99      jtapelim[i][type3   ]     99
       jtapelim[b][type4   ]      99      jtapelim[i][type4   ]     99
       jtapelim[b][type5   ]      99      jtapelim[i][type5   ]     99
       jtapelim[b][type6   ]      99      jtapelim[i][type6   ]     99
       jtapelim[b][type7   ]      99      jtapelim[i][type7   ]     99
  35)  jpelimit[b] ...       none        jpelimit[i] ...       none
  36)  jmpptime[b] ...       none        jmpptime[i] ...       none
  37)  jmppbarrier[b]        none        jmppbarrier[i]        none
  38)  pmpptime[b] ...       none        pmpptime[i] ...       none
Are these values OK? (y or n)  [y]
Entry looks like:
jones:co:John R. Jones
jones:ui:624:di:/hot/u1/rnl/tmp/jones:sh:/bin/csh:dr:/:pw:v0u28k2K1wtX6
jones:gi:100,104,105,98
jones:ai:0
jones:rg:102:as:100
jones:dc:default:cm:default:pm:default
jones:ic:default:vc:default
jones:pj[b]:100:pj[i]:100
jones:js[b]:1048576:js[i]:1048576:ps[b]:1048576:ps[i]:1048576
jones:tp:type0[b]:99:tp:type0[i]:99:tp:type1[b]:99:tp:type1[i]:99
jones:tp:type2[b]:99:tp:type2[i]:99:tp:type3[b]:99:tp:type3[i]:99
jones:tp:type4[b]:99:tp:type4[i]:99:tp:type5[b]:99:tp:type5[i]:99
jones:tp:type6[b]:99:tp:type6[i]:99:tp:type7[b]:99:tp:type7[i]:99+
test 1 -ne 0
+ rm -rf /hot/u1/jones
+ mkdir /hot/u1/jones
+ test /hot/u1/jones != /hot/u1/jones
+ test 0 -eq 0 -a 1 -ne 0
+ chgrp 100 /hot/u1/jones
+ chown 624 /hot/u1/jones
+ chacid -s root /hot/u1/jones
Do you wish to add more new users? (y or n)  [y] n
execing udbgen - please wait
udbgen complete                          (At this time, nu executes udbgen.)
```

Procedure: Modifying user records by using /etc/nu

To update UDB fields, follow the same procedures as for adding new user logins, except use the -m option, rather than the -a option.

Note

A UDB field setting of 0 means "infinite," except for tape access, where 0 means the user has no tape privileges.

Example /etc/nu session that modifies a user's login:

The following example shows how to update user login entries in the UDB by using the nu utility. The example changes the account group (acids) for user jones (**bold** indicates what you would type):

```
# /etc/nu -m
cmd/nu/nu.c                >>> Modify mode <<<Enter user identifier
(login or uid) [quit]: jones
Entry is now:
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  pwage ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624
   6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/jones
   8)  shell ...... /bin/csh
   9)  acids ...... root (0)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... none
  18)  valcomps ... none
  19)  permits .... none
```

```
  20)  intcls...... 0              21)  maxcls...... 0
Press 'return' for the rest of the entry...
  22)  intcat...... none           23)  valcat...... none
  24)  disabled ... 0              25)  trap ....... 0
  26)  jproclim[b] ..       100         jproclim[i] ..       100
  27)  jcpulim[b] ...      none         jcpulim[i] ...      none
  28)  pcpulim[b] ...      none         pcpulim[i] ...      none
  29)  jmemlim[b] ...      none         jmemlim[i] ...      none
  30)  pmemlim[b] ...      none         pmemlim[i] ...      none
  31)  pfilelim[b] ..      none         pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576         jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576         psdslim[i] ...   1048576
  34)  jtapelim[b][type0   ]    99      jtapelim[i][type0   ]        99
       jtapelim[b][type1   ]    99      jtapelim[i][type1   ]        99
       jtapelim[b][type2   ]    99      jtapelim[i][type2   ]        99
       jtapelim[b][type3   ]    99      jtapelim[i][type3   ]        99
       jtapelim[b][type4   ]    99      jtapelim[i][type4   ]        99
       jtapelim[b][type5   ]    99      jtapelim[i][type5   ]        99
       jtapelim[b][type6   ]    99      jtapelim[i][type6   ]        99
       jtapelim[b][type7   ]    99      jtapelim[i][type7   ]        99
  35)  jpelimit[b] ...      none         jpelimit[i] ...      none
  36)  jmpptime[b] ...      none         jmpptime[i] ...      none
  37)  jmppbarrier[b]       none         jmppbarrier[i]       none
  38)  pmpptime[b] ...      none         pmpptime[i] ...      noneSelect
field to be modified (1-38, q (discard changes), or e (make changes)):
9
Which accounts? (names or numbers, use commas, ? for list) [0] jones
You selected accounts:
jones , 624
Are these correct? (y or n) [y] Entry is now:
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  age ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624    6)  gids ....... cray (100)  test (104)
trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/jones
   8)  shell ...... /bin/csh
   9)  acids ...... jones (624)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
```

```
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... none
  18)  valcomps ... none
  19)  permits .... none
  20)  intcls...... 0            21)  maxcls...... 0
  22)  intcat...... none         23)  valcat...... none
  24)  disabled ... 0            25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
  28)  pcpulim[b] ...      none       pcpulim[i] ...      none
  29)  jmemlim[b] ...      none       jmemlim[i] ...      none
  30)  pmemlim[b] ...      none       pmemlim[i] ...      none
  31)  pfilelim[b] ..      none       pfilelim[i] ..      none
  32)  jsdslim[b] ...   1048576       jsdslim[i] ...   1048576
  33)  psdslim[b] ...   1048576       psdslim[i] ...   1048576
  34)  jtapelim[b][type0   ]      99     jtapelim[i][type0   ]      99
       jtapelim[b][type1   ]      99     jtapelim[i][type1   ]      99
       jtapelim[b][type2   ]      99     jtapelim[i][type2   ]      99
       jtapelim[b][type3   ]      99     jtapelim[i][type3   ]      99
       jtapelim[b][type4   ]      99     jtapelim[i][type4   ]      99
       jtapelim[b][type5   ]      99     jtapelim[i][type5   ]      99
       jtapelim[b][type6   ]      99     jtapelim[i][type6   ]      99
       jtapelim[b][type7   ]      99     jtapelim[i][type7   ]      99
  35)  jpelimit[b] ...      none       jpelimit[i] ...      none
  36)  jmpptime[b] ...      none       jmpptime[i] ...      none
  37)  jmppbarrier[b]       none       jmppbarrier[i]       none
  38)  pmpptime[b] ...      none       pmpptime[i] ...      noneSelect
field to be modified (1-38, q (discard changes), or e (make
changes)): e

Do you want to modify any more ./udb entries? (y or n) [y]n
done.
execing udbgen - please wait
udbgen complete.
```

Procedure: Deleting a user record by using /etc/nu

Caution

Deleting a user from the system requires more prudence than adding a user to the system because you may be removing valuable data from the system.

Before removing the user from the UDB, you should determine whether any pertinent files are needed from the account. If files are needed, you can disable the user account by setting the PERMBITS_RESTRICTED bit in permbits to prevent the user from running. Setting PERMBITS_NOBATCH prevents batch jobs from running, and setting PERMBITS_NOIACTIVE prevents interactive jobs from running.

To remove a user account completely, follow these basic steps:

1. Save any important files the user owned on the system. You may want to back up these files to tape or have someone in the deleted user's department copy necessary files to another directory.

   Example:

```
# rsv
# tpmnt -l nl -p /tmp/tapedev -v vsn -b 4096
# ls -a /usr/trng/jones | cpio -o > /tmp/tapedev
# rls -a
```

2. Disable the user's entry from /etc/udb by using the /etc/nu -d command, as follows:

```
# /etc/nu -d
```

   You will be prompted to enter the login name or UID of the user you want to disable.

Note

If you want to keep accounting records in order or if you want to ensure that the user ID is not reused, you may choose not to complete step 3.

3. Remove the user from the UDB files by using the `/etc/nu -k` command, as follows. This command removes files under the user's login directory and that directory removes the user's mailbox and accounting records:

```
# /etc/nu -k jones
```

Example `/etc/nu` session that disables and removes a user's login:

The following is an example `/etc/nu` session that disables and then removes user `jones` from the system:

```
# /etc/nu -d
cmd/nu/nu.c                >>> Deletion mode <<<Enter user identifier
(login or uid) [quit]: jones
Entry is now:
   1)  name ....... jones
   2)  passwd ..... v0u28k2K1wtX6 (encrypted)
   3)  pwage ........ force
   4)  comment .... John R. Jones
   5)  uid ........ 624
   6)  gids ....... cray (100)  test (104)  trng (105)  usrsrc (98)
   7)  dir ........ /hot/u1/jones
   8)  shell ...... /bin/csh
   9)  acids ...... jones (624)
  10)  root ....... /
  11)  resgrp ..... Users (102)
  12)  permbits ... none
  13)  shares ..... 100
  14)  deflvl ..... 0
  15)  maxlvl ..... 0
  16)  minlvl...... 0
  17)  defcomps ... none
  18)  valcomps ... none
  19)  permits .... none
  20)  intcls...... 0            21)  maxcls...... 0
  22)  intcat...... none           23)  valcat...... none
  24)  disabled ... 0            25)  trap ....... 0
  26)  jproclim[b] ..       100       jproclim[i] ..       100
  27)  jcpulim[b] ...      none       jcpulim[i] ...      none
```

```
  28)  pcpulim[b] ...       none        pcpulim[i] ...       none
  29)  jmemlim[b] ...       none        jmemlim[i] ...       none
  30)  pmemlim[b] ...       none        pmemlim[i] ...       none
  31)  pfilelim[b] ..       none        pfilelim[i] ..       none
  32)  jsdslim[b] ...    1048576        jsdslim[i] ...    1048576
  33)  psdslim[b] ...    1048576        psdslim[i] ...    1048576
  34)  jtapelim[b][type0  ]     99     jtapelim[i][type0   ]        99
       jtapelim[b][type1  ]     99     jtapelim[i][type1   ]        99
       jtapelim[b][type2  ]     99     jtapelim[i][type2   ]        99
       jtapelim[b][type3  ]     99     jtapelim[i][type3   ]        99
       jtapelim[b][type4  ]     99     jtapelim[i][type4   ]        99
       jtapelim[b][type5  ]     99     jtapelim[i][type5   ]        99
       jtapelim[b][type6  ]     99     jtapelim[i][type6   ]        99
       jtapelim[b][type7  ]     99     jtapelim[i][type7   ]        99
  35)  jpelimit[b] ...       none        jpelimit[i] ...       none
  36)  jmpptime[b] ...       none        jmpptime[i] ...       none
  37)  jmppbarrier[b]        none        jmppbarrier[i]        none
  38)  pmpptime[b] ...       none        pmpptime[i] ...       none
Do you want to delete this entry? (y or n) [y] y

Entry for user jones has been deleted.
Do you want to delete any more users? (y or n) [y] n
execing udbgen - please wait
udbgen complete.

# nu -k jones
cmd/nu/nu.c     71.7    10/30/92 09:04:35 (hot:./nu.cf60)

User jones is already disabled; no directory deletion done.

Entry for user jones has been killed.
execing udbgen - please wait.
udbgen complete.
#
```

Using /etc/udbgen 7.6

The /etc/udbgen utility lets you make changes to the UDB either interactively or as a batch job. The /etc/udbgen utility is actually the program underlying the /etc/nu utility. The batch capability of /etc/udbgen lets you add or modify many accounts at one time. When used with the udbsee command, the udbgen command with its directives can be a powerful and efficient tool in maintaining many accounts.

Note

If you have configured the nu utility to skip prompting for specific UDB fields, you must use udbgen to access these fields.

When using the /etc/udbgen command to create a new user login, you must specify the create directive. You may use the /etc/udbgen program in the following three ways:

- Interactive submission: When using udbgen interactively, you must use the quit directive to exit the utility; this action updates the UDB files.
- Batch submission: You can place directives in a file and submit them all at once to the UDB.
- Individual submission: You can submit directives individually.

With all three methods, you can enter multiple UDB field names and their values. You can place more than one field on a create directive line or each field may be on separate lines. The recommended method for udbgen is the batch approach: place the directives in a file and then use that file as input to the /etc/udbgen command.

You may choose, for test purposes, to modify a private copy of the UDB files, rather than the ones contained in the /etc directory; see the example on page 193 of the following procedure.

The format of the `create` directive is as follows:

`create:`*user_name*`:uid:`*uid-number*`:gids:`*group_names*`:field_name:`*field_value*`:`

Field	Description
`create:`	Adds the specified user's information to the UDB; if a UDB record already exists for this user name, a warning message is displayed and the record is not changed.
user_name:	User login name to be created; must be a unique name within the database.
`uid:`*uid_number*:	Specifies a `uid` value or `next` to have `udbgen` assign a value.
`gids:`*group_names*:	Specifies one or more group IDs or names.
`field_name:`*field_value*:	One or more field values; you can put multiple field values on one line, or you can put each field on a separate line.

Note

You **must** include the colon at the end of the directive.

The remainder of this subsection provides the following:

- Procedure for adding users by using the `/etc/udbgen` utility
- Procedure for transferring initial files to the login directory when using the `/etc/udbgen` utility
- Procedure for updating user logins in the UDB by using the `/etc/udbgen` utility
- Procedure for deleting users from the UDB by using the `/etc/udbgen` utility

Procedure: Adding users to /etc/udb by using /etc/udbgen

Important

Before you begin this procedure, make sure you have completed the following:

- Determined the UDB field settings for the user account.
- Added needed group(s) to `/etc/group` to which the user will belong.
- Added needed account(s) that will be associated with the user if you have accounting implementation on your system.

For details on these procedures, see pages 155 through 163. Also, you may choose, for test purposes, to modify a private copy of the UDB files, rather than the ones contained in the `/etc` directory; to do this, see the example at the end of this procedure.

1. Complete **one** of the following three methods of using `/etc/udbgen` to add a user to your system:

   - Create a file of `/etc/udbgen` directives that has this format; you must include the colon at the end of the line:

     ```
     create:username:uid:uid
     number:gids:group_membership_list:field_name:field_value:
     field_name:field_value:
     field_name:field_value:
     ```

     Then submit the changes to the `/etc/udbgen` database by typing the following command line:

     ```
     # /etc/udbgen udbgen_directives_filename
     ```

Example of directives submitted in batch file (**bold** indicates what you type):

```
# vi udb.source
(Enter udbgen directives.)
# cat udb.source
create:john:uid:next:
comment:John Smith:
pwage:force:
gids:cray,test,trng:
acids:testing,training:
dir:/user1/trng/john:
shell:/bin/csh:
resgrp:102:
psdslim[b]:1000000:
pmemlim[i]:8000:
psdslim[i]:1000000:
shares:100:
# udbgen udb.source
Input style: udb
Added 1 record
#
```

or

- Type `/etc/udbgen` to enter interactive mode and reply to the prompt that has the following format; you must include the colon at the end of the line:

```
create:username:uid:uid number:gids:group_membership_list:field_name:field_value:
quit
```

or

- If you must make only one or two changes, you can submit directives to the database individually by typing a line that has the following format:

```
/etc/udbgen -c "create:username:uid:uid number:field_name:field_value:"
```

Example of directives submitted individually (**bold** indicates what you would type):

```
# /etc/udbgen -c "create:john:shell:/bin/sh:uid:next:gids:cray,test,trng:"
# /etc/udbgen -c "update:john:resgrp:102:"
# /etc/udbgen -c "delete:mary:"
```

2. Verify that your entry was added by using the udbsee command.

3. Assign the initial password for the user.

   If you use the udbgen command, you cannot set the password field to an initial password. Instead, you must use the /bin/passwd command to change the password. You and the new user must agree on an initial password for the account. Choose one that is not easy to guess. Only the superuser may change another user's password. You may have chosen to set the UDB passwd field to * or left it empty (indicated by two colons in succession, ::). If no assignment was made to this field during the user's login creation, the field is assigned the symbol *.

Caution

If you have left the password field empty, anyone who knows the login can use this account. Your system is open to abuse.

Example:

```
# /etc/udbgen  -c "create:john:uid:next:gids:cray,test,trng:"
# /bin/passwd john
New password:                    (The password is not visible on your screen.)
Reenter new password:
#
```

4. Create a login directory for the user.

   The /etc/udbgen command does not automatically create a login (home) directory. The dir for each entry in /etc/udb specifies the initial working directory (home) for each user at login time. As the system administrator, you must create that directory by using /bin/mkdir. Because you currently have a user ID of 0 and a group ID of 0, the directory created also will be assigned these permissions. You must make the user's directory accessible to the user by changing the permissions, group, and ownership of the directory. This involves executing the chmod, chgrp, and chown commands.

   The following is a brief review of how UNICOS permissions are defined (see pages 190 through 192 for examples).

   Format:

```
/etc/chmod permissions filename
```

Permissions are set up in three groups, and they can be displayed by using the `ls -l` command:

	User	Group	Other
-	rwx	rwx	rwx
d	rwx	rwx	rwx

The - symbol indicates that the file is a regular file. The `d` indicates that the file is a directory file. The `r`, `w`, and `x` indicate permissions for read, write, and execute, respectively. If the `r`, `w`, or `x` is present, that permission is set for that category of users (user, group, or other). If a - symbol is in any of the fields, except for the first field, that permission is turned off for that category of users. You can represent these fields numerically, as follows:

400, 40, and 4 = Readable by user, group, and other, respectively.

200, 20, and 2 = Writable by user, group, and other, respectively.

100, 10, and 1 = Executable by user, group, and other, respectively.

Example:

To give user, group, and others read, write, and execute permissions, calculate the permission fields to use with the `chmod` command:

user = read + write + execute

400 + 200 + 100 = 700

group = read + write

40 + 20 = 60

other = execute

1 = 1

```
# /etc/chmod 761 /user1/trng/jones
```

Also see the following examples.

Example of creating a login directory:

1. Create the directory by using the `/bin/mkdir` command.

   Format: `/bin/mkdir` *new_login_directory_name*

   Example:
   ```
   # mkdir /user1/trng/jones
   ```

2. Change the ownership of the directory by using the `/bin/chown` command.

   Format: `/bin/chown` *new_login_name new_login_directory_name*

   Example: `# /bin/chown jon /user1/trng/jones`

3. Change the group of the directory by using the `/bin/chgrp` command.

   Format: `/bin/chgrp` *new_group new_login_directory_name*

   Example: `# /bin/chgrp swtng /user1/trng/jones`

4. Change the permissions of the directory by using the `/bin/chmod` command.

   Format: `/bin/chmod` *permissions new_login_directory_name*

   Example: `# /bin/chmod 761 /user1/trng/jones`

Note

If you want to move existing files into the login directory, use the procedure to transfer initial files to the login directory (see page 195).

Examples of /bin/chown, /bin/chgrp, and /bin/chmod follow:

```
sn1601% pwd
/sn1601/sdiv/unicos/jones%
su root
Password:
# ls -la
total 21
drwx------   3 jones   os          4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root        4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-r--r--r--   1 root    root         192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt

# /bin/chown jones mnt
# ls -la
total 21
drwx------   3 jones   os          4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root        4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-r--r--r--   1 jones   root         192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt

# /bin/chgrp tng mnt
# ls -la
total 21
drwx------   3 jones   os          4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root        4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-r--r--r--   1 jones   tng          192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt

# /bin/chmod 761 mnt
# ls -la
total 21
drwx------   3 jones   os          4096 Mar 21 17:43 .
drwxr-xr-x 100 root    root        4096 Mar 24 13:14 ..
-rw-r--r--   1 jones   os           121 Sep 13  1991 .cshrc
-rwxrw---x   1 jones   tng          192 Oct 11 17:28 mnt
-rw---x--x   1 root    os           82 Oct 11 17:31 umnt
```

Example of using a private copy of UDB files for test purposes:

You may choose, for test purposes, to modify a private copy of the UDB files, rather than the ones contained in the `/etc` directory. After you have set up a private UDB, use the `-p` option with the `/etc/udbgen` command, as follows:

1. Create a directory to contain your private UDB, as follows:

```
# mkdir/myudb
```

2. Create a `group` file in your new directory, as follows:

```
# cp /etc/group ./myudb/group
```

3. Create an `acid` file in your new directory, as follows:

```
# cp /etc/acid ./myudb/acid
```

To verify that the user login was created correctly, use the `udbsee` command. Then move or copy the UDB files contained in the directory specified by the `-p` option into the `/etc` directory, as shown in the following example.

Example of directives submitted interactively (**bold** indicates what you would type):

```
# /etc/udbgen -p /user1/jones/etc
/etc/udbgen: 1>create:john:uid:next:
/etc/udbgen: 2>comment:John Smith:
/etc/udbgen: 3>pwage:force:
/etc/udbgen: 4>gids:cray,test,trng:
/etc/udbgen: 5>acids:testing,training:
/etc/udbgen: 6>dir:/user1/trng/john:
/etc/udbgen: 7>shell:/bin/csh:
/etc/udbgen: 8>resgrp:102:
/etc/udbgen: 9>psdslim[b]:1000000:
/etc/udbgen: 10>pmemlim[i]:8000:
/etc/udbgen: 11>shares:100:
/etc/udbgen: 12>quit
Added 1 record
#
```

Procedure: Transferring initial files to the login directory when using /etc/udbgen

1. You may want to create a directory, such as /usr/skel, to hold templates for such files as .profile, .cshrc, .login, and .exrc. The /etc/udbgen command does not automatically copy skeleton files to the user's home directory.

   For descriptions of how to set up the /etc/profile and /etc/cshrc files, see subsection 7.7, page 203.

2. After you have created /usr/skel and the template files, copy the desired files to the user's home directory by using the cpset command, which lets you specify the mode, owner, and group of the destination files. cpset installs object files in binary directories.

   Example:

```
# cpset /usr/skel/.cshrc /usr/home/john 700 john trng
# cpset /usr/skel/.login /usr/home/john 700 john trng
```

Procedure: Updating user logins in the UDB by using `/etc/udbgen`

The method for updating user logins is similar to that for adding new user logins. Different directives are used, however, to accomplish the task. Some of the fields (such as the `gids` field) have editing suffixes that may be used with the `/etc/udbgen` command. These editing suffixes are as follows:

= Indicates that the next value(s) replace the existing value.

\+ Indicates that the following values will be appended to the current values of the field (see example 1, page 198).

– Indicates that the following values will be deleted from the list of current values for the field.

You may use the following steps to change every field except the `passwd` field. To change the `passwd` field, use the `/bin/passwd` command, as described in step 3 of the procedure for adding users to `/etc/udb` by using `/etc/udbgen` (see the `passwd` man page for further information). You change the user login name by deleting the old user login and creating a new user login under the new name.

Caution

It is a **very dangerous** practice to delete users who may be logged in. This procedure should wait until a time when you know the user is not running anything on the system.

The following steps summarize the user login update process:

1. Decide which UDB fields you want to change.
2. If the user will be placed in a new group that you will reference by name, make the desired entry in `/etc/group`.
3. If the user will be placed in a new account group that you will reference by name, make the desired entry in `/etc/acid`.
4. Make the desired entry in `/etc/udb` by using the `/etc/udbgen -c` command with the `update` directive.
5. If you change the user's login directory, create the new directory and copy over any existing files to the new directory.

Examples of updating user logins by using udbgen:

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

Example 1. Adding a new group ID

This example adds the new group ID (`gids`) `usrsrc` to user `john` :

```
# /etc/udbgen -c "update:john:gids+:usrsrc:"
```

Example 2. Changing the user's shell

This example changes the login shell for user `john` to the POSIX shell. Because the POSIX shell was specified, you also must create a `.profile` file.

```
# /etc/udbgen -c "update:john:shell:/bin/sh:"
# cpset /usr/skel/.profile /user1/trng/john
```

Example 3. Changing the user's login directory

This example changes the login directory for user `john` to `/usr1/john`. Formerly, user `john`'s login directory was `/user1/trng/john`. The `mkdir`, `chown`, `chgrp`, and `chmod` commands are used to create the `/usr1/john` directory and to assign proper ownership and permissions for the directory. The last three commands remove `john`'s old login directory.

Caution

If the user is running anything on the system, you should never change a home directory. This is especially critical if libraries are removed.

```
# /etc/udbgen -c "update:john:dir:/user1/john:"
# mkdir /user1/john
# chown john /user1/john
# chgrp trng /user1/john
# chmod 700 /user1/john
# cd /user1/trng/john
# find . -print | cpio -pdm /user1/john
# rm -rf /user1/trng/john
```

Example 4. Using the udbsee command as a filter to add an account ID (acid)

This example uses the udbsee and udbgen commands to add an account ID (acid) of 10 to all user accounts that have a group ID (gid) of 103. In all, 46 login accounts are affected. This is a typical example of how a large-scale update to the UDB is performed:

```
# udbsee -a -e 'gids ~ "103"' -f "name" -m 'update:%s:acids+:10:\n'|/etc/udbgen
46 entries converted to source
Input style: udb
Updated 46 records
#
```

Example 5. Changing the user's password

This example uses the /bin/passwd command to change the password for user john:

```
# /bin/passwd john
New password:              (The password is not visible on your screen.)
Reenter new password:
```

Procedure: Deleting a user from the UDB by using `/etc/udbgen`

Caution

Deleting a user from the system requires more prudence than adding a user to the system, because you may remove valuable data from the system. Before removing a user from the UDB, you should determine whether any pertinent files are needed from the account. If files are needed, you can disable the user account by setting the *passwd* field to *****, using the `udbgen` command.

It is a **very dangerous** practice to delete users who may be logged in. This procedure should wait until a time when you know the user is not running anything on the system.

To remove a user account completely, perform the following basic steps:

1. Make it impossible for the user to log in by using the `udbgen` command, as follows:

```
# /etc/udbgen -c "update:john:passwd:*:"
```

2. Ensure that the user has nothing running on the system.
3. Save any important files the user owned on the system. You may want to back up these files to tape or have someone in the deleted user's department copy necessary files to another directory.

   Example:

```
# rsv
# tpmnt -l nl -p /tmp/tapedev -v vsn -b 4096
# ls -a /usr/trng/john | cpio -o > /tmp/tapedev
# rls -a
```

4. Delete files from the user's home directory and any other directories on the system by using multiple `rm` commands, and remove the user's `home` directory. Also remove the user's mailbox, `/usr/mail/`*username*.

   Example:

```
# rm -rf /user1/trng/john
# find / -user john -exec rm {}\;
# rm -f /usr/mail/john
```

Note

If you want to keep accounting records in order or if you want to ensure that the user ID is not reused, you may choose not to complete step 5.

5. Remove the user from the UDB files by using the `delete` directive of the `udbgen` command. The `delete` directive has the `delete:`*userid*: format; you must include the colon at the end of the directive:

```
# /etc/udbgen -c "delete:john:"
```

Maintaining user environment files
7.7

This subsection describes the following procedures:

- Setting up an `/etc/profile` file
- Setting up an `/etc/cshrc` file
- Transferring users to another file system

When the user logs in to the system, the `/bin/login` script executes the program in the UDB `shell` field. If you specify `/bin/sh` or `/bin/rsh`, the following files are executed (if they exist) by `/bin/sh` or `/bin/rsh`:

```
/etc/profile
$HOME/.profile
```

If you specify `/bin/csh`, the program executes the following files in the order shown:

```
/etc/cshrc
$HOME/.cshrc
$HOME/.login
```

As the administrator, you must maintain the `/etc/profile` and `/etc/cshrc` files, which are described in this subsection.

Procedure: Setting up an `/etc/profile` file

When the `/bin/login` script invokes the default shell (`/bin/sh`, which is the POSIX shell, or `/bin/rsh`, which is the restricted shell), it reads and executes the commands found in the `/etc/profile` file. This lets you set up a standard environment for all users. Users may alter this set up environment through the `$HOME/.profile` file to personalize their environment.

Note

If you want to change the `.profile` file, see the `sh`(1) man page, which describes the supported shells and the `shell` script syntax.

A typical system profile, `/etc/profile`, might perform the following functions (the line references refer to the example that follows):

1. Set and export the directory search path (lines 4 and 5).
2. Set the file creation mask, using the `umask` command (line 6).
3. If using one of these shells (line 8), do the following functions:
   - Display the message of the day (line10).
   - If the `.motd` file exists, display it (lines 11 through 13).
   - Check for the existence of mail (lines 15 through 17).
   - Display the names of current news items (lines 18 through 20).
   - Set the user's prompt (lines 21 through 25).
   - Set effective user ID if user uses the `/bin/su` command (line 27).

An example `/etc/profile` file follows:

```
 1#       SCMID@(#) /etc/profile
 2
 3 trap "" 1 2 3
 4 PATH=/bin:/usr/bin:/usr/ucb:/usr/lbin
 5 export PATH
 6 umask 022
 7 case "$0" in
 8 -sh | -rsh | -ksh)
 9      trap : 1 2 3
10      cat /etc/motd
11      if [ -f ../.motd ] ; then
12              cat ../.motd
13      fi
14      trap "" 1 2 3
15      if mail -e ; then
16           echo "You have mail."
17      fi
18      if [ -x /usr/bin/news -a -d /usr/news ] ; then
19              news -n
20      fi
21      if id | grep 'uid=0' > /dev/null ; then
22              PS1="`uname -n`# "
23      else
24              PS1="`uname -n`$ "
25      fi
26      ;;
27  -su)
28      :
29      ;;
30  esac
31  trap 1 2 3
```

Procedure: Setting up an /etc/cshrc file

If a user has chosen to run the C shell (/bin/csh), the commands found in /etc/cshrc are executed. You should set up the same environment variables found in /etc/profile in the C shell start-up file.

Note

If you want to change the .profile file, see the csh(1) man page, which describes the shell command syntax.

Users can alter this setup environment through the $HOME/.cshrc and $HOME/.login files to personalize their environment. A typical system profile, /etc/cshrc, might perform the following functions (the line references refer to the example that follows):

1. Set and export the shell variable (line 3).
2. Set and export the directory search path (line 4).
3. Start up the history mechanism (line 5).
4. Set the file creation mask, using the umask command (line 6).
5. Display the message of the day (line 7).
6. Check for the existence of mail (lines 8 and 9).
7. Display the names of current news items (lines 10 through 12).
8. Set the user's prompt (line 13).

An example /etc/cshrc file follows:

```
 1  # SCMID@(#) etc/cshrc
 2
 3  setenv SHELL /bin/csh
 4  set path = ( /bin /usr/bin /usr/ucb /usr/lbin /usr/ucb)
 5  set history = 24
 6  umask 022
 7  cat /etc/motd
 8  mail -e
 9  if ( $status == 0 ) echo "You have mail."
10  if (-d /usr/news) then
11    news -n
12  endif
13  set prompt = "'uname -n'$prompt"
```

Procedure: Transferring user accounts to another file system

If a group of users must be transferred to another file system, use the `cpio` command to copy them. If all users are copied at the same time, the `cpio` command helps preserve any links the users had among their files.

Caution

Be sure to update the user's home directory in the UDB. When you do this, also ensure that none of the users are running.

Example:

```
# cd /user_a
# find john tom sue mike alice -print | cpio -pdm /user_b
# rm -rf john tom sue mike alice
```

Communicating with Users [8]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

As a system administrator, you must communicate with your users frequently. Several methods of communication are available for you to use. The method to use in any specific instance generally is determined by the urgency of your message.

The following list describes the types of communication you will maintain with users, as well as the commands associated with that kind of communication:

Type of communication	Command
Issuing emergency messages only	`/etc/wall`
Issuing critical messages	`/etc/issue`
Issuing special messages (message of the day)	`/etc/motd`
Issuing normal (noncritical) communication to all users	`/usr/news`
Communicating with specific users	`write` and `mail`

This section describes when you should use each type of communication and gives examples of each.

Related user communication documentation
8.1

The following documentation contains detailed information covered in this section:

- *UNICOS User Commands Reference Manual*, publication SR–2011: `mail`(1), `news`(1), `su`(1), `wall`(8), and `write`(1) man pages
- *UNICOS File Formats and Special Files Reference Manual*, publication SR–2014: `issue`(5) and `motd`(5) man pages

Issuing emergency messages only
8.2

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

To run the command, type the `/etc/wall` command. The `wall` command responds by telling you to type your message and to press CONTROL-d when you are finished.

To ensure that all users who are currently logged in see a message sent by `wall`, run the command while you have `root` privileges; otherwise, the message goes only to users who allow messages to be written to their terminals (see `mesg`(1)).

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

Typically, the `wall` command is used to send the following messages:

- Warnings that the system will soon be brought down for scheduled downtime. Users who log in after the message is sent, however, miss the message and should be notified by the `/etc/issue` file (see the `login`(1) man page).
- Warnings that the system must be brought down immediately to address a system emergency.
- Warnings that a particular file system has run out of disk space and that users should make an immediate effort to delete any unneeded files (see the description of the `-g` option on the `wall`(8) man page).

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

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

Issuing critical messages

8.3

The `/etc/issue` file is displayed while a user is logging in before he or she has successfully logged in to the system. It is an ordinary text file, and you may place messages in it by using any UNICOS text editor. All interactive users can determine from the message whether they want to log in to the system.

Messages placed in `/etc/issue` should be brief and so important that users may need the information to decide whether to log in to the system. Possible messages include the following:

- Warnings that the system will be brought down soon (so that users who do not see a `wall` message are not surprised when the system is brought down shortly after they log in)
- Warnings that the system is being used for dedicated time and that not all users can log in

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

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

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

To delete an issue message, type the following command:

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

Issuing special messages (message of the day)
8.4

The `/etc/motd` (message-of-the-day) file is displayed to users after they are logged in to the system. The `/etc/motd` file is an ordinary text file, and you may place messages in it by using any UNICOS text editor.

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

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

---

Note

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

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

```
# vi /etc/motd

============================================================
This machine will be brought down at 5:30 pm today for
maintenance work.  It should be down for only 1 hour.
============================================================
```

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

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

Issuing noncritical communication to all users

8.5

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

Note

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

Because users are not notified of the existence of a new news file until the next time they log in, and because there is no guarantee that any given user will see the file (a user may choose to ignore the item by not running the news command), /usr/news is appropriate for items that are not time-sensitive or items that are of interest to only some of the system's users. These categories include the following:

- Notices about recent system changes, such as a newly installed version of a command or library
- Explanations of imminent system reconfigurations or changes
- Explanations of recent system problems and their possible effects on users

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

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

The following options are commonly used with the news command:

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

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

The following is an example of creating a news item:

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

                        John Doe  (jd@cray)
```

To execute a news item, type the following command:

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

To delete a news item, type the following command:

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

Using the `write` command

8.6

The `write` command initiates immediate person-to-person communication with a user who is logged in by opening that user's tty or pty for writing and copying each line of text you type to his or her screen. To write to a user who has a login name of `dolores`, for example, you would issue the following command:

```
# write dolores
```

If user `dolores` happened to be logged in on more than one tty or pty, you could specify the connection:

```
# write dolores ttyp001
```

If, in this example, user `dolores` is currently logged in, a message appears on her screen indicating that you are writing to her. Typically, user `dolores` replies by writing back to your account; each line of text she types appears on your screen.

Given the immediate nature of its communication, `write` allows you to perform the following functions:

- Converse with a user
- Obtain information about what a user is doing
- Warn a specific user to stop what he or she is doing
- Instruct a specific user to clean up his or her directories

Because each typed line appears on the other user's terminal without regard for what that person may be typing at the moment, it is easy for the other user's messages to your terminal to appear to interfere with your typing. This problem is customarily solved by having the two users take turns typing, ending a message with an `o` on a line by itself (standing for "over," much as in a two-way radio conversation). To end such a session, either user then ends a message with an `oo` on a line by itself (for "over and out"). Thus, a typical "conversation" carried out by using the `write` command might look like this (your input appears in **bold**):

```
# write dolores
Message from dolores (ttyp001) - Mon May 11 08:20:15 - ...
Yes
o
Please clean up your account, we are out of space.
o
All right, I will.
o
Thank you.
oo
<EOT>
```

Because many users either do not know of this etiquette when using `write`, or do not follow it, they think that `write` is difficult to use. In practice, it is used rather sparingly, mainly when more convenient forms of communication (such as simply calling the user on the telephone) are impossible. Taking steps to educate your user community in the proper use of the `write` command will prove valuable when `write` is the appropriate communication method.

Using the `mail` command

8.7

The `mail` command provides a way to leave messages for specific users, whether or not they are currently logged in to the system.

The `mail` command is used as shown in the following example:

```
mail ralph
Type in message
<CONTROL-d>
```

You may specify more than one account name; in which case, copies of the message go to each user specified. The next time users to whom you (or anyone else) have sent mail messages log in to the system, the system alerts them to the fact that they have mail messages waiting. The `mail` command is thus particularly well suited for messages such as the following:

- Instructions to clean up directories
- Asking or responding to questions
- General communication

In theory, there is no guarantee that the recipient of a mail message will actually see the message, because the recipient may choose not to run the `mail` command to read the message; however, in practice, most users read their mail when they log in.

For more information, see the `mail`(1) and `mailx`(1) man pages.

Log Files [9]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

This section describes several log files that are important for you to monitor. Information found in these files can help you determine appropriate actions. You can access the logs described in this section through normal file manipulation commands such as `tail`, `cat`, `pg`, and `more`.

This section describes the following:

- The `/etc/boot.log` file
- The `/etc/rc.log` file
- The `/etc/syslog.conf` file and the `syslog` daemon, `/etc/syslogd`, which works with the `/etc/syslog.conf` file to record entries into the following system log files:
  - `/etc/dump.log`
  - `/usr/adm/sulog`
  - `/usr/adm/nu.log`
  - `/usr/adm/sa/sa`*DD*
  - `/usr/adm/sl/slogfile`
  - `/usr/spool/msg/msglog.log`
  - `/usr/lib/cron/cronlog`
  - `/usr/tmp/nqs.log`
  - `/usr/adm/errfile`
  - `/usr/spool/dm`
- Cleaning up system logs

For information about accounting logs and reports, see section 10, page 237.

Related log files documentation
9.1

The following documentation contains information that you will find useful in understanding the material presented in this section:

- *UNICOS Administrator Commands Reference Manual,* publication SR–2022: `brc`(8), `cron`(8), `newsys`(8), `reduce`(8), `sar`(8), and `syslogd`(8) man pages
- *CRAY IOS-V Commands Reference Manual,* publication SR–2170
- *CRAY EL Series IOS Commands Reference Manual,* publication SR–2408: `errpt`(8) man page
- *UNICOS User Commands Reference Manual*, publication SR–2011: `at`(1), `batch`(1), `cat`(1), `crontab`(1), `date`(1), `logger`(1), `more`(1), `sar`(1), `syslogd`(8), `tail`(1), and `uname`(1) man pages

/etc/boot.log file
9.2

The `/etc/boot.log` file records boot dates and times for a system. When the `/etc/rc` script is executed, it appends a record to the `/etc/boot.log` file. The file is composed of output from the `/bin/date` and `uname -a` commands. The format of the `/etc/boot.log` file includes the system name, node, release, version, and hardware information. To determine the last time a system was booted, see this log. The format is as follows:

```
date, uname -a yy/mm/dd hh:mm system node release version hardware
```

Example:

```
# cat /etc/boot.log
93/09/10 08:07 sn1703c cool 8.0.2 CRAY Y-MP
```

For further information, see the `date`(1) and `uname`(1) man pages.

`/etc/rc.log` file
9.3

The `/etc/rc.log` file records the events that occurred the last time the `/etc/rc` (multiuser startup) script was run.

`/etc/syslog.conf` file
9.4

The `syslog` configuration file, `/etc/syslog.conf`, defines the messages that are processed and where they are recorded. An example of the `/etc/syslog.conf` file follows (for a description of the fields, see the `syslogd`(8) and `syslog`(3) man pages):

```
# (Messages processed)                          (Stored location)
#
*.debug                                         /usr/adm/syslog/debug
#
mail.debug                                      /usr/spool/mqueue/syslog
#
kern.debug                                      /usr/adm/syslog/kern
#
daemon,auth.debug                               /usr/adm/syslog/auth
#
*.err;kern.debug;daemon,auth.notice;            /usr/adm/syslog/daylog
#
*.alert;kern.err;daemon.err                     operator
#
*.alert                                         root
```

System logs
9.5

The `syslog` daemon, `/etc/syslogd`, provides UNICOS with the ability to route messages to regular disk files or to forward them to mail accounts. The `/etc/syslogd` daemon reads and logs messages into a set of files specified by the administrator in the `/etc/syslog.conf` configuration file. `/etc/syslogd` configures itself at start-up time and when it encounters a hang-up signal. The `/etc/newsys` shell script starts it.

The `/usr/ucb/logger` command places entries into the system log. For example, if you restart a daemon in the middle of the day, you can log this event by using the following command:

```
# /usr/ucb/logger -p user.info restarted development copy of syslog daemon
```

This subsection includes information about the following topics:

- Message sources
- Priority levels
- `syslog` daemon startup
- System log files

Message sources 9.5.1

Messages may be given to the `syslog` daemon, `/etc/syslogd`, from the following sources or facilities:

Source/ Facility	Description
`auth`	Messages that the authorization system (that is, `login`, `su`, or `getty`) generates.
`daemon`	Messages that system daemons (such as `telnetd`, `ftpd`, and `errdaemon`) generate.
`kern`	Messages that the kernel generates. The daemon reads kernel messages from the `/dev/klog` device.
`local0`	Reserved for local use (`local0` through `local7` are available).
`mail`	Messages that the mail system generates.
`mark`	Informational-level messages are sent; default interval is every 20 minutes (set by the `syslogd -m` command).
`user`	Messages that user processes generate. Users write messages (see the `logger(1)` man page) to the named pipe `/dev/log`.

Priority levels 9.5.2

The following eight priority levels are defined for messages that the system log daemon handles; they are listed below in order of highest to lowest priority:

Priority	Description
`emerg`	Panic condition, which is usually broadcast to all users
`alert`	Condition that you should correct immediately
`crit`	Critical condition
`err`	Errors
`warning`	Warning message
`notice`	Condition that is not an error condition, but possibly should be specially handled
`info`	Informational message
`debug`	Message useful only when debugging a program

`syslog` *daemon startup* 9.5.3

The `/etc/newsys` shell script starts `/etc/syslogd` and renames any existing log files. As released, the `/etc/newsys` shell script saves the 10 most recent copies of the log files and deletes the oldest. To preserve more or fewer log files, adjust this limit by editing the `/etc/newsys` shell script. Two shell functions, `quantity()` and `time_based()`, control the preservation of old log files, which are saved under the `/usr/adm/syslog/oldlogs` directory. A description of the `quantity()` and `time_based()` shell functions follows, followed by examples of their use and examples of the `/usr/adm/syslog` and the `/usr/adm/syslog/oldlogs` files.

Function	Description
`quantity()`	Preserves the specified quantity of the specified log files. `quantity()` is called with at least two arguments. The first is the number of copies to keep. The remaining arguments are the names of the files to be preserved. It will retain *x* copies of each file and delete the oldest.
`time_based()`	Preserves old log files on the basis of time, rather than system restarts. `time_based()` is passed at least two arguments. The first is the number of days to preserve files. The remaining arguments are the names of the actual files.

Note

If the base name of the log file consists of more than 6 characters, `time_based()` will not work. The pattern match in `find` will fail.

Examples:

```
#
#  Save 20 copies of daylog and debug
#
quantity 20 daylog debug
#
#  Save the last 30 days worth of kern and auth
#
time_based 30 kern auth
```

Examples of system log files follow:

```
# cd /usr/adm/syslog
# ls -lF
total 44

-rw-r--r--  1 root            0 Nov  9 11:03 auth
-rw-r--r--  1 root        15465 Nov  9 15:52 daylog
-rw-r--r--  1 root        15465 Nov  9 15:52 kern
drwxr-xr-x  2 root        11232 Nov  9 11:03 oldlogs/
```

```
# /usr/adm/syslog 5=> tail kern

Nov  9 15:42:39  unicos: NFS server sn218 not responding, giving up
Nov  9 15:42:39  unicos: NFS fsstat failed for server sn218: TIMED OUT
Nov  9 15:42:40  unicos: NFS server sn218 not responding, giving up
Nov  9 15:42:40  unicos: NFS getattr failed for server sn218: TIMED OUT
```

```
# /usr/adm/syslog 6=> cd oldlogs
# /usr/adm/syslog/oldlogs 7=> ls -CF

10-09.5.kern  10-17.6.kern    10-22.3.kern    10-29.1.kern
11-04.1.kern
10-10.0.auth  10-17.7.auth    10-23.0.auth    10-29.2.auth
11-04.2.auth
10-10.0.kern  10-17.7.kern    10-23.0.kern    10-29.2.kern
11-04.2.kern
10-11.0.auth  10-17.8.auth    10-23.1.auth    10-29.3.auth
11-04.3.auth
10-11.0.kern  10-17.8.kern    10-23.1.kern    10-29.3.kern
11-04.3.kern
10-11.1.auth  10-17.9.auth    10-23.2.auth    10-30.0.auth
11-05.0.auth...
10-16.0.auth   10-21.0.auth   10-27.0.auth
11-02.6.auth
.
.
.
10-16.0.auth   10-21.0.auth   10-27.0.auth    11-02.6.auth    daylog.0
10-16.0.kern   10-21.0.kern   10-27.0.kern    11-02.6.kern    daylog.1
10-16.1.auth   10-21.1.auth   10-27.1.auth    11-03.0.auth    daylog.10
10-16.1.kern   10-21.1.kern   10-27.1.kern    11-03.0.kern    daylog.11
```

/usr/adm/sulog
9.5.4

The `/usr/adm/sulog` file contains a line of information for every attempted use of the `/bin/su` command since this version of the file was started. The line indicates whether the attempt was successful. You could monitor this log for attempted system breaching or other malicious use of a system. `root` should own this file, with no `read` or `write` permissions for others. The format of the log is as follows:

SU *MM/DD hh.mm flag tty olduser-newuser*

In the following sample `/usr/adm/sulog` file, the entry that contains a minus sign (line 6) indicates an unsuccessful attempt to use the `/bin/su` command:

```
# cat /usr/adm/sulog
SU 03/11 07:00 + console root-adm
SU 03/11 07:59 + ttyp000 guest-root
SU 03/11 08:13 + ttyp001 jones-root
SU 03/11 11:14 + ttyp002 jones-root
SU 03/11 11:33 + ttyp001 smith-root
SU 03/11 12:26 - ttyp001 smith-root
SU 03/11 12:26 + ttyp001 smith-root
```

/etc/dump.log
9.5.5

The `/etc/dump.log` file contains the time and reason for each system dump. The system supplies the time and the user supplies the reason. By default, the dump is located in `/etc/dump.log` and can be accessed using the normal file manipulations, such as `tail`, `cat`, and `more`. When the system is changing out of single-user mode, `brc` calls `coredd` to copy a dump file to a file system. To reconfigure the location of the dump, use the menu system. To change the location of this log file, use the `cpdmp -l` command.

Note

This is a system dump log. It is **not** the log created by the `dump` utility (which is the `/etc/dumpdates` file).

An example of an `/etc/dump.log` follows:

```
# cat /etc/dump.log

cpdmp: 035120 blocks on dump device - waiting to be copied
01/26/94 07:27:09   coredd: Copying system dump into /core//04260727.
UNICOS dump copied to=/core//04260727/dump
   dump taken: 04/26/93 at 07:18:51
   reason: PANIC: master.s: EEX interrupt in UNICOS kernel
```

/usr/adm/nu.log
9.5.6

The new user log contains information about new user accounts on the system that are created by using `/etc/nu`. It includes entries about who created the account and the time it was added, information about the default environment settings, and the IDs. The `/etc/nu` command creates this file (for further information about `/etc/nu`, see section 7, page 149).

The following types of user account transactions are recorded into `/usr/adm/nu.log`: `changed`, `added`, `deleted`, and `destroyed`.

An excerpt from a `nu.log` file follows:

```
Text goes  here
# cat /usr/adm/nu.log
jones:co:L B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
        changed to
jones:co:Lauren B. Jones
jones:ui:840:di:/home/sis/jones:sh:/bin/csh:dr:/
jones:gi:178
jones:ai:0
jones:rg:178:as:100
jones:dc:none:cm:none:pm:none
jones:ic:none:vc:none
jones:pj[b]:100:pj[i]:100
jones:tp:type0[b]:3:tp:type0[i]:3:tp:type1[b]:3:tp:type1[i]:3
jones:tp:type2[b]:3:tp:type2[i]:3:tp:type3[b]:3:tp:type3[i]:3
        by jones on Mon Sept  13 11:51:00 1993
```

`/usr/adm/sa/saDD`
9.5.7

The `sar` command uses this data collection file to report system activity. The `/usr/lib/sa/sadc` and `/usr/lib/sa/sa1` commands write data to this file; they must be scheduled by `cron` to run at frequent intervals (such as, every 15 minutes).

The `usr/adm/sa/saDD` file is too large and too varied to show a representative example. It is filled with multiple types of reports, each with many different output fields.

For more information about system activity reporting, see the `sar`(1) and `sar`(8) man pages and *UNICOS System Administration*, publication SG–2113.

/usr/adm/sl/slogfile
9.5.8

This data file records UNICOS multilevel security (MLS) event information. The `reduce` command, executable only by the security administrator, reads this data file. The `reduce` command extracts, formats, and outputs entries from UNICOS MLS event files. The MLS event logging daemon, `slogdemon`, collects security-relevant records from the operating system by reading the character special pseudo device `/dev/slog`. An excerpt of the output from the `reduce` command follows:

```
# /etc/reduce -s 04021300 -u jones -p

Apr  2 14:49:21 1993  Validation        o_lvl: 0  s_lvl: 0  jid:0  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]    ********
    Login uid: jones(8863)
  Login to [jones(8863),tng(146)] : Okay   via 128.162.121.20   on /dev/ttyp042
Apr  2 14:49:21 1993  Setuid Syscall    o_lvl: 0  s_lvl: 0  jid:1255  pid:17183
    r_ids:[jones(8863),tng(146)]   e_ids:[jones(8863),tng(146)]          ********
    Login uid: jones(8863)
   Setuid call from root (0) to jones (8863) was successful
:
:
```

/usr/spool/msg/msglog.log
9.5.9

The `/usr/spool/msg/msglog.log` file contains messages and replies to and from the operator. Following is an excerpt from a `msglog.log` file:

```
# cat /usr/spool/msg/msglog.log

Sep 16 09:51:20  Message    1: WARNING THRESHOLD ON /nasc
Sep 16 09:58:59  Message    2: CRITICAL THRESHOLD ON /nasc
:
:

Jun  9 23:34:02  Message daemon stopped
Jun 10 00:43:15  Message daemon started
Jun 10 04:07:49  Message    1: From ghe:  How are you?
Jun 10 04:08:44  Reply    1: good
:
:
Jun 18  12:41:52  Informative: ********** SYSTEM ACCOUNTING RESTARTED for 0618/*
Jun 18 12:48:21  Informative:  *************** SYSTEM ACCOUNTING COMPLETE Thu J*
:
```

/usr/lib/cron/cronlog
9.5.10

The /usr/lib/cron/cronlog file reports the status of all commands that cron executes, including at, batch, and crontab. When UNICOS is brought to multiuser mode, the old log file is copied to /usr/lib/cron/OLDLOG.

Various types of error messages may be present in the cronlog file, including messages about when cron was started and stages of job execution. The cronlog file has the following format:

```
CMD: command_executed username process_id job_type start_time username process_id
job_type end_time rc=error return code
```

The *job_type* argument can have one of the following values:

a = at(1) job

b = Batch job

c = cron(8) job

An example of `/usr/lib/cron/cronlog` follows:

```
! *** cron started ***   pid = 3654 Thu Sep 16 17:47:44 1993
! new user (ce) with a crontab Thu Sep 16 17:47:45 1993
! new user (nfs) with a crontab Thu Sep 16 17:47:45 1993
! new user (root) with a crontab Thu Sep 16 17:47:46 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 3687 c Thu Sep 16 17:48:01 1993
<  root 3687 c Thu Sep 16 17:48:02 1993
>  CMD: /usr/lib/acct/ckpacct
>  root 4291 c Thu Sep 16 18:00:01 1993
>  CMD: /usr/lib/sa/sa1 600 1
>  root 4292 c Thu Sep 16 18:00:01 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 4293 c Thu Sep 16 18:00:01 1993
<  root 4293 c Thu Sep 16 18:00:02 1993
<  root 4292 c Thu Sep 16 18:00:02 1993
<  root 4291 c Thu Sep 16 18:00:04 1993
>  CMD: $HOME/scripts/runsequence cpuseq b
>  ce 4731 c Thu Sep 16 19:30:01 1993
>  CMD:       date >>/home/swts/geir/60564.cron/date.log
>  root 4732 c Thu Sep 16 19:30:01 1993
<  root 4732 c Thu Sep 16 19:30:01 1993
<  ce 4731 c Thu Sep 16 19:30:12 1993
```

`/usr/tmp/nqs.log`
9.5.11

The NQS log, created by the NQS log daemon, contains NQS activity. Its default location is the ASCII file `/usr/spool/nqs/log` (to change the location of the log file, use the `qmgr set log_file` command; to see where the current log file resides, use the `qmgr show parameters` command). Access to `/usr/spool/nqs` is restricted; however, if you have the correct permissions, you can access the NQS log file by using normal file manipulations, such as `tail`, `cat`, and `more`. If you experience problems with NQS, use a `tail -f` command on this file to observe what NQS is doing.

A sample nqs.log file follows:

```
# cat /usr/tmp/nqs.log

NQS(INFO): local mid = 130
I$nqs_boot(): TZ=CST6CDT
NQS(DEBUG): tra_read():0, pid 4033,  state=0, sequence#=0, tid=0
NQS(DEBUG): gen_shrpri_tree(): completed setudb.
NQS(INFO):  gen_shrpri_tree(): Fair Share turned off, Share_wt sched factor set.
NQS(DEBUG): gen_shrpri_tree(): Sh_Decay_usage =  0.0000, Sh_Run_rate =  1.0000
NQS(DEBUG): gen_shrpri_tree(): Share_basis & SHAREBYACCT = 8
NQS(DEBUG): gen_shrpri_tree(): childcnt = 1, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 2, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 3, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 4, st[0].childsum = 0
NQS(DEBUG): gen_shrpri_tree(): childcnt = 5, st[0].childsum = 0
NQS(INFO):  nqs_ldconf(): i = 1NQS(INFO): nqs_ldconf(): Pipe queue gale; Dest count: 1
NQS(INFO):  nqs_ldconf(): Creating new destination 0NQS(INFO):  nqs_ldconf(): batch
NQS(INFO):  nqs_upd.c(): Adding new destn batch to head of queue
NQS(INFO):  upd_addquedes(): Updating queue gale destinations
NQS(INFO):  upd_addquedes(): Destination 0;  832NQS(INFO):  nqs_ldconf(): i = 1
NQS(INFO):  upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO):  upp_setlogfil(): Set/Reset command - $$/usr/spool/nqs/log
NQS(INFO):  netdaemon(): Listening on TCP/IP port: nqs
NQS(INFO):  nqs_rbuild(): Set flag for first time thru spawn.
NQS(INFO):  nqs_boot(): BOOTDONE, Database present.
NQS(INFO):  upp_setchkpntdir(): New directory = /usr/spool/nqs/private/root/chkt
NQS(INFO):  upp_setlogfil(): Logfilename - /usr/spool/nqs/log
NQS(INFO):  upp_setlogfil(): Set/Reset command - #$/usr/spool/nqs/log
NQS(INFO):  upp_setsnapfil(): New pathname = /home/swts/cjd/nqs_snapfile
```

/usr/adm/errfile
9.5.12

The error log is a binary file that contains error records from the operating system. errpt processes error reports from the data. The /etc/errdemon command (see the errdemon(8) man page) reads /dev/error and places the error records from the operating system into either the specified file, or errfile, by default. The /etc/rc (see the brc(8) man page) script starts /etc/errdemon, and /etc/mverr starts a new errfile.

/usr/spool/dm/*
9.5.13

If UNICOS Data Migration Facility (DMF) software is configured on your system, the /usr/spool/dm/dmdlog.*YYMMDD* files record activities that pertain to data migration.

A sample /dm/dmdlog.*YYMMDD* file follows:

```
# cat /usr/spool/dm/dmdlog.930912

        dmdlog.930912

10:55:29 Data Migration daemon 35745 initializing, release level 6100
10:55:29 0 index entries in database
10:55:29 Command request pipe initialized, fd = 7
10:55:29 Kernel request pipe initialized, fd = 8
10:55:29 initmsp: msp fake, pid = 35751, wt_fd = 10, rd_fd = 11
10:55:29 machine id set to 2158163973
10:55:29 First available handle for assignment is 2158163973:1
10:55:30 0 incomplete MSP entries found
10:55:30 0 soft-deleted premigration files found
   .
   .
   .
10:56:35 Counts - permdel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - retrybu,     0,     0,     0,     0,  0,  0
10:56:35 Counts - krecall,    10,    20,    20,     0,  0,  1
10:56:35 Counts - kremove,    28,    28,    28,     0,  0,  1
10:56:35 Counts - kcancel,     0,     0,     0,     0,  0,  0
10:56:35 Counts - invalid,     0,     0,     0,     0,  0,  0
10:56:35 Counts -  pclear,     0,     0,     0,     0,  0,  0
10:56:35 Current mem  = 94437
10:56:35 Stopping daemon processing
10:56:35 Data migration daemon stopped, exit=0
```

Note

The following log files also exist for each file system under data migration:

- `.dmloght`, which is a log file that the `dmhit` command generates.
- `.dmlogct`, which is a log file that the `dmmctl` command generates.
- `.dmlogsm`, which is a log file that the `dmfree` command generates.

Cleaning up system logs
9.6

Some log files are recycled during each reboot, some logs accumulate content slowly and must be cleaned up only occasionally, and some log files accumulate content quickly and should be monitored and cleaned up frequently. This subsection describes each group of log files.

Log files recycled during each reboot
9.6.1

The following log files are recycled during each reboot; therefore, you do not have to monitor them for space consumption. If any one of the log files must be saved, however, you should copy them to a location of your choice before shutting down the system. In case you forget their location, most of the log files are linked to `/usr/spool/ccflogs`.

Log files that recycle are as follows:

- `/etc/rc.log` (log file from `init 2` function)
- `/usr/adm/sulog` (including all `su` records)
- `/usr/spool/msg/msglog.log` (messages and replies from and to an operator)
- `/usr/lib/cron/log` (all `cron` entries since reboot)
- `/usr/tmp/nqs.log` (all NQS entries)

Small accumulative log files 9.6.2

The following log files accumulate content slowly, but you should clean them up occasionally so that they do not consume space needlessly:

- `/etc/boot.log` (records boot dates and times for a system)
- `etc/dump.log` (records crash dump dates and dump file locations)
- `usr/adm/nu.log` (records all `/etc/nu` output)

Large accumulative log files 9.6.3

The system activity report (`sar`) data and report log files accumulate content quickly; therefore, you should monitor these and clean them up frequently. If not managed promptly, these log files could potentially saturate the `/usr` file system. All `sar` data is saved up to 31 days in `/usr/adm/sa/sa`*DD*. At the end of each month, you should dump them to a file server or to tape; otherwise, newer collected data will overwrite them. The `sar` reports (stored in `/usr/adm/sa/sar`*DD*) are kept only up to 7 days, because the reports usually are not backed up. To change the number of days you want to keep `sar` data or `sar` reports, modify `/usr/lib/sa/sa2`.

You also should monitor the following log files:

- Email log file
- User mail files if not read and cleared
- NQS log files (these can grow quickly)
- `errpt` files when active disk errors or tape activity exists
- MLS log files, which are located in `/usr/adm/sl`

Accounting [10]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

UNICOS provides two types of system accounting, standard UNIX System V accounting or Cray system accounting (CSA). You may use one or the other of these accounting packages at your site. To help you decide which accounting package to use, see subsection 10.3, page 240, which describes the unique features of CSA.

This section describes CSA, which is the more complete and frequently used of the two accounting types. It includes the following:

- An overview of CSA, including unique CSA features, descriptions of directories and files, and the `/usr/lib/acct/csarun` primary daily accounting shell script.
- Procedures to follow so that you can set up CSA and execute daily accounting procedures that result in the generation of a variety of reports.

Your accounting configuration file is located in `/etc/config/acct_config`. A sample file is provided at the end of this section; the sample file may differ slightly from the one included with your system.

For information on using standard UNIX System V accounting, see *UNICOS System Administration*, publication SG–2113.

Related accounting documentation
10.1

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

- *UNICOS System Administration*, publication SG–2113, "Accounting" section
- *UNICOS Administrator's Commands Reference Manual,* publication SR–2022: `acctdusg`(8), `chargefee`(8), `csaboots`(8), `csabuild`(8), `csacon`(8), `csacrep`(8), `csadrep`(8), `csaedit`(8), `csajrep`(8), `csaline`(8), `csanqs`(8), `csapacct`(8), `csaperiod`(8), `csarecy`(8), `csarun`(8), `csaverify`(8), `devacct`(8), `diskusg`(8), `dodisk`(8), `nulladm`(8), `runacct`(8), and `setacid`(8) man pages
- *UNICOS User Commands Reference Manual,* publication SR–2011: `acctcom`(1), `ja`(1), `last`(1B), and `who`(1) man pages

Concepts and terminology
10.2

The following concepts and terms are important in CSA:

Term	Description
Daily accounting	Unlike the standard daily accounting, CSA's accounting can be run as many times as necessary during a day. However, this feature is still referred to as daily accounting.
Periodic accounting	Accounting similar to the standard UNICOS monthly accounting. CSA, however, lets system administrators specify the time periods for which "monthly" or cumulative accounting will be run. Thus, periodic accounting can be run more than once a month.

Term	Description
Recycled data	By default, accounting data for active sessions is recycled until the session terminates. CSA reports data for only terminated sessions, unless you invoke the `csarun` command with the `-A` option. `csarun` places recycled data into data files in the `/usr/adm/acct/day` directory. These data files are suffixed with `0`; for example, per-process accounting data for active sessions from previous accounting periods is in the `/usr/adm/acct/day/pacct0` file. For information about recycled data, see *UNICOS System Administration*, publication SG–2113.
Session	CSA organizes accounting data by sessions and boot times, and then it places the data into a session record file. For non-NQS jobs, a *session* consists of all accounting data for a given job ID during one boot period. A *session* for an NQS job consists of the accounting data for all job IDs associated with the job's NQS sequence number/machine name identifier. NQS jobs may span multiple boot periods. If a job is restarted, it has the same job ID associated with it during all boot periods in which it runs. Rerun NQS jobs have multiple job IDs. CSA treats all phases of an NQS job as being in the same session.
Uptime/boot period	A period delineated by the system boot times found in `/etc/csainfo`. The `csaboots` command writes to this file during system boot.

Unique features of CSA
10.3

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

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

Note

The system overhead for device accounting is fairly low. However, the amount of accounting data produced in the worst cases is more than double that produced by standard accounting. The more device accounting data is kept, the more file system space is required. If one device is accounted for, processes that use that device generate twice as much accounting data as a process that did not use the device or the same process without device accounting. However, for 1 to `NODEVACCT` device types, the maximum size of the accounting data does not increase, except that more processes may use one of the devices.

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

- Capability to perform additional site-specific processing during daily accounting.
- Error recovery and automatic repairing of file systems.

For detailed information on these facilities, see *UNICOS System Administration*, publication SG–2113.

Accounting directories and files
10.4

This subsection provides a brief overview of the CSA file and directory structure. The following directories apply to both UNIX System V and CSA and are the main accounting directories: for a more complete description of the files and directories, see *UNICOS System Administration*, publication SG–2113.

Note

Consider configuring the `/usr/adm` directory as another file system so that if `/usr/adm` fills up, other directories (such as `/usr/mail`) are still usable.

The `/tmp` directory also is used while the `csarun` script is running. (For information about the `/tmp` directory, see section 5, page 57.)

Directory	Description
`/usr/lib/acct`	Contains all of the programs and scripts used to run either CSA or System V UNIX accounting. (For a complete list of programs and scripts, see *UNICOS System Administration*, publication SG–2113.) The only CSA program not located here is `/etc/csaboots` (see the `csaboots`(8) man page), which records boot times at system startup. Programs used only by CSA begin with the characters `csa`. This directory also may contain the `csa.archive1`, `csa.archive2`, `csa.fef`, and `csa.user` user-exit scripts if you enable them. (To determine whether your UNICOS release level allows these scripts, see *UNICOS System Administration*, publication SG–2113.)
`/usr/adm/acct/day`	Contains current and recycled accounting files for per-process, disk, and daemon accounting: • `dtmp` (disk accounting data) • `nqacct*` (NQS daemon accounting data) • `pacct*` (per-process accounting data) • `tpacct*` (tape daemon accounting data)

Note

Accounting files in `/usr/adm/acct/day` whose names include the suffix `0` contain data from sessions that did not complete during the previous accounting periods.

During CSA data processing, sites may select to archive the raw and/or processed data offline. For a description of how to do this, see "The `csarun` command" subsection of the "Accounting" section in *UNICOS System Administration*, publication SG–2113. By default, all raw data files are deleted after use and are not archived.

Directory	Description
`/usr/adm/acct/fiscal`	Contains periodic files created by `csaperiod` (CSA) or `monacct` (System V). Within this directory, the `rpt/MMDD/hhmm/rprt` file contains a variety of CSA periodic reports.
`/usr/adm/acct/nite`	Contains files that are reused daily by `csarun` (CSA) or the `runacct` (System V) procedure. Contains processing messages and errors (files that have names beginning with `E` and ending with the date and time).
`/usr/adm/acct/sum`	Contains cumulative summary files updated by `csarun` (CSA) or `runacct` (System V). Within this directory, the `rpt/MMDD/hhmm/rprt` file contains a variety of CSA daily reports.
`/usr/adm/acct/work`	Contains temporary files used by daily accounting procedures.

The following are the main basic accounting files:

File	Description
`/etc/config/acct_config`	Accounting configuration file; contains the configurable parameters used by the accounting commands.
`/etc/csainfo`	Contains system boot time, written to this file by `/etc/csaboots`.
`/etc/wtmp`	Contains login and logout records for users. Records tty, process ID, type of process, and connect-time accounting data. For information about fixing `wtmp` errors, see subsection 10.9, page 253.
`/usr/adm/acct/day/pacct`	Contains data file written by the UNICOS kernel. Source of all process accounting for both CSA and System V accounting.
`/usr/adm/acct/nite/statefile`	Contains the name of the next reentrant state so that the `csarun` accounting script can be restarted at a specified point. For descriptions of CSA states (files that have names beginning with `E` and ending with the date and time contain errors), see subsection 10.8, page 250.

CSA outputs the following six data files:

File	Description
`/tmp/AC.`*`MMDD`*`/`*`hhmm`*`/Super-record`	Session record file; this file usually is deleted after CSA has used it.
`/usr/adm/acct/fiscal/data/`*`MMDD`*`/`*`hhmm`*`/pdacct`	Consolidated periodic data.

File	Description
`/usr/adm/acct/fiscal/data/`*MMDD*`/`*hhmm*`/cms`	Periodic command usage data.
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cacct`	Consolidated daily data; if you specify the `-r` option, `csaperiod` deletes this file.
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/cms`	Daily command usage data; if you specify the `-r` option, `csaperiod` deletes this file.
`/usr/adm/acct/sum/data/`*MMDD*`/`*hhmm*`/dacct`	Daily disk usage data; if you specify the `-r` option, `csaperiod` deletes this file.

Note

Occasionally, sites run on numerous `/` and `/usr` file systems and want to maintain the same accounting files throughout. The easiest way to accomplish this is to put `/usr/adm` or `/usr/adm/acct` on a separate file system and mount this file system along with each different system. You also must copy two other files, `/etc/csainfo` and `/etc/wtmp`, from the previously booted `/`. You must copy these files to the new `/` file system before it is brought up. If you do not copy `/etc/csainfo` to the new `/` correctly, `csarun` may abort abnormally. When `/etc/wtmp` is not copied, incorrect connect time data is reported.

Daily operation overview of CSA
10.5

When UNICOS is run in multiuser mode, accounting behaves as described in the following steps and shown in Figure 1, page 248. However, if you modify CSA to meet your own requirements, the following steps may not reflect the process at your site:

1. `/etc/csaboots`.

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

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

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

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

   c. `turnacct` starts per-process accounting.

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

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

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

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

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

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

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

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

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

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

7. `/usr/lib/acct/csarun` (also see subsection 10.7, page 249)

   a. Executed by `cron` at specified times.

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

   c. Produces accounting reports and a consolidated data file.

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

   a. Executed by `/etc/shutdown`.

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

   c. `turnacct` stops per-process accounting.

   d. `turndacct` stops daemon accounting.

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

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

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

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

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

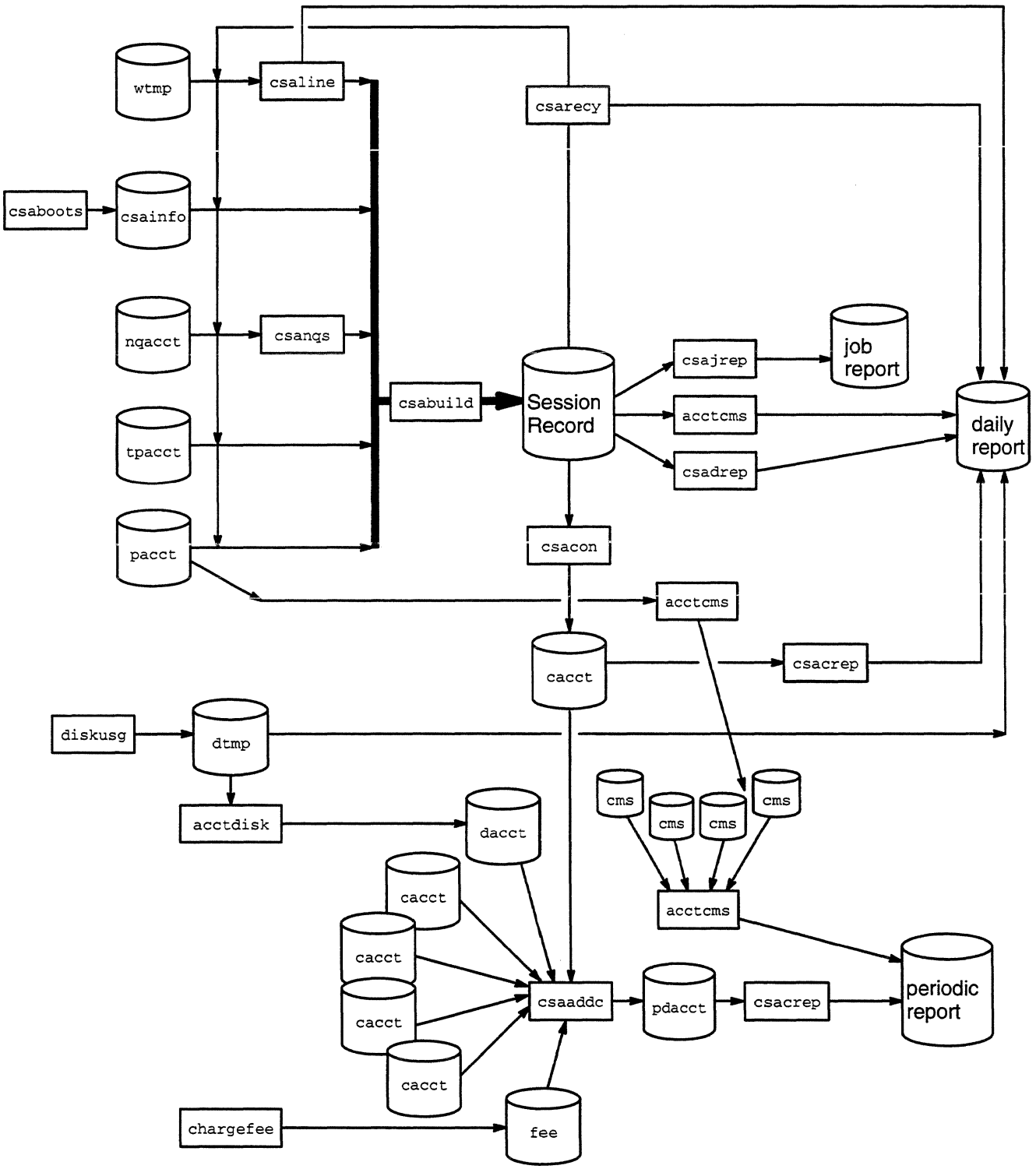


Figure 1. Daily operation overview of CSA

Customizing your system billing procedure
10.6

UNICOS system accounting has been designed to be easy for you to customize to meet your site's requirements. For instance, you may want to change the way the system charges *system billing units* (SBUs) for various kinds of services, to define the weighting factor used in calculating SBUs for various system services (such as tape requests, NQS requests, or connect time), to change the definition of the memory integral to be used in connection with memory charges for multitasking programs, and so on. You may want to set prime time equal to nonprime time and charge based on NQS queue usage. To do so, you must modify the `/etc/config/acct_config` file (a sample file is included beginning on page 264 of this section). By default, all SBUs are set to 0.0. Read the commented instructions throughout the file to determine which lines of the `acct_config` file you want to modify for your site's needs. You also can use your local SBU calculations by modifying the default algorithms defined in `/usr/src/cmd/acct/lib/acct/user_sbu.c`, compiling, and relinking the accounting programs.

The csarun command
10.7

The `/usr/lib/acct/csarun` command (see the `csarun`(8) man page) is the primary daily accounting shell script. It processes connect, per-process, and daemon accounting files and usually is initiated by `cron` during nonprime hours. `csarun` also contains four user-exit points, allowing you to tailor the daily run of accounting to your specific needs (for information on setting up user exits callable from `csarun` and callable from `runacct`, see *UNICOS System Administration*, publication SG–2113).

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

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

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

Note

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

The csarun error and status messages are placed in the /usr/adm/acct/nite directory. The progress of a run is tracked by writing descriptive messages to the active file. Diagnostic output during the execution of csarun is written to fd2log. The lock and lock1 files prevent concurrent invocations of csarun; csarun aborts if these two files exist when it is invoked. The clastdate file contains the month, day, and time of the last two executions of csarun.

Errors and warning messages from programs called by csarun are written to files that have names that begin with E and end with the current date and time. For example, Ebld.11121400 is an error file from csabuild for a csarun invocation on November 12, at 14:00.

If csarun detects an error, it sends an informational message to the operator by using msgi(1), sends mail to root and adm, removes the locks, saves the diagnostic files, and terminates execution. When csarun detects an error, it will send mail either to MAIL_LIST if it is a fatal error, or to WMAIL_LIST if it is a warning message, as defined in the /etc/config/acct_config configuration file.

CSA accounting states

10.8

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

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

Example:

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

`csarun 0415 0400 CMS`
Restarts at the specified time and at the CMS state.

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

The following is a list of CSA accounting states:

State	Description
ARCHIVE1	First user exit of the `csarun` script. You can use this script to archive the raw and preprocessed accounting files. The shell . command executes the `/usr/lib/acct/csa.archive1` script, if it exists. By default, this feature is disabled.
ARCHIVE2	Second user exit of the `csarun` script. You can use this script to archive the session record file. The shell . command executes the `/usr/lib/acct/csa.archive2` script, if it exists. By default, this feature is disabled.
BUILD	The per-process, NQS, tape, and connect accounting data is organized into a session record file.
CLEANUP	Cleans up temporary files, removes the locks, and then exits.
CMS	Produces a command summary file in `cacct.h` format. The `cacct` file is put into the `/usr/adm/acct/sum/data/`*MMDD/hhmm* directory for use by `csaperiod`.

State	Description
DREP	Generates a daemon usage report based on the session file. This report is appended to the daily accounting report, /usr/adm/acct/sum/rpt/*MMDD*/*hhmm*/rprt.
FEF	Third user exit of the csarun script. You can use this script to execute a front-end formatter. The shell . command executes the script /usr/lib/acct/csa.fef, if it exists.
PREPROC	NQS and connect time (wtmp) accounting files are run through preprocessors. File names of preprocessed files are prefixed with a P in the /usr/adm/acct/work/*MMDD*/*hhmm* directory.
REPORT	Generates the daily accounting report and puts it into /usr/adm/acct/sum/rpt/*MMDD*/*hhmm*/rprt. A consolidated data file, /usr/adm/acct/sum/data/*MMDD*/*hhmm*/cacct, also is produced from the session record file. In addition, accounting data for unfinished sessions is recycled.
SETUP	The current accounting files are switched by using the turnacct and turndacct files. These files are then moved to the /usr/adm/acct/work/*MMDD*/*hhmm* directory. File names are prefaced with W. The /etc/wtmp and /etc/csainfo files also are moved to this directory.
USEREXIT	Fourth user exit of the csarun script. You can use this script to execute local accounting programs. The shell . command executes the /usr/lib/acct/csa.user script, if it exists.

State	Description
WTMPFIX	The wtmpfix(8) command checks the wtmp file in the work directory for accuracy. Some date changes cause csaline(8) to fail; therefore, wtmpfix tries to adjust the time stamps in the wtmp file if a data change record appears. If wtmpfix cannot fix the wtmp file, you must repair the wtmp file manually, as described in subsection 10.9.
VERIFY	By default, per-process, NQS, and tape accounting files are checked for valid data. Records with data that is not valid are removed. Names of bad data files are prefixed with BAD. in the /usr/adm/acct/work/* directory.

Fixing wtmp errors
10.9

The wtmp files generally cause the highest number of errors in the day-to-day operation of the accounting subsystem. When the date is changed, and the UNICOS system is in multiuser mode, a set of date change records is written into the /etc/wtmp file. When a date change is encountered, the wtmpfix (see the fwtmp(8) man page) program adjusts the time stamps in the wtmp records.

Some combinations of date changes and reboots, however, slip by wtmpfix and cause csaline to fail. The following steps show how to repair a wtmp file:

```
$ cd /usr/adm/acct/work/MMDD/hhmm
$ /usr/lib/acct/fwtmp < Wwtmp > xwtmp
$ ed xwtmp
   (delete corrupted records)
$ /usr/lib/acct/fwtmp -ic < xwtmp > Wwtmp
   (restart csarun at the wtmpfix state)
```

If the wtmp file is beyond repair, create a null Wwtmp file. This prevents any charging of connect time.

Verifying data files
10.10

To verify data files, use the `csaedit`, `csapacct`, and `csaverify` commands. `csaedit` and `csapacct` verify and delete bad data records; `csaverify` only flags bad records. By default, `csaedit` and `csaverify` are invoked in `csarun` to verify the data files.

These commands may allow files that contain bad data, such as very large values, to be verified successfully.

Editing data files
10.11

You can use the `csaedit` and `csapacct` commands to verify and remove records from various accounting files. The following example shows how you can use `csapacct` to verify and remove bad records from a per-process (`pacct`) accounting file.

In this example, `csapacct` is invoked with verbose mode enabled (valid data records are written to the `pacct.NEW` file):

```
$ /usr/lib/acct/csapacct -v pacct pacct.NEW
```

The output produced by this command line is as follows:

```
Bad record - starting byte offset is 077740 (32736)
   invalid pacct record - bad base parent process id 97867
Found the next magic word at byte offset 0100130, ignored 120 bytes
Found 394 BASE records
Found 4 EOJ records
Found 1 MTASK (multitasking) records
Found 0 ERROR records
Found 0 IO records
Found 0 SDS records           # not on CRAY-2, J90, or EL systems
Found 0 MPP records           # not on CRAY-2, J90, or EL systems
Found 0 PERFORMANCE records   # not on CRAY-2 systems
Outputted records for 398 processes
Ignored 120 bytes from the input file
```

You can use `csaedit` and `csapacct` in conjunction with `csaverify`, by first running `csaverify` and noting the byte offsets of the first bad record. Next, execute `csaedit` or `csapacct` and remove the record at the specified offset. The following example shows how you can verify and then edit a bad `pacct` accounting file:

1. Verify the `pacct` file by using the following command line; the following output is received:

```
$ /usr/lib/acct/csaverify -P pacct

/usr/lib/acct/csaverify: pacct: invalid pacct record - bad base parent process id
    97867 byte offset: start = 077740 (32736)  word offset: start = 07774 (4092)
/usr/lib/acct/csaverify: pacct: invalid pacct record - bad magic word 03514000
   byte offset: start = 0100070 (32824)  word offset: start = 010007 (4103)
```

2. Delete the record found at byte offset 32736, as follows (valid records are written to `pacct.NEW`):

```
$ /usr/lib/acct/csapacct -o 32736 pacct pacct.NEW
```

3. Reverify the new `pacct` file to ensure that all the bad records have been deleted, as follows:

```
$ /usr/lib/acct/csaverify -P pacct.NEW
```

You can use `csaedit` to produce an abbreviated ASCII version of some of the daemon accounting files and `acctcom` to generate a similar ASCII version of `pacct` files.

Data recycling
10.12

A system administrator must correctly maintain recycled data to ensure accurate accounting reports. Data recycling allows CSA to bill sessions properly that are active during multiple accounting periods. By default, the `csarun` script reports data only for sessions that terminate during the current accounting period. Through data recycling, CSA preserves data for active sessions until the sessions terminate.

In the `Super-record` file, `csabuild` flags each session as being either active or terminated. `csarecy` reads the `Super-record` file and recycles data for the active sessions. `csacon` consolidates the data for the terminated sessions, which `csaperiod` uses later. `csarun` invokes `csabuild`, `csarecy`, and `csacon`.

The `csarun` command puts recycled data in the `/usr/adm/acct/day` directory. Data files with names suffixed with 0 contain recycled data. For example, `ctime0`, `nqacct0`, `pacct0`, `tpacct0`, `usacct0`, and `uptime0` are generally the recycled data files that are found in `/usr/adm/acct/day`.

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

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

For detailed information about data recycling, see *UNICOS System Administration*, publication SG–2113.

Procedure: Setting up CSA

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

Note

Before you begin this procedure, determine your company's system billing units (SBUs). For detailed information about making site-specific modifications, see "Tailoring CSA" subsection of the "Accounting" section in *UNICOS System Administration*, publication SG–2113.

---

1. Modify configurable accounting parameters manually or by using the menu system. Following are some of the types of changes you might want to make; parameters that pertain to USCP, MPP, and SDS accounting do not apply to CRAY J90 or CRAY EL systems:

   - Setting up SBUs for per-process accounting (`pacct`) data; by default, no SBUs are calculated. You can set several variables (for example, weighting factors are set through over a dozen variables, including `P_BASIC`, `P_TIME`, and `P_STIME`). (For information, see *UNICOS System Administration*, publication SG–2113.)
   - Charging for NQS jobs (`NQS_TERM_`*xxx* variables); also see step 4.
   - Modifying the file system on which `/usr/adm/acct` resides; the default is `/usr` (`ACCT_FS` variable).
   - Working with an alternative accounting configuration file (the `ACCTCONFIG` variable).
   - Compiling a list of users to whom mail is sent when a warning or error is detected. The default is `root` (`/`) and `adm` (`WMAIL_LIST` and `MAIL_LIST` variables).
   - Enabling NQS and tape daemon accounting at system startup (`NQS_START` and `TAPE_START` variables).
   - Changing the minimum number of free blocks (the `MIN_BLKS` variable) needed in `ACCT_FS` to enable accounting or to run `csarun` or `csaperiod`. The default is 500 free blocks.

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

```
Configure System
...Accounting Configuration
```

```
                              Accounting Configuration

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


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

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

2. Accounting is started by default each time `/etc/rc` is invoked. Make sure that `csaboots` is invoked from `rc`, and not from `/etc/inittab` or `/etc/brc`. If necessary, add the following lines to the `/etc/rc` script in the Accounting script section, just before `/usr/lib/acct/startup` script section:

```
# Accounting.
#
if [ -x /etc/csaboots ]; then
        echolog "Writing accounting boot time."
        x /etc/csaboots -v >> $RC_LOG
fi
```

3. Ensure that the following lines are included in `/etc/shutdown` to turn off per-process accounting and daemon accounting before the system is brought down:

```
if [ -x /usr/lib/acct/shutacct ]
then
        /usr/lib/acct/shutacct
        echo "Process accounting stopped."
fi
```

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

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

NQS_START and TAPE_START parameters to on to enable NQS and online tapes, respectively.

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

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

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

or

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

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

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

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

Generally, `root` executes `dodisk` through `cron` (see step 6). `csarun` incorporates the disk usage data into the daily accounting report.

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

Warning

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

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

If you want to run periodic accounting, ensure that an entry similar to the following is included in `/usr/spool/cron/crontabs/root`; this command generates a periodic report on all consolidated data files found in `/usr/adm/acct/sum/data/*` and then deletes those data files:

```
15 5 1 * * /usr/lib/acct/csaperiod -r 2> /usr/adm/acct/nite/pd2log
```

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

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

```
set-xS
```

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

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

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

```
# USMID @(#)acct/src/acct/holidays
#
#       (c) Copyright Cray Research, Inc.  Unpublished Proprietary Information.
#       All Rights Reserved.
#
# Prime/Nonprime Table for UNICOS Accounting System
#
# Curr  Prime   Non-Prime
# Year  Start   Start
#
  199x  0730    1730    (Accounting software references this line to confirm current year)
#
# Day of        Calendar        Company
# Year          Date            Holiday
#
    1           Jan 1           New Year's Day
  146           May 25          Memorial Day
  184           Jul 2           Independence Day Thursday
  185           Jul 3           Independence Day Friday
  186           Jul 4           Independence Day
  251           Sep 7           Labor Day
  331           Nov 26          Thanksgiving Day
  332           Nov 27          Thanksgiving Friday
  359           Dec 24          Christmas Eve
  360           Dec 25          Christmas Day
  366           Dec 31          New Year's Eve day
```

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

Daily CSA reports
10.13

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

- Interactive connect time by ttyp.
- CPU usage by user ID and account ID. For a description of the fields in this report, see the "Accounting" section of *UNICOS System Administration*, publication SG–2113,.
- A listing of active interactive and batch jobs by job ID.
- Disk usage by user ID and account ID.
- Command summary data by total CPU time used. For a description of the fields in this report, see the "Accounting" section of *UNICOS System Administration*, publication SG–2113.
- Last interactive login information by date.
- Job mix (interactive versus NQS), tape, and NQS usage.

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

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

```
#
#    SN5228 - acct_config - Edition 40  [Mon Jan  3 14:34:41 CST 1994]
#    Created by Configuration Generator Rev. 80.60
#
#
#    SN5228 - acct_config - Edition 22  [Mon Nov 15 13:15:11 CST 1993]
#    Created by Configuration Generator Rev. 80.60
#
#
#    SN5147 - acct_config - Edition 8  [Mon Jun  7 14:58:30 CDT 1993]
#    Created by Configuration Generator Rev. 80.60
#
# USMID @(#)skl/etc/config/acct_config    70.8    04/20/93 17:17:19
#
#     (C) COPYRIGHT CRAY RESEARCH, INC.
#     UNPUBLISHED PROPRIETARY INFORMATION.
#     ALL RIGHTS RESERVED.
#
#
#     This file contains the parameter labels and values used by the
#     accounting software.
#
##############################################################################
#
#     Connect time SBUs.
#  The following section contains the labels and values which pertain to
#  connect time accounting.
#
##############################################################################
#
# The CON_PRIME parameter defines the weighting factor for prime time
# connect time.  This is in billing units per second.
CON_PRIME    0.0
# The CON_NONPRIME parameter defines the weighting factor for nonprime
# time connect time.  This is in billing units per second.
CON_NONPRIME    0.0
##############################################################################
#
#     Device accounting SBUs (available only on non-Cray2 systems).
#  The following section contains the labels and values which pertain to
```

```
#  device accounting.
#
#
#########################################################################
#
# The System Administrator's Guide (SG-2113) describes device accounting
# configuration in detail.
#
# For each device type to be billed, 3 fields must be filled out.
#   1) Logical I/O sbu  -  unit per logical I/O request.
#   2) Characters xfer sbu  -  unit per character transferred.
#   3) Device name  -  Device type name.  This field must be surrounded by
#                      double quotes if the name contains embedded spaces.
#    Block devices: The name should match the type assigned to
#      the block device.  For example, if the device to be
#      mounted on /tmp is a dd49 with logical device cache,
#      then this device should be labeled as "dd49 with ldcache"
#      or numerically 7.
#    Character devices: The name should match the major device numbers
#      in /usr/src/uts/c1/cf/devsw.c.
#
# Sites may add new types as long as the number assigned to that type does
# not exceed MAXBDEVNO (for block devices) or MAXCDEVNO (for character
# devices).  A site may also change the meaning of the default device types.
#
# MAXBDEVNO and MAXCDEVNO may be increased (see /usr/include/sys/param.h),
# but this will increase the size of the largest possible accounting record.
# This in turn may necessitate more disk space for the pacct files and/or
# a larger kernel stack.
#
#
#    Block device SBUs.
# The numeric suffixes for the "BLOCK_DEVICE" labels must be ascending from
# 0 to MAXBDEVNO - 1.  MAXBDEVNO is currently 10.
#
#             Logical        Characters           Device
#   label     I/O Sbu         Xfer Sbu        Name
BLOCK_DEVICE0    0.0        0.0        "dd29"
BLOCK_DEVICE1    0.0        0.0        "dd39"
BLOCK_DEVICE2    0.0        0.0        "dd40"
BLOCK_DEVICE3    0.0        0.0        "dd49"
```

```
BLOCK_DEVICE4     0.0          0.0          "dd29 with ldcache"
BLOCK_DEVICE5     0.0          0.0          "dd39 with ldcache"
BLOCK_DEVICE6     0.0          0.0          "dd40 with ldcache"
BLOCK_DEVICE7     0.0          0.0          "dd49 with ldcache"
BLOCK_DEVICE8     0.0          0.0          "ssd"
BLOCK_DEVICE9     0.0          0.0          "bmr"
#
#    Character device SBUs.
# The numeric suffixes for the "CHAR_DEVICE" labels must be ascending from
# 0 to MAXCDEVNO - 1.  MAXCDEVNO is currently 35.  The suffixes must match
# the minor numbers  in /dev.  These minor numbers are defined in
# /usr/src/uts/c1/cf/devsw.c in the cdevsw[] array.
#
#              Logical         Characters            Device
#   label      I/O Sbu          Xfer Sbu          Name
CHAR_DEVICE0     0.0          0.0          "hpm"
CHAR_DEVICE1     0.0          0.0          "ios-tty"
CHAR_DEVICE2     0.0          0.0          "systty"
CHAR_DEVICE3     0.0          0.0          "memory"
CHAR_DEVICE4     0.0          0.0          "hyperchannel"
CHAR_DEVICE5     0.0          0.0          "expander tape"
CHAR_DEVICE6     0.0          0.0          "expander printer"
CHAR_DEVICE7     0.0          0.0          "hsx"
CHAR_DEVICE8     0.0          0.0          "error-device"
CHAR_DEVICE9     0.0          0.0          "expander disk"
CHAR_DEVICE10    0.0          0.0          "low speed channel"
CHAR_DEVICE11    0.0          0.0          "bmx tape"
CHAR_DEVICE12    0.0          0.0          "pty-master"
CHAR_DEVICE13    0.0          0.0          "pty"
CHAR_DEVICE14    0.0          0.0          "?"
CHAR_DEVICE15    0.0          0.0          "bmx daemon"
CHAR_DEVICE16    0.0          0.0          "net"
CHAR_DEVICE17    0.0          0.0          "net"
CHAR_DEVICE18    0.0          0.0          "disk control"
CHAR_DEVICE19    0.0          0.0          "secded"
CHAR_DEVICE20    0.0          0.0          "security log"
CHAR_DEVICE21    0.0          0.0          "cpu control"
CHAR_DEVICE22    0.0          0.0          "logger"
CHAR_DEVICE23    0.0          0.0          "disk maintenance"
CHAR_DEVICE24    0.0          0.0          "data migration"
CHAR_DEVICE25    0.0          0.0          "?"
```

```
CHAR_DEVICE26     0.0         0.0         "?"
CHAR_DEVICE27     0.0         0.0         "?"
CHAR_DEVICE28     0.0         0.0         "?"
CHAR_DEVICE29     0.0         0.0         "?"
CHAR_DEVICE30     0.0         0.0         "?"
CHAR_DEVICE31     0.0         0.0         "?"
CHAR_DEVICE32     0.0         0.0         "?"
CHAR_DEVICE33     0.0         0.0         "?"
CHAR_DEVICE34     0.0         0.0         "?"
##############################################################################
#
#    Multitasking CPU time SBUs.
#  The following section contains the labels and values which pertain to
#  multitasking.
##############################################################################
#
# The MUTIME_WEIGHT variables define the weighting factors that
# are used to bill user CPU time for multitasking programs.  It
# is used in conjunction with the ac_mutime array (see
# /usr/include/sys/acct.h), which defines the amount of user
# CPU time the multitasking program spent with i + 1 CPUs connected.
#
# MUTIME_WEIGHTi defines the marginal cost for getting the i-th + 1
# CPU at one instant.  If the MUTIME_WEIGHT values are set to less
# than 1.0, there will be an incentive for multitasking.  If the
# values are set to 1.0, multitasking programs will be charged for
# user CPU time just as all other programs.
#
# There must be an MUTIME_WEIGHT variable for each of the cpus
# available on the machine.
#
MUTIME_WEIGHT0          1.0         # 1 CPU
MUTIME_WEIGHT1          1.0         # 2 CPU
MUTIME_WEIGHT2          1.0         # 3 CPU
MUTIME_WEIGHT3          1.0         # 4 CPU
MUTIME_WEIGHT4          1.0         # 5 CPU
MUTIME_WEIGHT5          1.0         # 6 CPU
MUTIME_WEIGHT6          1.0         # 7 CPU
MUTIME_WEIGHT7          1.0         # 8 CPU
##############################################################################
#
```

```
#    NQS SBUs.
#  The following section contains the labels and values which pertain to
#  NQS accounting.
#
##############################################################################
#
#    Set the values to 1 if jobs, or portion of jobs, which
#    terminate with the specified termination code are to be billed.
#    Otherwise, set the value to 0.  By default, all portions of a
#    request will have sbus calculated for them.
#
NQS_TERM_EXIT        1        # Request exited
NQS_TERM_REQUEUE     1        # Request requeued for a restart
NQS_TERM_PREEMPT     1        # Request preempted
NQS_TERM_HOLD        1        # Request held
NQS_TERM_OPRERUN     1        # Request rerun by operator
NQS_TERM_RERUN       1        # Request non-operator rerun



#
#    Set NQS_NUM_QUEUES to be the number of queues for which you want
#    to set sbus.
#
NQS_NUM_QUEUES       3
#
#    Set the sbus associated with each queues.  There must be
#    NQS_NUM_QUEUES sbu/queue pairs.  The labels' numeric suffixes
#    must be ascending from 0 to NQS_NUM_QUEUES.  Thus, if
#    NQS_NUM_QUEUES is 0, no NQS_QUEUEx values need be defined.
#
#    If an sbu value is set to less than 1.0, there is an incentive
#    to run jobs in this queue.  If the value is set to 1.0, the
#    jobs will be charged as though it were a normal, non-NQS job.
#    If the sbu is set to 0.0, there is no charge for jobs running
#    in this queue.  For queues not listed below, the sbu is set
#    to 1.0.
#
#  label    sbu    queue_name
NQS_QUEUE0    1.0    b_30_5
NQS_QUEUE1    1.0    b_600_1
```

```
NQS_QUEUE2    1.0    b_1200_1
#
#    Set NQS_NUM_MACHINES to the number of originating machines for
#    which you want to set sbus.
#
NQS_NUM_MACHINES        2
#
#    Set the sbus associated with each originating machine.  There must
#    be NQS_NUM_MACHINES sbu/machine pairs.  The sbu values are set
#    in the same manner as those for the queues.  Once again, the
#    numeric label suffixes must be ascending from 0 to NQS_NUM_MACHINES.
#    Thus, if NQS_NUM_MACHINES is 0, no NQS_MACHINEx values need be
#    defined.
#
#  label    sbu    machine_name
NQS_MACHINE0    1.0    sn1405
NQS_MACHINE1    1.0    sn2024
##############################################################################
#
#    Pacct SBUs.
#  The following section contains the labels and values which pertain to
#  pacct (kernel) accounting.
#
##############################################################################
#
#    Set the prime time weighting factors.
#    On non-Cray2 systems:
#       If P_STIME is nonzero, then P_SCTIME and P_INTTIME must be zero.
#       If P_SCTIME and P_INTTIME are nonzero, then P_STIME must be zero.
#       This is so there won't be multiple billing of system cpu time.
#
P_BASIC        0.0    # Basic prime time weighting factor
P_TIME        0.0    # General time weighting factor
P_STIME        0.0    #   System CPU time weighting factor (unit/sec)
P_UTIME        0.0    #   User CPU time weighting factor (unit/sec)
P_ITIME        0.0    #   I/O wait time weighting factor (unit/sec)
#
#    P_SCTIME and P_INTTIME are used only on non-Cray2 systems.
#
P_SCTIME    0.0    #   System call weighting factor (unit/sec)
P_INTTIME    0.0    #   Interrupt time weighting factor (unit/sec)
```

```
P_MEM         0.0    # General memory weighting factor
P_XMEM         0.0    #   CPU time memory weighting factor (unit/Kw-min)
P_IMEM         0.0    #   I/O wait time memory weighting factor (unit/Kw-min)
P_IO         0.0    # General I/O weighting factor
P_BYTEIO     0.0    #   I/O char xfer weighting factor (unit/char xferred)
P_PHYIO        0.0    #   Physical i/o req weighting factor (unit/phy i/o req)
P_LOGIO        0.0    #   Logical i/o req weighting factor (unit/log i/o req)
#
#    The following 3 SDS weighting factors are used only on non-Cray2
#    machines.
#
P_SDSMEM     0.0    # SDS memory integral weighting factor (unit/Mw-sec)
P_SDSLOGIO     0.0    #   SDS logical i/o req weighting factor (unit/log req)
P_SDSBYTEIO     0.0    #   SDS char xferred (unit/char transferred)
#
#    Set the non-prime time weighting factors.
#    On non-Cray2 systems:
#       If NP_STIME is nonzero, then NP_SCTIME and NP_INTTIME must be zero.
#       If NP_SCTIME and NP_INTTIME are nonzero, then NP_STIME must be zero.
#       This is so there won't be multiple billing of system cpu time.
#
NP_BASIC     0.0    # Basic non-prime time weighting factor
NP_TIME        0.0    # General time weighting factor
NP_STIME     0.0    #   System CPU time weighting factor (unit/sec)
NP_UTIME     0.0    #   User CPU time weighting factor (unit/sec)
NP_ITIME     0.0    #   I/O wait time weighting factor (unit/sec)
#
#    NP_SCTIME and NP_INTTIME are used only on non-Cray2 systems.
#
NP_SCTIME     0.0    #   System call weighting factor (unit/sec)
NP_INTTIME     0.0    #   Interrupt time weighting factor (unit/sec)
NP_MEM         0.0    # General memory weighting factor
NP_XMEM         0.0    #   CPU time memory weighting factor (unit/Kw-min)
NP_IMEM         0.0    #   I/O wait time memory weighting factor (unit/Kw-min)
NP_IO         0.0    # General I/O weighting factor
NP_BYTEIO     0.0    #   I/O char xfer weighting factor (unit/char xferred)
NP_PHYIO     0.0    #   Physical i/o req weighting factor (unit/phy i/o req)
NP_LOGIO     0.0    #   Logical i/o req weighting factor (unit/log i/o req)
##############################################################################
#
#    Tape SBUs.
```

```
#  The following section contains the labels and values which pertain to
#  tape accounting.
#
#########################################################################
#
#    The following section sets the sbu values for each of the
#    TP_MAXDEVGRPS tape device groups.  TP_MAXDEVGRPS is defined
#    in /usr/include/acct/dacct.h.  At this time, only 2 device
#    groups which are used: TAPE and CART.  However, there must
#    be TP_MAXDEVGRPS "TAPE_SBU" variables defined.  The TAPE_SBU
#    numeric suffix must be ascending from 0 to TP_MAXDEVGRPS - 1.
#
#    The fields are:
#      Device_group    Device group name
#      Mount        Billing unit per mount
#      Reserve    Billing unit per reserve second
#      Read        Billing unit per byte read
#      Write        Billing unit per byte written
#
#    Note:  On Cray2 systems, TAPE_SBU0 is always for tape devices,
#      and TAPE_SBU1 is always for cart devices.
#
#
#        Device
#        Group    Mount        Reserve        Read        Write
TAPE_SBU0    TAPE    0.0        0.0        0.0        0.0
TAPE_SBU1    CART    0.0        0.0        0.0        0.0
TAPE_SBU2    SILO    0.0        0.0        0.0        0.0
TAPE_SBU3    UNUSED    0.0        0.0        0.0        0.0
TAPE_SBU4    UNUSED  0.0            0.0            0.0            0.0
TAPE_SBU5    UNUSED  0.0            0.0            0.0            0.0
TAPE_SBU6    UNUSED  0.0            0.0            0.0            0.0
TAPE_SBU7    UNUSED  0.0            0.0            0.0            0.0
#########################################################################
#
#    USCP SBUs.
#  The following section contains the labels and values which pertain to
#  USCP accounting.
#
#########################################################################
#
```

```
#
#     The USCP_MAXMF parameter defines the number of mainframes
#     for which sbus are to be set.  This value must be at least 1.
#
USCP_MAXMF     2              (Set this to 0 for CRAY J90 and CRAY EL systems.)
#
#     The following parameters set the sbu values for each of the
#     USCP_MAXMF mainframes.  Sbus must be set for each of the
#     US_MAXTTYPE transfer types.  US_MAXTTYPE is defined in
#     /usr/include/acct/dacct.h.
#
#     Mainframes not listed below are given sbu values of 0.0.
#
#       USCP_MAINFRAME sets the 2 character mainframe identifier.
#
#       The following parameters set the runtime and sectors transferred
#       sbus for each of the various transfer types.  Runtime sbus
#       are in units per second.  Sectors transferred (xfer) sbus
#       are in units per sectors transferred.
#
#       USCP_INTER sets the sbus for interactive disposes and fetches
#         (obsolete).
#       USCP_DISPOSE sets the sbus for disposes.
#       USCP_FETCH sets the sbus for fetches.
#       USCP_GET sets the sbus for gets.
#       USCP_PUT sets the sbus for puts.
#       USCP_SAVE sets the sbus for saves.
#
USCP_MAINFRAME0        SB
#             runtime        xfer
USCP_INTER0         0.0         0.0
USCP_DISPOSE0         0.0         0.0
USCP_FETCH0         0.0         0.0
USCP_GET0         0.0         0.0
USCP_PUT0         0.0         0.0
USCP_SAVE0         0.0         0.0
USCP_MAINFRAME1        YJ
#             runtime        xfer
USCP_INTER1         0.0         0.0
USCP_DISPOSE1         0.0         0.0
USCP_FETCH1         0.0         0.0
```

```
USCP_GET1         0.0         0.0
USCP_PUT1         0.0         0.0
USCP_SAVE1         0.0         0.0
##############################################################################
#
#    Miscellaneous parameters which are sometimes reset.
# The following section contains miscellaneous parameters that can be
# reset by the site.
#
##############################################################################
#
# The ACCT_FS parameter defines the file system on which /usr/adm/acct
# resides.  It is used when checking the amount of free space on /usr/adm/acct.
#
ACCT_FS         /usr
#
# The HOLIDAY_FILE parameter defines the location of the holidays file.
# This parameter should be an absolute pathname.
#
HOLIDAY_FILE    /usr/lib/acct/holidays
#
# The MAIL_LIST parameter is a list of users to whom mail is sent
# if errors are detected in the various shell scripts.
#
MAIL_LIST    "root adm"
#
# The MEMINT parameter is used to select the memory integral.
#
MEMINT         2
#
# The MIN_BLKS parameter sets the minimum number of free blocks on the
# ACCT_FS filesystem that need to be available.  If less than MIN_BLKS
# free blocks is available, accounting is disabled, or processing via
# runacct or csarun is halted.
#
MIN_BLKS    500
#
# The NQS_START parameter enables or disables NQS accounting when
# /usr/lib/acct/startup is executed.  Valid values are "on" and "off".
# If NQS accounting is enabled here, it must also be enabled by NQS
# via the qmgr(8) "set accounting on" command.
```

```
#
NQS_START     on
#
# The NUM_HOLIDAYS parameter sets the upper limit on the number of
# holidays that can be defined in HOLIDAY_FILE.
#
NUM_HOLIDAYS     20
#
# The PERF_NAME0 parameter sets the type which is to be specified with
# devacct(1M) when enabling and disabling performance accounting.
#
PERF_NAME0     perf_01
#
# The TAPE_START parameter enables or disables tape accounting when
# /usr/lib/acct/startup is executed.  Valid values are "on" and "off".
# If tape accounting is enabled here, the "-c" option must be used
# when starting the tape daemon, tpdaemon(8).
#
TAPE_START     on
#
# The USCP_START parameter enables or disables uscp accounting when
# /usr/lib/acct/startup is executed.  Valid values are "on" and "off".
# Uscp accounting does not need to be enabled by the uscp daemon.
#
USCP_START     on      (Set this to off for CRAY J90 and CRAY EL systems.)
##############################################################################
#
#    Miscellaneous parameters generally not reset.
# The following section contains miscellaneous parameters that are not
# generally changed by a site.  Care must be used when some of these are
# modified.
#
##############################################################################
#
# The A_SSIZE parameter is the maximum number of sessions in 1
# accounting run that can be processed by acctprc1(1M).
#
A_SSIZE         10000
#
# The A_TSIZE parameter is the maximum number of tty line names
# in 1 accounting run that can be processed by acctcon1(1M) and
```

```
# csaline(1M).
#
A_TSIZE        1000
#
# The A_USIZE parameter is the maximum number of distinct login
# names in 1 accounting run that can be processed by acctprc1(1M)
# and acctprc2(1M).
#
A_USIZE        5000
#
# The ACCTOFF string is written to /etc/wtmp when accounting is
# turned off by shutacct(1M).  This string should be a maximum
# of 11 characters.
#
ACCTOFF        acctg off
#
# The ACCTON string is written to /etc/wtmp when accounting is
# turned on by startup(1M).  This string should be a maximum of
# 11 characters.
#
ACCTON        acctg on
#
# The BUILD_MAXFILES parameter sets the upper limit on the number
# of files which can be processed by csabuild(1M).
#
BUILD_MAXFILES    200
#
# The MAX_CPUS parameter sets the upper limit on the number of
# cpus a machine can have.  This value must be at least as large
# as the number of cpus on your machine.
#
MAX_CPUS    32
#
# The MAXICYLS parameter sets the upper limit on the number disk
# cylinders involved in the inode region that must be read by the
# Cray2 version of diskusg(1M).  This is an expected worse case
# based on 8000 inode sectors / 16 regions = 512 sectors with the
# dd49's being the smallest at 360 sector/cylinder, requiring 2
# reads per area for an average distribution.
#
MAXICYLS    32
```

```
#
# The MAXILIST parameter sets the maximum number of ilist expected
# in a filesystem.  This parameter is used only the by the Cray2
# version of diskusg(1M).
#
MAXILIST    200
#
# The MAXUSERS_DISK parameter sets the upper limit on the number
# of user id/account id pairs that can be handled by the Cray2
# version of diskusg(1M).
#
MAXUSERS_DISK    5000
# The NCLUSTER parameter sets the upper limit on the number of
# partitions in a filesystem.  This parameter is used only by the
# Cray2 version of diskusg(1M).
#
NCLUSTER    100
# The NSYS parameter sets the upper limit on the number of different
# reasons a wtmp record can be written.  Generally, these records are
# written by acctwtmp(1M) when accounting is turned on or off.
#
NSYS            20
##############################################################################
#
#    User defined labels.
# The following section contains user defined labels and values.
# These labels are used in the site tailored sbu routines.  The
# format of the following lines must be:
#    label        value
#
##############################################################################
```

Adding Your Cray Research System to Your Network [11]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

This section describes the minimal steps you must take to place your CRAY J90 or CRAY EL system on an existing TCP/IP network. After your CRAY J90 or CRAY EL system is a functional member of the network, you can do additional configuration. This section also includes a table that describes some of the most common TCP/IP configuration files.

After you have placed your CRAY J90 or CRAY EL system on an existing TCP/IP network by using the information in this section, refer to *UNICOS System Administration*, publication SG–2113, for additional networking configuration information; the *UNICOS System Administration* publication also includes which menus to use for various networking tasks.

Related network information

11.1

The following publications contain additional information that will be of use to you:

- *UNICOS System Administration*, publication SG–2113

- *UNICOS Administrator Commands Reference Manual,* publication SR–2022 (man pages):

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

- *UNICOS User Commands Reference,* publication SR–2011 (man pages):

ftp(1)	netstat(1)	telnet(1)
hostname(1)		

- *UNICOS File Formats and Special Files Reference,* publication SR–2014 (man pages):

gated-config(5)	lo(4)	services(5)
hosts(5)	networks(5)	shells(5)
hosts.equiv(5)	protocols(5)	
inetd.conf(5)	rhosts(5)	

- The following publications are also recommended, but Cray Research does not provide them:
  - *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture*, Douglas Comer. Prentice Hall, 1991.
  - *UNIX Networking*, S. Kochan and P. Wood. Hayden Books, 1989.
  - *Introduction to the Internet Protocols*, Charles Hedrick, Rutgers University, 1987. Anonymous `ftp` from `cs.rutgers.edu`.
  - *Introduction to the Administration of an Internet-based Local Network*, Charles Hedrick, Rutgers University, 1988. Anonymous `ftp` from `cs.rutgers.edu`.

Procedure: Adding a CRAY J90 or CRAY EL system to an existing TCP/IP network

To place a CRAY J90 or CRAY EL system on an existing TCP/IP network, you should complete the following steps.

Note

To add your CRAY J90 or CRAY EL system to an existing TCP/IP network, you must have several configuration files. The easiest way to create these files is to configure the CRAY J90 or CRAY EL system to talk to another host on the network, copy the necessary files from that machine to your CRAY J90 or CRAY EL system, and then change them. The first five steps of this procedure make the changes to allow you to talk to another host. You then copy files you need and change them for your CRAY J90 or CRAY EL system. This procedure assumes you are either an administrator of your network or that you will have a network administrator as a resource when you add your CRAY J90 or CRAY EL system to your existing TCP/IP network.

1. Create a minimal `/etc/hosts` file.

   The `/etc/hosts` file contains the database of all locally known hosts on the TCP/IP and OSI network. Create an `/etc/hosts` file that contains a local host entry, entries for the CRAY J90 or CRAY EL system, and an entry for at least one other host on the same network as the CRAY J90 or CRAY EL system. The entry format is as follows:

   IPaddress host_name annotations

   Example:

```
# cat /etc/hosts

123.0.0.1      localhost loghost (local host)
456.789.16.8   cray cray-eth (your CRAY J90 or CRAY EL system)
456.789.16.125cyclone cyclone-eth1 (other host)
```

2. Compile a binary hosts file.

   Cray Research systems support a binary `/etc/hosts` file called `/etc/hosts.bin`. Create this file by using the `/etc/mkbinhost` command, as follows:

```
# /etc/mkbinhost
/etc/hosts.bin: 3 entries written
```

3. Update the `/etc/config/interfaces` file.

   The `/etc/config/interfaces` file defines all network interfaces on the Cray Research system. Change the host name for each interface on your system to match those you chose in step 1; for additional information, see the `initif`(8) man page and *UNICOS System Administration*, publication SG–2113. The entry format is as follows (the *hycf_file_name* field is used only if a HYPERchannel connection exists; the HYPERchannel connection is not supported on CRAY J90 systems):

   interface_name hycf_file_name family address ifconfig parameters

   Example:

```
# cat /etc/config/interfaces
...some comment lines omitted...
lo0   -   inet localhost  -
en0   -   inet cray.net   -     netmask 0xffffff00
fddi0  - inet cray.fddi.net-  netmask 0xffffff00
atm0   - inet cray.atm.net -  netmask 0xffffff00
```

4. Activate the changes by executing the `/etc/initif` script; an example follows:

```
# /etc/initif
Configuring all network interfaces: lo0 en0 fddi0 atm0
```

5. Create a default route.

   This step creates a default route to let you communicate with hosts that are on different networks than the Cray Research system. To reach hosts that are not on the same FDDI or Ethernet network as the CRAY J90 or CRAY EL system, you must have a route. To create a route, execute the `/etc/route` command, as shown in the following example:

```
# /etc/route add default otherhost
add net default: gateway otherhost
```

   The *otherhost* is a host that is on the same network as the CRAY J90 or CRAY EL system and connects to one or more additional networks.

   Place this command in the `/etc/tcpstart.mid` script so that it will be run automatically at system startup.

6. Test the network.

   Test the network connections to create a route by using the `ping` command and view the configuration by using the `netstat` command. The `ping` command tests whether or not you can reach another host on the network. If `ping` succeeds, you can be confident that the hardware and routing works on all hosts and gateways between you and the system you are pinging. The `netstat` command has many options. The `-i` option lets you view the list of network interfaces currently initialized (configured up); the `-r` option lets you view the routing table. The `-iv` option provides a table of cumulative statistics for transferred packets, errors, and collisions for each interface that was autoconfigured. (The interfaces that are statically configured into a system but are not located at boot time are not shown.) The network address (currently Internet-specific) of the interface and the maximum transmission unit (mtu) in bytes also are displayed. You should become familiar with how these displays look on your system so that you will recognize changes and problems immediately.

   Examples:

```
# /etc/ping otherhost
PING otherhost: 56 data bytes
64 bytes from 123.123.12.13: icmp_seq=0. time=10. ms
<CONTROL-c>
```

```
# netstat -i
Name      Mtu   Network          Address          Ipkts     Ierrs    Opkts     Oerrs
en0*      1496  cray-net         cray             0         0        2         0
fddi0     4352  crau-fddi-net    cray-fddi        249466    0        57636     0
fddi1*    4352  none             none             0         0        0         0
lo0       65535 loopback         localhost        264       0        264       0
```

Note

An * in the Name column of the `netstat -i` command output indicates that the interface is not available to your CRAY J90 or CRAY EL system, so your CRAY J90 or CRAY EL system cannot access that network.

7. Reboot the system.

   Reboot the system to verify that all of your changes are handled automatically during system startup. (For booting procedures, see section 3 of this guide.) At some point, you should again test the network by using `ping` and view the configuration by using `netstat`.

8. Transfer full configuration files from another system.

   At this point, you are ready to do full local networking configuration by transferring existing copies of configuration files, as shown in the following example. Save copies of your original files and add the new entries for the CRAY J90 or CRAY EL system to the files you transfer (for example, use `ftp` to transfer the `/etc/hosts` file from another system on your network).

   Example:

```
# cd /etc
# cp hosts hosts.sav
# ftp cyclone
Connected to cyclone.cray.com.
220 fred FTP server (Version 5.2 Fri Feb 18 14:09:58 CDT 1994) ready.
Remote system type is UNIX.
Using binary mode to transfer files.
Name (fred:root): sam
331 Password required for sam
Password:                    <-- Enter your password
230 User sam logged in.
ftp> get /etc/hosts hosts
200 PORT command successful.
150 Opening BINARY mode data connection for /etc/hosts (328758 bytes).
226 Transfer complete.
328758 bytes received in 0.6 seconds (5.3e+02 Kbyte/s)
ftp> quit
221 Goodbye.
# vi /etc/hosts              <-- Add the new hosts entries
# /etc/mkbinhost
/etc/hosts.bin: 2675 entries written
```

Your CRAY J90 or CRAY EL system should now be on the network.

Common TCP/IP configuration files
11.2

After you have the Cray Research system on the network, you should do a few additional things to make sure it is a fully functional member of your network, including configuring `inetd`, adding additional routes, and updating other configuration files. If you are using the Menu System, you should update these files by using the Menu System options. Table 2 describes some of the most common TCP/IP configuration files. *UNICOS System Administration*, publication SG–2113, describes all of these files.

Table 2. TCP/IP configuration files

File (relative to `/etc`)	Description	Change
`config/daemons`	Lists system and network daemons to start at system boot.	Probably
`gated.conf` or `tcpstart.mid`	Contains routes to be installed at system boot.	Yes
`config/hostname.txt`	Contains text host name for TCP/IP.	Probably not
`hosts`	Maps Internet addresses to host names.	Yes
`hosts.equiv`	Lists trusted hosts for `rlogin`, `rsh`, etc.	Optional
`hycf.`*xxx*	Maps physical addresses to Internet addresses for nonbroadcast media (for example, HYPERchannel and HIPPI).	Yes, if you have HYPERchannel or HIPPI
`inetd.conf`	Lists network services to be handled by `inetd`.	Probably not
`config/interfaces`	Lists network interfaces and their characteristics.	Yes
`networks`	Maps network names to network Internet addresses.	Optional
`protocols`	Maps protocol names to protocol numbers.	No
`$HOME/.rhosts`	Lists trusted users for `rlogin`, `rsh`, etc.	Optional

Table 2. TCP/IP configuration files (continued)

File (relative to `/etc`)	Description	Change
`services`	Maps protocol and port numbers to service names.	Probably not
`shells`	Lists shells allowed for `ftpd`.	Probably not

Configuring NIS [12]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

Related NIS documentation

12.1

The following documentation contains information covered in this section:

- *UNICOS System Administration*, publication SG–2113
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `netstart`(8), `udbgen`(8), `ypxfr`(8), `ypbind`(8), `ypinit`(8), `yppasswdd`(8), `yppush`(8), `ypserv`(8), `ypstart`(8), and `ypxfr`(8) man pages
- *UNICOS User Commands Reference Manual*, publication SR–2011: `domainname`(1), `udbsee`(1), `yppasswd`(1), and `ypwhich`(1) man pages

What is NIS?

12.2

The Network Information Service (NIS) is a network service that allows information such as passwords and group IDs for an entire network to be held in one database. (NIS was formerly known as Yellow Pages.)

Implemented with the Remote Procedure Call (RPC) and eXternal Data Representation (XDR) library routines, UNICOS NIS has the following features:

- Look-up service: UNICOS NIS maintains a set of databases that can be queried through the use of pointers, or "keys." Programs can request the value associated with a particular key, or all of the keys, in a database.
- Network service: Programs do not have to know the location of data or how it is stored. Instead, they use a network protocol to communicate with a database server that contains the information.
- Distributed service: Databases are fully replicated on several machines, known as "NIS servers." The servers propagate updated databases among themselves, ensuring consistency.

The UNICOS NIS environment includes at least one Cray Research system and one or more other hosts that also run NIS.

NIS databases contain maps; a *map* contains information that is usually found in an ASCII configuration file. Each map contains a set of keys and associated values. For example, the `passwd` map contains user names (the keys) and their associated `/etc/passwd` file entries (the values). The NIS maps are stored in `dbm` format. The `makedbm` command converts an ASCII file into a `dbm` format file that NIS can use. Usually, you do not have to worry about `dbm` format or the `makedbm` command. To generate the maps, you will use the makefile in the `/etc/yp` directory. For further information on the internal map format, see the `dbm`(3) and `makedbm`(8) man pages.

Cray Research supports the following maps on systems running the UNICOS operating system:

Map	Description
`group`	Performs the function of the `/etc/group` file: mapping group names to group IDs.
`netgroup`	Defines networkwide groups used for permission checking when doing remote mounts, remote logins, and remote shells.

Map	Description
`passwd`	Performs a few selected functions of the UDB on UNICOS systems; namely, password, home directory, and shell lookup.
`publickey`	Used for secure RPC, the `publickey` map contains public key/private key pairs for users and hosts on the network. For more information about running secure RPC, see the NIS section in *UNICOS System Administration*, publication SG–2113.

An important distinction to make is that a Cray Research system running UNICOS can **serve** other NIS maps for its domain; however, the Cray Research system (as a client) **consults** only the preceding maps. For more information about supported maps, see the NIS section in *UNICOS System Administration*, publication SG–2113.

An *NIS domain* is a specified set of NIS maps. The set of maps for a given domain is stored in a directory named after the domain. To assign hosts to a particular domain, use the `domainname` command.

Servers provide resources; clients use them. There are two types of NIS servers: master servers and slave servers. The *master server* contains the NIS maps. You can change NIS maps only on the master server. A *slave server* contains copies of the NIS maps that it obtains from the master server for its domain.

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

Note

You should configure your CRAY J90 or CRAY EL system as an NIS slave server.

This section contains procedures for the following:

- Using the menu system to configure your CRAY J90 or CRAY EL system as an NIS slave server
- Configuring your CRAY J90 or CRAY EL system as an NIS slave server without using the menu system
- Configuring user accounts to use NIS

Caution

UNICOS NIS differs from the NIS facility used on other systems based on the UNIX system. Administration of an NIS domain that includes a Cray Research system is different from administration of an NIS domain that does not. For example, UNICOS NIS does not support the broadcast feature. If you are unfamiliar with NIS, you should first read the NIS documentation for your other systems. After you have familiarized yourself with the general NIS mechanism, read *UNICOS System Administration*, publication SG–2113, to familiarize yourself with UNICOS NIS before you configure NIS on your CRAY J90 or CRAY EL system.

Procedure: Using the menu system to configure your CRAY J90 or CRAY EL system as an NIS slave server

Note

This procedure assumes that another host on the network is already configured as an NIS master server and that your CRAY J90 or CRAY EL system can communicate with that host.

To configure NIS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group. For information about starting system daemons, see section 4, page 49, *UNICOS System Daemons*.

Before you configure NIS on your CRAY J90 or CRAY EL system, be sure to read the cautionary note and other important information on pages 286 and 288.

1. Enable the menu system to configure NIS. To give the menu system permission to change the configuration for NIS, ensure that the `NIS configuration` option is set to `YES` in the `Configure System ==> Configurator Automation Options` menu. Also, ensure that the `Configure System ==> Major Software Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

2. Assign your CRAY J90 or CRAY EL system to an NIS domain.

   Select the `Configure System ==> Network Configuration ==> NIS Configuration` menu. Enter the NIS domain name, and then activate the NIS configuration. A sample menu screen follows:

```
Configure System
...Network Configuration
......NIS Configuration
```

```
                              NIS Configuration

S-> NIS domain name
    Import the NIS configuration ...
    Activate the NIS configuration ...
```

3. Run the `/etc/yp/ypinit` script with the `-s` option to configure the CRAY J90 or CRAY EL system as an NIS slave server. Include the host name of the NIS master for your domain on the command line, as follows:

```
# /etc/yp/ypinit -s NISmasterserver
```

Running `ypinit -s` *NIS_masterserver* causes a copy of the NIS maps to be transferred from the master to the slave server (your Cray Research system running UNICOS) and placed in the `/etc/yp/`*domainname* directory. Running this command also adds the slave server to the `ypservers` map for your domain.

Note

You must run `ypinit` only once, when you first install the host as a slave server. After that, you can perform map updates by using either the `yppush` command from the master server or the `ypxfr` command from the slave server.

---

4. Start the NIS daemons, `ypserv` and `ypbind`.

Note

The procedure for starting NIS daemons differs from the procedure for starting other system daemons.

An administrator (`root`) can start all daemons manually from the command line. To start the `ypbind` daemon, use the `-h` option, as follows:

```
# /etc/ypserv
# /etc/ypbind -h yourCRAYJ90orCRAYELhostname
```

You may start the daemons manually when you first install NIS to verify that everything is working. After that, the daemons will be started automatically each time the system boots because when you activated the menu in step 1, your NIS domain name was written into the `/etc/config/ypdomain.txt` file. At system startup, the `/etc/ypstart` script accesses the `/etc/config/ypdomain.txt` file, automatically sets the NIS domain name, and then starts the `ypserv` and `ypbind` daemons. You **must not** add the daemons to the `/etc/config/daemons` file. The `ypstart` script assumes that you are running the Cray Research system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

5. Verify that the CRAY J90 or CRAY EL system has bound to itself by using the `ypwhich` command. It is normal to see that the domain has not bound the first time you execute `ypwhich`; simply enter it a second time, as follows:

```
# ypwhich
Domain domainname not bound.
# ypwhich
yourCRAYJ90orCRAYELsystem
```

Procedure: Configuring your CRAY J90 or CRAY EL system as an NIS slave server without using the menu system

Note

This procedure assumes that another host on the network is already configured as an NIS master server and that your CRAY J90 or CRAY EL system can communicate with that host.

To configure NIS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group. For information about starting system daemons, see section 4, page 49, *UNICOS System Daemons*.

Before you configure NIS on your CRAY J90 or CRAY EL system, be sure to read the cautionary note and other important information on pages 286 and 288.

1. Edit the `/etc/config/rcoptions` file and set the `RC_YP=` parameter to `YES`.
2. Assign your CRAY J90 or CRAY EL system to an NIS domain.

   To set the NIS domain, use the `domainname` command as follows; *domainname* is the name of your NIS domain:

```
# domainname domainname
```

3. Run the `/etc/yp/ypinit` script with the `-s` option to configure the CRAY J90 or CRAY EL system as an NIS slave server. Include the host name of the NIS master for your domain on the command line, as follows:

```
# /etc/yp/ypinit -s NISmasterserver
```

   Running `ypinit -s` *NIS_masterserver* causes a copy of the NIS maps to be transferred from the master to the slave server (your Cray Research system running UNICOS) and placed in the `/etc/yp/`*domainname* directory. Running this command also adds the slave server to the `ypservers` map for your domain.

Note

You must run `ypinit` only once when you first install the host as a slave server. After that, you can perform map updates by using either the `yppush` command from the master server or the `ypxfr` command from the slave server.

4. Start the NIS daemons, `ypserv` and `ypbind`.

Note

The procedure for starting NIS daemons differs from the procedure for starting other system daemons.

An administrator (`root`) can start all daemons manually from the command line. To start the `ypbind` daemon, use the `-h` option, as follows:

```
# /etc/ypserv
# /etc/ypbind -h yourCRAYJ90orCRAYELhostname
```

You may start the daemons manually when you first install NIS to verify that everything is working. After that, you should configure the daemons so that they are started automatically each time the system boots; see "To have NIS start automatically when you start UNICOS" at the end of this procedure.

5. Verify that the CRAY J90 or CRAY EL system has bound to itself by using the `ypwhich` command. It is normal to see that the domain has not bound the first time you execute `ypwhich`; simply enter it a second time, as follows:

```
# ypwhich
Domain domainname not bound.
# ypwhich
yourCRAYJ90orCRAYELsystem
```

To have NIS start automatically when you start UNICOS

To start NIS automatically when you start UNICOS, specify the NIS domain name by placing the name in the `/etc/config/ypdomain.txt` file, as follows:

```
# echo your_NIS_domain_name > /etc/config/ypdomain.txt
```

When you start UNICOS in the future, the `/etc/ypstart` script will access the `/etc/config/ypdomain.txt` file, set the NIS domain name automatically, and then start the `ypserv` and `ypbind` daemons. You **must not** add the daemons to the `/etc/config/daemons` file. The `ypstart` script assumes that you are running the Cray Research system as an NIS slave server. Any other configuration will require you to modify the `ypstart` script.

Procedure: Configuring user accounts to use NIS

Note

This procedure assumes that your CRAY J90 or CRAY EL system has already been configured as an NIS slave server (see the preceding procedure).

You can configure NIS so that a user's password, home directory, and default shell are obtained from the NIS `passwd` map, rather than from the UNICOS user database (UDB).

1. Set the user account `permbits` to `yp` and the `passwd`, `dir`, and `shell` fields to null by using `udbgen`.

   Example:

```
# /etc/udbgen -c "update:john:permbits:yp:passwd::dir::shell::"
```

2. Verify that the user database entry is correct, using the `udbsee` command.

   Example:

```
# udbsee john
create  :john:   uid     :10055:
        comment :John Stephen Smith:
        passwd  ::
        gids    :175:
        acids   :10055:
        dir     ::
        shell   ::
        root    :/:
        logline         :/dev/ttyp003:
        loghost         :asbestos:
        logtime         :748554978: # Mon Jan 10 14:56:18 1994
        resgrp          :175: # uid
        permbits        :yp:
        .
        .
        .
```

3. Inform NIS users that they must use the `yppasswd` command to change their password, rather than the `passwd` command. The `passwd` command also will inform NIS users to use

`yppasswd` if they forget (see section 7, page 189). The `yppasswd` command works only if you have started the `yppasswdd` daemon on the NIS master server machine.

Configuring NFS [13]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

Related NFS documentation

13.1

The following documentation contains information covered in this section:

- *UNICOS System Administration*, publication SG–2113
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `automount`(8), `biod`(8), `cnfsd`(8), `exportfs`(8), `mount`(8), `mountd`(8), `nfsd`(8), `nfsidmap`(8), and `sdaemon`(8) man pages
- *UNICOS File Formats and Special Files Reference Manual*, publication SR–2014: `exports`(5) and `fstab`(5) man pages

What is NFS?

13.2

The network file system (NFS) is a Cray Research software product that allows users to share directories and files across a network of machines.

NFS users can use standard I/O system calls, commands, and permission controls to access files from any file system. Similarly, other NFS users can make use of file systems by using remote commands from anywhere in the local network

environment. You can use NFS in diverse administrative environments through the use of the ID mapping facility (see subsection 13.3). By default, this facility is on in the UNICOS kernel. The user interface to NFS is transparent.

NFS uses a server/client system to provide access to files on the network. A *fileserver* is any machine that allows a portion of its local disk space to be exported (made available for mounting on a host machine). A *client* is any machine that makes a request for an exported file system. When a user issues an I/O call for a file that resides on a file system mounted by NFS, the call is transmitted to the server machine. When the server receives the request, it performs the indicated operation. In the case of read or write requests, the indicated data is returned to the client or written to disk, respectively. This processing is transparent to users, and it appears that the file resides on a disk drive that is local.

NFS client operations are separate from NFS server operations. This section describes the procedures for configuring a CRAY J90 or CRAY EL system as an NFS client and as an NFS server.

For addition information about NFS, including information about the following topics, see *UNICOS System Administration*, publication SG–2113:

- NFS automounter (`automount`(8) command), which is a program that runs on an NFS client that mounts and unmounts NFS file systems on demand. With the automounter, NFS file systems are mounted only when users are accessing them.
- General security concerns; although UNICOS NFS is an excellent tool for sharing files between computer systems, it also makes the files on a server vulnerable to unauthorized access.
- Kerberos authentication, which can be required for NFS access to exported UNICOS file systems.

What is ID mapping and when would I use it?

13.3

In UNICOS, file access is controlled by checking the numeric user ID (UID) and group ID (GID) against the permissions bits for a file. These same rules apply to NFS. Therefore, in a standard NFS implementation, UNICOS NFS is designed to be used within one administrative domain, which is sometimes called a *flat administrative space*. An *administrative domain* is a set of hosts, usually managed by the same authority, in which all users share a common set of UIDs and GIDs. With NIS, a given user or group ID always refers to the same user or group within the administrative domain. This allows all hosts in the NFS group to interpret the authentication information passed in the NFS requests in the same way. Traditional NFS environments make use of NIS (formerly called Yellow Pages) to achieve a flat administrative space. NIS is a distributed look-up service that maintains a common database of UID and GID information for members of an administrative domain. An NIS domain is one administrative domain.

If your Cray Research system resides entirely within one NIS domain and does not interact with hosts outside that domain, you probably will not have to configure NFS ID mapping. However, Cray Research systems are often shared by many different administrative domains, making the creation of a single, flat administrative space for user and group identification technically and/or organizationally difficult. Because a given ID can refer to different users or groups in different administrative domains, this would prevent NFS from being used in such an environment, or would cause serious security problems.

The Cray Research system NFS ID mapping facility was designed to allow different administrative domains to participate in cross-mounting NFS file systems without creating a single, flat administrative space.

Following is a description of circumstances in which it is desirable and circumstances in which it is necessary for the Cray Research NFS server to access ID mapping information:

- Account IDs, or ACIDs, which are unique to UNICOS, are not passed across the network as part of the NFS protocol. If ID mapping is configured, NFS servers can use the requesting user's ACID for operations such as file creation. This allows NFS-created files to be charged correctly when using ACIDs for disk accounting and/or file quotas.

- On UNICOS MLS systems (with or without using the IP security option), a UNICOS NFS server must be able to validate requests based on the user's security levels and compartments. If you want to export and serve file systems on a UNICOS MLS system, ID mapping is required.
- If you want to export file systems by using the `-krb` option (Kerberos authentication), ID mapping is required so that the kernel has a place to put a list of authenticated addresses for each Kerberos user.

For additional information on NFS ID mapping, see the "Network File System (NFS)" section in *UNICOS System Administration*, publication SG–2113.

Procedure: Configuring a CRAY J90 or CRAY EL system as an NFS client

Note

If you are the administrator on the server(s) and the client(s), you should know whether you exported the file systems you want to mount. To see the file systems that are currently exported, execute the `exportfs` command without arguments on the server.

1. Make entries in the `/etc/fstab` file that describe the file systems you want mounted using NFS. You can use the menu system to do this step or you can do this manually.

   If you are using the menu system, you must first enable the menu system to configure NFS. To give the menu system permission to change the configuration for NFS, change the `NFS configuration` option to `YES` in the `Configure System ==> Configurator Automation Options` menu. Also, ensure the `Configure System ==> Major Software Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you must change the `Major Software Configuration` menu, you must rebuild your kernel.

   Then, select the `Configure System ==> File System (fstab) Configuration ==> NFS File Systems` menu, add your entries, and update the form file. Then activate your changes through the `File Systems (fstab) Configuration` menu. A sample `Network File System Configuration` menu screen follows:

```
Configure System
...File System (fstab) Configuration
......NFS File Systems
```

```
                        Network File System Configuration
      Host      Name                Mount           Rw  Quota  Suid  Auto  Bg  So
     -------  ----------------  -------------  --  -----  ----  ----  --  -- >
E->  tngmoon  /usr/bin           /UTNA/sunbin   ro                      bg  so
```

If you are not using the menu system, edit the `/etc/config/rcoptions` file and set the `RC_NFS=` parameter to `YES`. Then make entries in the `/etc/fstab` file that describe the file systems you want mounted using NFS. The following example shows sample entries; for more information about the options, see the `fstab`(5) and `mount`(8) man pages and *UNICOS System Administration*, publication SG–2113:

```
# cat fstab
#
# Mainframe file system table (fstab)
#
#       There are six fields per line, separated by white space.
#       1.  device name
#       2.  filesystem name
#       3.  filesystem type
#       4.  mount options
#       5.  dump frequency
#       6.  pass number to check file system
#
/dev/dsk/root   /                       NC1FS  rw    1    1
/dev/dsk/home   /home                   NC1FS  rw    1    2
/dev/dsk/core   /core                   NC1FS  rw    1    2
/dev/dsk/usr    /usr                    NC1FS  rw    1    2
/dev/dsk/src    /usr/src                NC1FS  rw    1    2
#
# NFS file systems
#
tngmoon:/home/tngmoon/user1     /UTNA/user1     NFS     ro,soft,bg
tngmoon:/usr/bin                /UTNA/sunbin    NFS     ro,soft,bg
```

2. The only daemon you must start on an NFS client is `biod`, which is an optional client daemon that handles write-behind and read-ahead requests. Although this daemon is optional, you should run it to improve NFS performance. By default, four `biod`'s are started; you may be able to improve client performance by running more `biod` daemons. Ensure that the `biod` daemon is started by using the menu system or by doing it manually.

Note

To configure NFS, the `portmap` daemon must be running (that is, it must be set to `YES` in the `/etc/config/daemons` file); `portmap` is part of the TCP daemons group. For information about starting system daemons, see section 4, page 49.

If you are using the menu system, select the `Configure System ==> System Daemons Configuration ==> System Daemons Table` menu, set the `biod` daemon to `YES`, and update the form file. Then activate your change through the `System Daemons Configuration` menu. When you activate this change, the `biod` daemon will be started automatically each time you start UNICOS. A sample `System Daemons Table` menu screen follows:

```
Configure System
...System Daemons Configuration
......System Daemons Table
```

```
                                System Daemons Table

     TCP   snmpd       YES  *                 /etc/snmpd                    >
     TCP   -           YES  -                 /usr/bin/domainname     ""    >
     TCP   portmap     YES  *                 /etc/portmap                  >
     TCP   keyserv     NO   *                 /etc/keyserv                  >
     TCP   ntpd        NO   *                 /etc/ntpd                     >
     NFS   nfsd        YES  *                 /etc/nfsd               4     >
     NFS   exportfs    NO   *                 /etc/exportfs           -av   >
     NFS   mountd      YES  *                 /etc/mountd                   >
E->  NFS   biod        YES  *                 /etc/biod               4     >
     NFS   pcnfsd      NO   *                 /etc/pcnfsd                   >
      .
      .
      .
```

If you are not using the menu system, to start the `biod` daemon, edit the `/etc/config/daemons` file and set the NFS `biod` daemon to be `YES`. (Editing this file will ensure that the `biod` daemon will be started automatically each time you start UNICOS in the future.) Then execute the `/etc/sdaemon` script to start `biod` now, as follows:

```
# /etc/sdaemon -s biod
```

Note

You cannot do the remaining steps to this procedure by using the menu system.

3. If you do not want to configure UNICOS NFS ID mapping at this time, you should disable this feature by executing the following command (for information on UNICOS NFS ID mapping, see *UNICOS System Administration*, publication SG–2113):

```
# /etc/uidmaps/nfsidmap -d
NFS ID mapping is disabled.
```

Note

To disable NFS ID mapping permanently, place the `/etc/uidmaps/nfsidmap -d` command in the `/etc/uidmaps/Set.domains` file; otherwise, if you want NFS ID mapping disabled, you must execute step 3 each time you start UNICOS NFS.

4. Create mount points (empty directories in which the NFS file systems will be accessed on your system) for the file systems you will mount using NFS.

   Example:

```
# cd /UTNA
# mkdir user1
# mkdir sunbin
```

5. If you would like the file systems to be mounted using NFS automatically when you start UNICOS, create the `/etc/mountnfs` script and include the appropriate `mount` commands. Based on the preceding examples, a sample script follows:

```
# cat /etc/mountnfs
# Script for mounting NFS file systems
#
mount /UTNA/user1 &
mount /UTNA/sunbin &
```

6. Ensure that the `/etc/mountnfs` script is executable by executing the following command:

```
# chmod +x /etc/mountnfs
```

7. Run the `/etc/mountnfs` script to mount the file systems by executing the following command; when you start UNICOS in the future, the script will be run automatically:

```
# /etc/mountnfs
[1] 85260
[2] 85261
```

8. Verify that the file systems have been mounted by using the `/etc/mount` and `df` commands, as shown in the following examples:

```
# /etc/mount
/ on /dev/dsk/root read/write on Mon Jan 10 08:45:33 1994
/tmp on /dev/dsk/tmp read/write on Mon Jan 10 08:46:11 1994
/usr on /dev/dsk/usr read/write,rw on Mon Jan 10 08:46:12 1994
/home on /dev/dsk/home read/write,rw on Mon Jan 10 08:46:13 1994
/usr/src on /dev/dsk/src read/write,rw on Mon Jan 10 08:46:13 1994
/proc on /proc read/write on Mon Jan 10 08:46:14 1994
/els_src on /dev/dsk/els_src read/write on Mon Jan 10 09:14:58 1994
/UTNA/sunbin on tngmoon:/usr/bin read only,ro,soft,bg on Mon Jan 10 18:13:41 1994
/UTNA/user1 on tngmoon:/home/tngmoon/user1 read only,ro,soft,bg on Mon Jan 10 18:13:41
1994
```

```
# df
/UTNA/user1  (tngmoon:/home/tngmoon/user1):
                                  100955  1K blocks ( 48.2%)
/UTNA/sunbin (tngmoon:/usr/bin ):  65879  1K blocks ( 50.7%)
/els_src     (/dev/dsk/els_src ): 178106  4K blocks ( 59.4%)*   57677 I-nodes
/proc        (/proc            ): 119400  4K blocks ( 95.5%)      412 procs
/usr/src     (/dev/dsk/src     ):  27132  4K blocks ( 18.1%)*   25654 I-nodes
/home        (/dev/dsk/home    ): 671031  4K blocks ( 97.8%)*  169883 I-nodes
/usr         (/dev/dsk/usr     ): 152964  4K blocks ( 59.0%)*   26694 I-nodes
/tmp         (/dev/dsk/tmp     ): 495065  4K blocks ( 98.8%)*   98292 I-nodes
/            (/dev/dsk/root    ):  13417  4K blocks ( 17.9%)*   16169 I-nodes
```

Procedure: Configuring a CRAY J90 or CRAY EL system as an NFS server

1. Describe the file systems you want to allow other systems to mount using NFS by placing entries in the `/etc/exports` file. (For a complete list of export options, see the `exports`(5) man page.) You can use the menu system to place your entries in the `/etc/exports` file or you can do this manually.

   If you are using the menu system, ensure that the menu system has permission to change the configuration for NFS by verifying the `NFS configuration` option is set to `YES` in the `Configure System ==> Configurator Automation Options` menu. Also, ensure that the `Configure System ==> Major Software Configuration` menu has the `Network Information Service (NIS)` option set to `on`; if you need to change the `Major Software Configuration` menu, you must rebuild your kernel.

   Then select the `Configure System ==> Network Configuration ==> NFS Configuration ==> List of Exported File Systems` menu, add your entries, and update the form file. Then activate your changes through the `NFS Configuration` menu. A sample `NFS Exported File Systems` menu screen follows:

```
 Configure System
...Network Configuration
......NFS Configuration
.........List of Exported File Systems
```

```
                    NFS Exported File Systems Configuration
    File System  Clients  ro  rw clients  Anon  UID  Root Clients  Sum  Ker
    -----------  -------  --  ----------  ----  ---  ------------  ---  --- >
E->  /home                 rw              anon  -2   edge
```

   If you are not using the menu system, ensure that the `RC_NFS=` parameter is set to `YES` in the `/etc/config/rcoptions` file. Then edit the `/etc/exports` file to describe the file systems you want to allow other systems to mount using NFS.

2. Look at the list of these file systems by using the `cat /etc/exports` command, as shown in the following example:

```
# cat /etc/exports
/nasc    -root=edge:sn1234
/home    -rw
/tmp
/UTNA/goodstuff
```

3. Make all file systems described in the `/etc/exports` file available to NFS client systems by using the menu system or doing it manually.

   If you are using the menu system, select the `Configure System ==> System Daemons Configuration ==> System Daemons Table` menu, set the `Start up at boot time?` option to `YES`, and update the form file. Then activate your changes through the `System Daemons Configuration` menu. A sample `System Daemons Table` menu screen follows:

```
Configure System
...System Daemons Configuration
......System Daemons Table
```

```
                          System Daemons Table

 S-> Group                                              NFS
     Name                                               exportfs
     Start up at boot time?                             YES
     Kill action                                        *
     Executable pathname                                /etc/exportfs
     Command-line arguments                             -av
     Additional command-line arguments
     Additional command-line arguments
```

   If you are not using the menu system, you can make all file systems described in the `/etc/exports` file available to NFS client systems by executing the `/etc/exportfs -av` command, as follows.

```
# /etc/exportfs -av
```

Important

If you are not using the menu system, you **must** run the `/etc/exportfs -av` command each time your system is rebooted. You should automate the execution of this command by using the menu system.

4. Enable the NFS server daemons; at a minimum, the TCP/IP `portmap` daemon and the NFS `nfsd` and `mountd` daemons must be started on an NFS server. You can use the menu system to enable the daemons or you can do this manually. The `cnfsd` daemon is necessary **only** when you have more than one Cray Research system; it is intended for use only between Cray Research systems. For more information, see the `exports`(5) and `cnfsd`(8) man pages.

If you are using the menu system, select the `Configure System ==> System Daemons Configuration ==> System Daemons Table` menu, set the NFS server daemons to `YES`, and update the form file. Then activate your changes through the `System Daemons Configuration` menu. When you activate this change, the daemons will be started automatically each time you start UNICOS. A sample `System Daemons Table` menu screen follows:

```
Configure System
...System Daemons Configuration
......System Daemons Table
```

```
                                System Daemons Table

  TCP    snmpd       YES  *                /etc/snmpd                      >
  TCP    -           YES  -                /usr/bin/domainname     ""      >
  TCP    portmap     YES  *                /etc/portmap                    >
  TCP    keyserv     NO   *                /etc/keyserv                    >
  TCP    ntpd        NO   *                /etc/ntpd                       >
  NFS    nfsd        YES  *                /etc/nfsd               4       >
  NFS    exportfs    YES  *                /etc/exportfs           -av     >
  NFS    cnfsd       NO   *                /etc/cnfsd              4       >
  NFS    mountd      YES  *                /etc/mountd                     >
  NFS    biod        YES  *                /etc/biod               4       >
  NFS    pcnfsd      NO   *                /etc/pcnfsd                     >
  .
  .
  .
```

If you are not using the menu system, edit the `/etc/config/daemons` file to enable the NFS server daemons, as shown in the following example. (If you do not use the menu system, editing this file also will ensure that the daemons will be started automatically each time you start UNICOS.)

```
# vi /etc/config/daemons
TCP     portmap         YES     *               /etc/portmap
NFS     nfsd            YES     *               /etc/nfsd       4
NFS     cnfsd           NO      *               /etc/cnfsd      4
NFS     -               YES     -               /etc/exportfs   -av
NFS     mountd          YES     *               /etc/mountd
```

Then execute the `/etc/sdaemon` script as follows to start the NFS server daemons (you should have the TCP/IP `portmap` daemon already running):

```
# /etc/sdaemon -g NFS
```

NQS [14]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

The UNICOS Network Queuing System (NQS) is a batch facility your users can access to submit their batch jobs to UNICOS. Like many other utilities, NQS has both a user and an administrator interface. NQS lets users submit, terminate, monitor, and (within limits) control jobs submitted to either the local system or another appropriately configured computer system within the network. To send requests to an NQS queue, use the `qsub` command in the following ways:

- An interactive user can submit an NQS request to the local UNICOS system.
- A user can submit an NQS request to another UNICOS system in the user's network. By using the `qsub -q` command, a user also can submit a request to a remote UNIX system running a public-domain compatible version of NQS, as long as a TCP/IP link exists between the remote system and the target machine.
- Using the Cray Research Network Queuing Environment (NQE) running on any network host, a user can submit UNICOS batch requests to NQS running on the Cray Research system. The NQE feature for the UNICOS operating system consists of the Network Queuing EXtensions (NQX), Network Queuing System (NQS), and File Transfer Agent (FTA) components. NQE is a software product that consists of a set of servers and clients that allow batch requests to be executed across a load-balanced network of hosts known as a *batch complex*. Specifically, batch requests are submitted from NQE clients and executed at NQE master and execution servers. The NQX component of NQE provides a Network Load Balancer (NLB) that supports destination selection and load

balancing. NQX also provides the NQE client software commands that communicate with NQE servers to submit jobs to NQE and monitor NQE system and job status. The NQS component of NQE lets users submit, monitor, and control batch jobs for execution on a local or remote system running the UNICOS system. The FTA component of NQE queues synchronous or asynchronous outbound and inbound file transfers over the network.

For more information about NQE, see the following publications:

- *Introducing NQE*, publication IN–2153
- *NQE User's Guide*, publication SG–2148
- *NQE Administration*, publication SG–2150

For additional user-level NQS information, see subsection 14.7, page 351.

Note

When you first boot the CRAY J90 or CRAY EL system, NQS is **not** enabled. As an administrator, you must configure your system for NQS and ensure smooth throughput. You will create queues and establish specific characteristics (limits) for queues, such as size, processing priority, and access restrictions.

After you have configured and started NQS, you may want to submit test requests before allowing your users to submit their requests. To help you test your initial NQS, subsection 14.7, page 351, includes a list of NQS user commands and two examples of submitting requests to NQS.

The remainder of this section provides information that you need as a system administrator to set up an initial NQS environment at your site. This section will help you to do the following:

- Access NQS by using the `qmgr` command
- Understand a basic structure of NQS
- Configure an NQS environment comprised of three batch queues and a pipe queue
- Set up an NQS log file

- Set up NQS accounting
- Set the shell strategy
- Start NQS
- Stop NQS
- Obtain an image of your NQS configuration
- Submit jobs to NQS to test your system before you allow other users to access it
- Configure others to have NQS manager-level and NQS operator-level privileges

Related NQS documentation
14.1

The Network Queuing System (NQS) section, in *UNICOS System Administration*, publication SG–2113, contains in-depth information about NQS. You should read it to learn about and understand the full capabilities of NQS. The following publications also provide additional NQS documentation:

- *UNICOS User Commands Reference Manual*, publication SR–2011: `qalter`(1), `qchkpnt`(1), `qdel`(1), `qlimit`(1), `qmsg`(1), `qping`(1), `qstat`(1), and `qsub`(1) man pages
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022: `qconfigchk`(8), `qmgr`(8), `qstart`(8), and `qstop`(8) man pages
- *Introducing NQE*, publication IN–2153, and the NQE section in *UNICOS System Administration*, publication SG–2113, contain NQE information, which is outside the scope of this basic administration guide

Accessing NQS
14.2

To access NQS, use the `qmgr`(8) administrator command to enter the `qmgr` subsystem. Several commands are available to use within the `qmgr` subsystem. You can examine these commands by either looking at the `qmgr` man page or entering the `qmgr` utility and viewing the help screens. To enter the `qmgr` help utility, enter the command as shown in bold type in the following screen.

Note

The help information for the qmgr utility indicates the required characters for each command in uppercase.

```
qmgr

Qmgr 80.26: help

The available commands are as follows:

       ABort      ADd        Create     DElete     DIsable    ENable
       EXit       HElp       HOld       Initialize Lock       MODify
       MOVe       PReempt    PUrge      Quit       RELease    REMove
       RERun      RESEt      RESTore    RESUme     SChedule   SEGment
       SEt        SHOw       SHUTdown   SNap       STArt      STOp
     SUspend    Terminate Unlock
```

Manager and operator authorities within the NQS qmgr subsystem

14.3

Three levels of authority exist inside the NQS qmgr subsystem.

Ordinary users (nonmanagers) can issue qmgr display commands beginning with the verb show. Users can access only commands that show information; they cannot make changes.

The second level of authority is the *operator level*. A person who has operator-level authority can access only those commands used to monitor and control queue requests; that is, an operator can manipulate jobs in queues, move them from queue to queue, change the limits on a job, suspend and resume jobs, and such.

The third level of authority is the *manager level*. A person who has manager-level authority can use the complete set of qmgr commands to control and configure NQS and can configure NQS queues, start them, stop them, delete them, and such, in addition to executing any commands that an operator can perform.

The following are examples of `qmgr` commands that require full manager privileges:

```
add manager
set log_file
create batch_queue
create pipe_queue
remove queue
delete queue
```

The following are examples of `qmgr` commands that require operator privileges:

```
enable queue
hold request
suspend request
purge queue
```

Users can use display `qmgr` commands (beginning with the verb `show`). Examples are as follows:

```
show all
show managers
show queue
```

The procedure on page 343 describes how to add and remove `qmgr` managers and operators.

Sample NQS configuration steps

14.4

To give you an understanding of the basic structure of NQS, this subsection describes how to create an NQS configuration that is comprised of three batch queues and a pipe queue. Setting up the NQS log file, NQS accounting, and NQS shell strategy is also explained. Also included are a sample NQS directives file (which you may choose to use as your `/etc/config/nqs_config` file), a sample `/etc/config/NQS.startup` file, and procedures to configure and start NQS.

Note

When you receive your CRAY J90 or CRAY EL system, it has the `root` user already configured at the manager level of authority by default. However, when you first bring up your CRAY J90 or CRAY EL system, NQS is not enabled; therefore, you must configure your system for NQS and start NQS (described in this subsection).

After you enter the `qmgr` subsystem as `root`, you will have manager-level authority, and you can issue NQS commands to perform common operations, such as creating or removing queues or adding or removing other `qmgr` managers or operators.

This subsection includes the following information:

- Submitting NQS configuration directives to the NQS subsystem
- Types of directives that are necessary for a basic NQS configuration
  - Telling NQS your machine ID
  - Creating queues
- Making the connection between the pipe queues and the batch queues
- Setting up limits for the batch queues, including the following:
  - Individual queue limits
  - Queue complex limits
  - Global queue limits
- Setting the NQS log file
- Setting the debug level
- What to do if a user submits a job with no `qsub` directives
- Turning on and stopping the queues
- Recognizing a "good" NQS configuration
- Setting job limits for a job

Submitting NQS configuration directives to the NQS software
14.4.1

You can submit configuration directives to the NQS software in two ways:

- Type `qmgr` and press RETURN. You will see a prompt that looks like `Qmgr 80.26:`. At this point, you are working within the `qmgr` database software and can type directives one at a time. A very useful help feature is available at this level (type `help`).
- Edit a file in the directory of your choice, and create all of the appropriate configuration directives in one long script. When you are finished, submit the script to `qmgr`, which uses the redirection symbol in the command, as follows:

  `qmgr <` *filename*

The second method is the preferred method to use in setting up an initial NQS configuration file, and it is the method used in this section to set up a basic NQS.

Note

To submit NQS configuration directives to `qmgr`, you must have manager level of authority.

Types of directives necessary for a basic NQS configuration
14.4.2

This subsection describes the directives that are necessary for a basic NQS configuration, and it includes a sample NQS directives file, page 330, for you to refer to as you read about each directive.

Telling NQS your machine ID
14.4.2.1

You must configure a machine ID for your machine to define the local host system on which the NQS software is running. The syntax for the `qmgr` directives is as follows:

`add mid` *xx hostname*

The number (*xx*) you choose must not be a 0; it must be a unique number for the machine, and it must be $0 < mid \leq 2^{31} - 1$ (2,147,483,647). The local host system name (*hostname*) you choose should match the name of your machine as shown when you use the `hostname`(1) command. An identical entry for *hostname* must also be in the `/etc/hosts` file.

Example:

```
add mid 42 sn5316
```

To add host name aliases, use the `add name` command. You must add all possible routes as aliases, such as `sn5316.cray.com`.

The only valid transport service type for CRAY J90 or CRAY EL systems is TCP/IP (`tcp`).

Note

To configure NQS to remote host systems, see *UNICOS System Administration*, publication SG–2113.

If you are running NQS but will not be using networking, you must still configure a machine ID in the NQS database for your machine, as shown in the sample NQS directives file, page 330.

Creating the types of queues NQS uses 14.4.2.2

To control user requests, NQS uses the following types of queues:

- A *batch queue* accepts and processes user batch job requests, then later releases the jobs to UNICOS for execution. You can set a batch queue to have specific limits on users, amount of memory used, and so on.
- A *pipe queue* transfers requests to other queues.
- A *destination selection queue* routes requests to hosts per a load-balancing policy. This queue is available only if you have installed the Network Queuing Environment (NQE).

Batch requests that users submit to a batch queue are more commonly called *jobs*, and *processes* are parts of a job. Each job has the following attributes:

- Resource limits: Resource limits are specified explicitly by the user through a `qsub(1)` command option or implicitly by NQS, based on the queue in which the request resides, or in the UDB, based on whichever location is more restrictive. Typical limits are maximum file size, CPU time, and maximum memory size.

- Access restrictions: For each queue, access can be either restricted or unrestricted. If access is restricted, only requests submitted by users defined in the access set for the queue are accepted. If access is unrestricted, any request may enter the queue.

  The access set is composed of individual users or groups, as defined by the system administrator.

  You can use the `qmgr add users` command to add `root` explicitly to the queue access list; `root` no longer has automatic access to all queues.

- URM: The UNICOS Unified Resource Manager (URM) is a job scheduler that provides a high-level method of controlling the allocation of system resources to run jobs. When you enable URM, NQS registers certain jobs with URM. The jobs that are registered depend on which job scheduling type has been specified by the `qmgr set job_scheduling` command. The jobs registered with URM are in the scheduling pool and have a major status of `Q` and a substatus of `us`. URM has knowledge of the actual resource state of the machine and an evaluation and ranking mechanism that has weighting factors for 10 different aspects of a job's resource requirements. URM advises NQS when to initiate jobs that have been placed in the scheduling pool. (For more information on URM, see *UNICOS System Administration*, publication SG–2113.) When URM scheduling is disabled, the queue priority determines the order of requests in an NQS queue and the order in which requests are initiated. Queue priority can be determined by requested CPU, memory, number of shares (system resources a user is using), time in the queue, or the `qmgr schedule request` command.

- Nice value: The nice value is a request attribute that determines the execution priority of the processes that form the job. The nice value is set at the job level, and all processes of that job have the same nice value as the job. This value can be set explicitly or implicitly. Users can change the nice value of their jobs by using the `qsub -ln` command; however, users cannot increase their job priority, they can only lower it. You can set nice values by using the `qmgr` utility subcommands `set nice_increment` and `set nice_value_limit` (provided for compatibility with other NQS systems). Use these commands to set the nice value for any job placed in the specified queue when the job does not otherwise define an explicit job nice-increment. To run these commands, you must have NQS manager privileges.

- Shell interpretation: As one of a request's attributes, a user can specify the full path name of a shell to interpret the request's shell script. This is done with the `-s` option of the `qsub`(1) command. If this shell is not specified by a request, the NQS default shell strategy, defined in the `NQS.startup` file, is used. An administrator also can define the shell strategy by using the `qmgr shell_strategy` command (see the `SET shell_strategy` command).

Most sites can satisfy their batch job needs with only one pipe queue and a dozen or less batch queues. The number of batch queues depends on your site and job flow. With URM scheduling, only one batch queue should be necessary.

A pipe queue acts as a manager or decision-maker. A user submits a job to a pipe queue and that pipe queue decides in which batch queue the job fits, based on the information contained in the user's `qsub` directives about how much memory that job will need, how long it will take, and so on.

Batch queues are queues of jobs actually submitted to the system. As a system administrator, you must configure what you think appropriate limits will be for your batch queues, to ensure that the system receives the kind of job mix that you want to emphasize at your site.

In the sample NQS directives file, page 330, one pipe queue (`gateway`) is configured, and three batch queues (`little`, `medium`, and `large`) are configured.

The pipe queue in the sample is created by using the following configuration directive:

```
create pipe_queue gateway priority=60 server=(pipeclient)
```

The server path is the default location at which NQS installs its `pipeclient`. A `pipeclient` is spawned for requests in pipe queues and it enqueues a request at the appropriate destination queue.

Note

The pipe queue's priority **must** be higher than any of the batch queues. It is more important for decisions about the job mix, job flow, and system resources to be made by the pipe queue than for any individual batch queue to run any particular job.

For example, the batch queue `little` is created by using the following `qmgr` directive; the priority is set somewhat less than the priority of the pipe queue:

```
create batch_queue little priority=55 pipeonly
```

In the sample file, the batch queue `medium` has a priority of `45`, and the batch queue `large` has a priority of `35`. In this sample configuration, the most important queue after the pipe queue is queue `little`. This means that the pipe queue will look at queue `little` first to see whether it can accept more jobs.

In the sample file, the `pipeonly` directive at the end of each of the `create` directives ensures that each batch queue can receive new jobs **only** from a pipe queue and not from an individual user trying to submit a job directly to one of these batch queues.

A batch queue can be declared `loadonly`, meaning that it can accept only a limited number of jobs. The number of jobs that can be queued is restricted to the run limit of that queue. Therefore, if there are no other limits to consider, a load-only queue accepts only the number of jobs that it can run. Specifically, the *number of jobs queued + number of jobs running + number of jobs waiting + number of jobs arriving* $\leq$ *run limit*. You can use this for load sharing between multiple machines or across multiple queues on a single machine. If a job cannot enter a `loadonly` batch queue, it remains (in `WAIT` state) in the source pipe queue, and it is retried at regular intervals.

Making the connection between the pipe queue and the batch queues 14.4.3

In the sample file, users must submit their jobs to the pipe queue `gateway` because all of the batch queues are `pipeonly`. The `set destination` directive makes the link between the pipe queue `gateway` and the batch queues `little`, `medium`, and `large`, directing the pipe queue which batch queues to feed. The syntax is as follows:

```
set destination = (firstqueue, secondqueue, thirdqueue) pipequeue
```

In the example, the configuration directive is as follows:

```
set destination = (little, medium, large) gateway
```

Note

The order in which the destination batch queues are specified is very important. When the pipe queue has a new job to submit to a batch queue, the pipe queue compares the limits specified on the incoming job to the limits specified for each queue, and the pipe queue starts with the first specified destination queue first. This means that you should specify the destination batch queue with the **most** restrictive limits first and end your list with the destination batch queue with the **least** restrictive limits.

If the sample destination directive had been the following and if a user had a job that would take only 10 CPU seconds and use 1 Mword of memory and submitted the job to the pipe queue `gateway`, `gateway` would first see whether that job fit the restrictions of queue `large`:

```
set destination = (large, medium, little) gateway
```

Because queue `large` accepts jobs that would run forever, and use unlimited amounts of memory, `gateway` would decide that this job fit into queue `large`. In fact, **every** job submitted to `gateway` would end up in queue `large`, because it has the least-restrictive limits and was defined first in the destination statement.

Setting limits for the batch queues 14.4.4

Batch queue limits define which jobs the batch queue will accept. You can set individual batch queue limits, queue complex limits, and global queue limits.

Individual queue limits 14.4.4.1

Batch queues are differentiated primarily by their limits. Resource and other limitations help control the job mix and the job flow on the system. In the sample file, queue `little` has a run limit of 16. This means that no more than 16 jobs may run in queue `little` at one time. If a 17th job is submitted to the pipe queue `gateway` and that job would run in batch queue `little`, that 17th job is still submitted to queue `little` where it will wait until one of the other 16 jobs in queue `little` finishes and room exists for a new 16th job to run. The sample directives file sets the run limit of queue `little` to 16 by using the following `qmgr` directive:

```
set queue run_limit=16 little
```

Queue `little` has a user limit of 4, which means that no individual user can have more than four jobs running in queue `little` at one time. If a user submits a fifth job that would run in queue `little`, one of that user's original four jobs must finish before that fifth job may run in batch queue `little`. The sample directives file sets the user limit of queue `little` to 4 by using the following `qmgr` directive:

```
set queue user_limit=4 little
```

Queue `little` has a group limit of 8. This means that no more than eight jobs from the same group may run in queue `little` at one time. If three different people from the same group each submit 10 jobs, which all happen to fit in this queue (a possible total of 30 jobs), the user limit will allow each user to run only four jobs each (a possible total of 12 jobs), but the group limit will allow a total of only 8 jobs from these three people to run because they are all from the same group. If the first and second person each get their maximum of four jobs to run, the third person may not run any jobs at all, until one of those other 8 jobs finishes. The sample directives file sets the group limit of queue `little` to 8 by using the following `qmgr` directive:

```
set queue group_limit=8 little
```

The per_process and per_request limits can best be understood if you think of a request as a job, and a process as a piece of a job. In the sample file, queue little is limited to jobs that take 100 CPU seconds or less and processes that take 70 CPU seconds or less, and to jobs that use 4 Mwords of memory or less and processes that take 2 Mwords of memory or less. These limits were set using the following qmgr directives:

```
set per_request cpu_limit=100 little
set per_process cpu_limit=70 little
set per_request memory_limit=4mw little
set per_process memory_limit=2mw little
```

Note

You must specify MW in the set per_process memory_limit and set per_request memory_limit directives (the case of the letters is unimportant). If you omit the MW, the amount of memory allocated will be in bytes. If the per_request memory limit for queue little had specified 4, rather than 4mw, queue little would have been restricted to jobs that use 4 bytes or less of memory.

Queue complex limits
14.4.4.2

Another way to limit queues and system resources exists. A subgroup of queues can be considered as a whole, and limits can be applied to that set of queues, so that the pipe queue gateway must consider not only each individual queue's limits, but also the limits imposed on that grouping of queues. This kind of grouping is called a *queue complex*, and limits imposed on such a grouping are called *queue complex limits*.

The sample file does not include a directive to create a queue complex. However, one could be created as described here. For example, queue little has a run limit of 16 and a user limit of 4. Queue medium has a run limit of 8, and a user limit of 2. To create a queue complex composed of queue little and queue medium and name it complex1, use the following qmgr directive:

```
create complex = (little, medium) complex1
```

Even though queue little can run as many as 16 jobs at a time, and queue medium can run as many as 8 jobs at a time (for a possible total of 24 jobs running in both queues at one time),

you can limit the total number of jobs that can run at one time in these queues to 20 by configuring the two queues into a queue complex, as shown in the preceding command line, and setting their combined run limit to be 20 by using the following `qmgr` directive:

```
set complex run_limit = 20 complex1
```

This directive means that if 16 jobs are already running in queue `little`, queue `medium` may run no more than 4 jobs, so that the total number of jobs running in both queues added together will be 20 or fewer. If queue `medium` had 8 jobs running in it, queue `little` may run only 12 or fewer jobs, for a total of 20 in the queue complex `complex1`.

Although queue `little` lets a user submit up to four jobs, and queue `medium` lets a user submit up to two jobs, you can limit the total number of jobs that a user can submit to that queue complex to be 5 by using the following `qmgr` directive:

```
set complex user_limit = 5 complex1
```

This means that if a user has already submitted two jobs to queue `medium`, the user cannot submit more than three jobs to queue `little`, even though queue `little` usually would allow the user to submit four jobs.

Global queue limits 14.4.4.3

One final way exists to assign limits to queues. Limits can be assigned to all queues combined. Such limits are called *global limits*. The sample file limits the total number of jobs that can run on the system at one time to be 18 by using the following `qmgr` directive:

```
set global batch_limit = 18
```

The sample file limits the total number of jobs any one user can submit across all of the queues, added together, to be 4. This limit was set by using the following `qmgr` directive:

```
set global user_limit 4
```

The sample file also limits the total number of jobs that all users in a group can run concurrently to be 8. This limit was set by using the following `qmgr` directive:

```
set global group_limit 8
```

Setting the NQS log file 14.4.5

NQS maintains a log file in which time-stamped messages of NQS activity are recorded. These messages are in text format. To define the path name of the log file, use the following `qmgr` directive:

```
set log_file name
```

The *name* specified must be a full path name; otherwise, `qmgr` returns an error. If you specify the same log file name as the existing log file, a new log file is created and the existing file is overwritten. To display information about the current log file, use the `show parameters` directive.

The sample NQS directives file sets the NQS log file by using the following `qmgr` directive:

```
set log_file /usr/spool/nqs/log
```

Setting the debug level 14.4.6

To control the level of information recorded in the log file, set the debug level for NQS by using the following `qmgr` directive:

```
set debug level
```

Two sets of debug information are supported:

Level	Meaning
0	No debug information is recorded.
1 to 3	When you specify each higher number (level), the amount of debugging information that is recorded increases, respectively.

To avoid excess disk usage, set a nonzero level only when extra information is required to analyze a problem. By default, for a newly installed NQS system, you should set the debug level to 0. To display the current debug level, use the `show parameters` directive.

The sample NQS directives file sets the debug level by using the following `qmgr` directive:

```
set debug 0
```

What if a user submits a job with no `qsub` *directives?*
14.4.7

One final batch queue that is important to configure is the default batch request queue. This queue handles situations in which insufficient `qsub` directives were specified for a job.

This queue is identified to the system as the default batch request queue with the additional `qmgr` command `set default batch_request queue`. The syntax is as follows:

```
set default batch_request queue queuename
```

To equate the default batch request queue with the pipe queue that is defined in the sample file, the following directive is used:

```
set default batch_request queue gateway
```

If the default batch request queue is equated to your pipe queue, each of the following problem situations is handled successfully:

- If no destination queue is specified on an incoming job's `qsub` directives, the job will go to the default batch request queue, which is set to the pipe queue in the sample.
- Typically, the destination queue specified by an incoming job will be the pipe queue. However, if the destination queue specified on an incoming job's `qsub` directives is a batch queue and the batch queues were configured to be `pipeonly`, the user's job will fail.
- If no `qsub` directives indicating job limits were specified (such as expected CPU run time or expected maximum Mwords of memory that the job will use), the pipe queue sees that no job limit directives are specified and submits the job to the first batch queue listed in the `set destination qmgr` directive. In the sample, such a job would be submitted to queue `little`. The job then takes on the default queue limits, which can be identified by using the `qstat -f` command.
- If some, but not all, of the `qsub` directives necessary to determine in which batch queue the job should run were specified, the pipe queue will compare the limits specified by the job's `qmgr` directives to the known limits for each batch queue, starting with the first batch queue defined in the `set destination qmgr` directive. The pipe queue submits the job to the first batch queue defined that has greater limits than the resources requested by the job.

- Whether or not the `qmgr` limit directives were specified, if the resources that the job actually requires fall within the queue limits, the job will run. If the job tries to use more resources of any kind than the queue permits, the job will fail.

Note

If a job fails because it exceeded the allowed resources for the queue it was in, the user should go through the process described in subsection 14.4.10, page 329, to determine the appropriate `qsub` directives for that job.

Turning on and stopping the queues 14.4.8

Make sure that the `NQS.start` file is edited and then start the NQS daemon (for a sample `NQS.startup` file, see page 334). The `qping -v` command displays messages that state whether the local NQS daemon is running. Another way to determine whether the NQS daemon is running is to use the `ps -eaf | grep nqs` shell command, which lists all system processes and greps for the NQS daemon, `nqsdaemon`. If only a system prompt is returned after you type this command, the NQS daemon is not running.

If the daemon is not running, you must perform the following steps:

1. Change the third field of the `/etc/config/daemons` file, the start field, for the daemon with the tag NQS to read `YES`, rather than `NO`. This change causes the NQS daemon to be started automatically every time the system is brought up to multiuser mode.
2. Start the daemon without restarting the system by typing the `/etc/sdaemon -s NQS` shell command.

Then enable and start your NQS queues by using `qmgr` directives. Enable queues one at a time, by using the `enable` directive. For example, you can enable the sample three batch queues (`little`, `medium`, and `large`) and the sample pipe queue (`gateway`) by using the following commands:

```
enable queue little
enable queue medium
enable queue large
```

```
enable queue gateway
```

After the queues are enabled, they can be started, one at a time, with individual directives, such as `start queue little`, or with the following single directive that starts all queues:

```
start all_queues
```

Queues are disabled and stopped with similar commands, such as, `disable queue little` and `stop queue little`. You also can stop all of the batch and pipe queues at once by using the `stop all queues` directive.

Recognizing a "good" NQS configuration
14.4.9

It will take time for you to notice a pattern in how your site uses your CRAY J90 or CRAY EL system. One thing to examine is the average length of time jobs wait in a queue. If jobs in one queue seem to wait a much longer time than jobs in any other queue, or if some queues hardly ever seem to be used, it might indicate that you should change the granularity of your queue limits.

If daemon accounting was turned on and if the following `qmgr` directive was configured, the Cray system accounting (CSA) software reports average NQS queue wait times and other individual batch and pipe queue statistics:

```
set accounting on
```

Setting job limits for a job
14.4.10

To set job limits for a job, you must instruct your users to do the following:

1. To enter the NQS `qmgr` database, type `qmgr`.
2. Display all `qmgr` configuration information, including queue limits, by typing `show all`. (If you are not in `qmgr`, the `qstat -f` command also will show you this information.)
3. The first time a user submits a job, if the user does not know what limits to set (how long the job might take, how much memory it might use, and so on), the user should set the limits on the job so that the job falls into the batch queue that allows the most system resources to be used by a job. This ensures that the job will definitely run to completion.

4. All job output returns to the directory that the user was in when the job was submitted. It is important that your users know where to look for their job output. When a request completes processing, the standard output, standard error, and job log files associated with the request are returned to the user. To configure the default return of a request's job log file on or off, use the `qmgr set default job_log on|off` command.

   When the job output has returned, the user should examine the job for the actual amount of time, memory, and other resources consumed by the job.

   If a job cannot run, email is sent to the user who submitted the job.

5. Most sites put the highest priorities on the smallest queues. At this point, the user has enough information to submit the job with appropriate `qsub` directives setting limits high enough to allow the job to "grow" but low enough so that the job fits into the smallest queue possible (which probably has a higher priority than the next larger queue).

Sample NQS directives file 14.4.11

A sample NQS directives file follows. You can use these sample NQS directives to create the basic NQS described in this section of the guide, rather than using the more complex standard UNICOS `/etc/nqs_config` file. This sample may be simplistic compared to the configuration you may want to use in your production environment. The sample does not include the network feature, the URM feature, or several optional `set` directives such as `set per_request tape_limit`. For this type of information, see *UNICOS System Administration*, publication SG–2113.

Note

If you choose to use this sample NQS directives file to create your basic NQS configuration as the `/etc/config/nqs_config` file, you also must modify your `NQS.startup` file to match your new `/etc/config/nqs_config` file, as shown in the sample file in subsection 14.4.12, so that the default `NQS.startup` file definitions do not override your new NQS configuration. Your `NQS.startup` file definitions should always match your `/etc/config/nqs_config` file.

```
# The following defines machine identification (mid) specification for the CRAY J90 or
# CRAY EL system; always put the mid definition for the host node first. You can add other
# systems by using the same syntax.#
#
add mid 100 hostA
#
# The following defines user(s) to be NQS managers/operators; m = manager and
# o = operator.
#
add managers userA:m
add managers ops:o
#
# The following creates the batch queues and sets the defined attributes.
#
create batch_queue little priority=55 pipeonly
set queue run_limit=16 little
set queue user_limit=4 little
set queue group_limit=8 little
set per_request cpu_limit=100 little   (per-process = "piece of a job")
set per_process cpu_limit=70 little
set per_request memory_limit=4mw little        (You MUST specify mw (in any
set per_process memory_limit=2mw little        combination of uppercase and
#                                              lowercase) or limit will be in
#                                              terms of bytes, not words)
create batch_queue medium priority=45 pipeonly
set queue run_limit=8 medium
```

```
set queue user_limit=2 medium
set queue group_limit=3 medium
set per_request cpu_limit=600 medium
set per_process cpu_limit=570 medium
set per_request memory_limit=8mw medium
set per_process memory_limit=4mw medium
#
create batch_queue large priority=35 pipeonly
set queue run_limit=4 large
set queue user_limit=1 large
set queue group_limit=2 large
set per_request cpu_limit=unlimited large
set per_process cpu_limit=unlimited large
set per_request memory_limit=unlimited large
set per_process memory_limit=16mw large
#
# The following creates the pipe queue gateway and sets the defined attributes; the
# server path is the default location where NQS installs its pipe client.
#
create pipe_queue gateway priority=60 server=(pipeclient)
set queue run_limit=1 gateway
#
# The following sets the destination list for the pipe queue "gateway." The order is important:
# the system follows this order to determine into which queue an incoming job will be
# allowed; see subsection 14.4.3, page 322.)
#
set destination = (little, medium, large) gateway
#
# The following sets the NQS log file path name.
#
set log_file /usr/spool/nqs/log
#
# The following sets the log segments directory.
#
set segment directory /usr/spool/nqs/log_segments
#
# The following segments the log file at NQS startup.
#
set segment on_init on
#
# The following enables NQS daemon accounting; you must have the accounting daemon
# running and the accounting software configured to use this directive.
#
set accounting on
#
```

```
# Set desired global limits.
#
set global batch_limit = 18
set global user_limit = 4
set global group_limit = 8
#
# Set debug level.
#
set debug 0
#
# Set the log message header format.
#
set message_header long
#
# The following sets the checkpoint directory; NQS places all checkpoint files (created with
# the qchkpnt command at the time of NQS shutdown) into this directory. See subsection
# 14.7, page 351. This directory must reside on a file system that has sufficient space to
# contain a checkpoint image of every running batch job.
#
set checkpoint_directory /usr/tmp/nqs.chkpt
#
# Specifies the default batch request queue.
#
set default batch_request queue gateway
#
# Sets the job log return default.
#
set default job_log on
#
# The following enables and starts all queues.
#
enable queue little
enable queue medium
enable queue large
enable queue gateway
start all_queues
```

Sample `NQS.startup` *file* 14.4.12

A sample `NQS.startup` file follows. If you use the sample NQS directives file to create your basic NQS, you also should change the `/etc/config/NQS.startup` file that is provided on your system. See the following sample file and embedded comments.

```
/etc/config/NQS.startup# USMID @(#)nqs/example/NQS.startup
#
#   NQS.startup - Qmgr commands to start up NQS.
#
#   Example of site configurable script that resides in /etc/config;
#   /usr/bin/qstart invokes /etc/config/NQS.startup unless
#   -i option is specified.
#
#
#   Set general parameters.
#
#   Fewer time outs on qstat, qmgr, etc, if nqsdaemon locked in memory.
#
lock local_daemon
#
#   Site might not choose to run accounting during nonprime time to save disk
#   space.
#
set accounting on
#
#   Could have put checkpoint directory in /tmp/nqs/chkpnt for a benchmark;
#   so make sure it is in usual place for prime time.
#
set checkpoint_directory =(/usr/spool/nqs/private/root/chkpnt)
#
#   Debug level may have been set at 3 (verbose) for testing; reduce log
#   file size for prime time by reducing level.
#
# BELOW IS ALSO SET IN CONFIG FILE; LEVEL SHOULD MATCH
#
#
set debug 0
#
#   The selected priority determines the relative ordering of requests
#   moving between batch queues, not the execution-time priority.
#
set default batch_request priority 30
#
#
```

```
set default batch_request queue gateway
#                                ^^^^^ SHOULD = PIPE QUEUE NAME
#
#   NEXT THREE SETTINGS ARE OPTIONAL BUT WILL NOT AFFECT SET UP
#
#   Number of hours during which time a pipe queue can be unreachable;
#   a routing request that exceeds this limit is marked as failed.
#
set default destination_retry time 72
#
#   Number of minutes to wait before trying a pipe queue destination
#   that was unreachable at the time of the last attempt.
#
set default destination_retry wait 5
#
#   Number of hours that any request can reside within a pipe queue,
#   This parameter exists to prevent requests from filling up all
#   available queue space.
#
set lifetime 168
#
#   Might want to prevent log filling up /usr/spool/nqs.
#   Has to be absolute path name.
#
#  NEED NEXT LINE BUT MAY WANT TO CHANGE PATH
#
set log_file =(/tmp/nqs/nqs_log)
#
#   Assumes NQS account exists on CRAY J90 or CRAY EL system and belongs
#   to group root.
#   If -me or -mb on qsub this is who mail will be from.
#
#  YOU MAY WANT TO ADD OR SUBSTITUTE "root" TO BELOW
#
set mail nqs
#
#   Default shell strategy is FREE; want /etc/passwd (that is, udb) control
#   instead.
#
#   BELOW IS OPTIONAL
#
set shell_strategy login
#
```

```
#  BELOW MIGHT ALSO BE SET IN CONFIG FILE - MAKE THEM MATCH OR ELIMINATE
#  THEM FROM HERE
#
#   Now set global parameters.
#
set global batch_limit = 20
set global user_limit = 20
set global group_limit = 15
#
#   Consider this number an "oversubscription"; it does not have to be
#   the size of the machine.
#
# BELOW MIGHT ALSO BE SET IN CONFIG FILE - MAKE THEM MATCH OR ELIMINATE THEM
# FROM HERE
# BELOW NUMBER SHOULD = (available user memory as displayed during
#   "load" command output + size of swap device)
#
set global memory_limit = 256Mw
#
#   Consider this number an "oversubscription"; it does not have to be
#   the size of the SSD.  (CRAY J90 and CRAY EL systems do not support SSDs.)
#
# BELOW IS ALSO SET IN CONFIG FILE - ELIMINATE IN BOTH PLACES.  CRAY J90 and
# CRAY EL SYSTEMS DO NOT SUPPORT QUICKFILES. CRAY X-MP and Y-MP systems only.
#
set global quickfile_limit = 1Gw
#
# BELOW MIGHT ALSO BE SET IN CONFIG FILE - MAKE THEM MATCH OR ELIMINATE THEM
# FROM HERE.
#   The numbers in each tape group must reflect local site configuration.
#
set global tape_limit a = 10
set global tape_limit b = 5
set global tape_limit c = 5
set global tape_limit d = 0
set global tape_limit e = 0
set global tape_limit f = 0
set global tape_limit g = 0
set global tape_limit h = 0
```

```
#
#    Maximum number of pipe requests allowed to run concurrently.
#
#   BELOW IS OPTIONAL.
#
set global pipe_limit = 5
#
#   BELOW SHOULD BE ELIMINATED – CRAY J90 and CRAY EL systems do not support
quickfiles.
#
#   Quickfiles apply only to CRAY Y-MP systems.
#  These parameters must be set at every NQS or qfdaemon initiation.
#
set quickfile polling_interval = 150
set quickfile residence_interval = 600
#
#    Capture configuration status at startup; written to
#    /usr/lib/nqs/qstart.out unless otherwise specified.
#    These commands are entirely optional, but handy to refer to if there are problems.
#
sho queues
sho parameters
sho mids
sho man
#
#    Starts all queues.
#
start all_queues
#
#   Now NQS will begin scheduling jobs.
```

Procedure: Configuring and starting your local host NQS system

Note

To submit NQS configuration directives to `qmgr`, you must have manager (:m) level of authority. By default, `root` is already configured as an NQS manager on your system.

When your system arrives, the NQS database is empty. To get NQS to a functioning state, you must do the following:

1. Place directives into the empty NQS database
2. Turn on the NQS daemon
3. Enable and start the NQS queues

A set of directives is provided on your system; these directives are in the `/etc/nqs_config` file. Before you configure your local NQS system, you should determine whether this set of directives will meet your site's needs. You should set up one small queue (for example, 10 seconds and 1 Mword or fewer). In addition, review the directives from the Sample NQS Directives File, page 330, and consider needs such as the following (for both memory and CPU time) to help you determine your NQS configuration:

- The largest job to be run at one time
- The average number of jobs that will run at one time
- The number of large jobs that will run at one time
- The available user memory
- The available space on your swap device

1. Do one of the following:
   - Copy the `/etc/nqs_config` file into `/etc/config/nqs_config`
   - Create a new `/etc/config/nqs_config`

Note

Either method leaves the original `/etc/nqs_config` file unchanged so that you can refer to it in the future. The menu system and the `qstart` script use the `/etc/config/nqs_config` file as input.

2. Configure NQS by either using the menu system or doing it manually.

   If you are using the menu system, select the `Configure System ==> NQS Configuration` menu. To use the default NQS configuration, select and activate the `Import NQS configuration ...` and `Activate NQS configuration ...` actions of the menu.

   If you want to modify the default configuration, first import the default NQS configuration, then select the `Edit NQS configuration ...` menu, and then enter your changes. After you have made your changes, activate your configuration. After you have activated the configuration, the NQS daemon will use it. A sample `NQS Configuration` menu screen follows:

```
Configure System
...NQS Configuration
```

```
                              NQS Configuration

 S-> Edit NQS configuration ...
     Import NQS configuration ...
     Activate NQS configuration ...
```

Important

In the main `Configure System` menu, always keep `CONFIG NQE` set to `ON` even if you are not running NQE; NQS requires this setting.

If you are running the Network Queuing Environment (NQE) and want to modify your default NQS configuration, you should first check the variables listed in the Network Queuing Environment menu; the NQE variable settings override NQS settings.

   If you are not using the menu system, edit the set of directives in the `/etc/config/nqs_config` file to reflect your site's needs.

3. Submit your directives to the empty database by entering the following command:

```
qmgr < /etc/config/nqs_config
```

4. Start the NQS daemon by either using the menu system or doing it manually.

   If you are using the menu system, to start the NQS daemon automatically on subsequent system restarts, select the `Configure System ==> System Daemons Configuration ==> System Daemons Table` menu, change the `StartOpts` field of the NQS daemon to be

YES, and update the form file. Then activate your changes through the System Daemons Configuration menu. All daemons that have YES in the Start up at boot time? field will be started automatically in subsequent system startups. A sample System Daemons Table submenu screen for the NQS daemon follows:

```
Configure System
...System Daemons Configuration
......System Daemons Table
```

```
                          System Daemons Table

S-> Group                                        SYS2
    Name                                         NQS
    Start up at boot time?                       YES
    Kill action                                  /usr/bin/qstop
    Executable pathname                          /usr/bin/qstart
    Command-line arguments                       -i /etc/config/nqs_config
    Additional command-line arguments
    Additional command-line arguments
```

If you are not using the menu system, to start the NQS daemon automatically on subsequent system startups, set the third field of the /etc/config/daemons file to YES so that the NQS daemon is started automatically at system startup. An example of the NQS daemon line from the /etc/config/daemons file follows:

```
SYS2 NQS YES /usr/bin/qstop /usr/bin/qstart -i /etc/config/nqs_config
```

To start NQS now (manually), you can use the following command. By default, the following command reads the NQS line of the /etc/config/daemons file and starts NQS based on the command in column 5 and all arguments in column 6 (if you are using the menu system, the command in column 5 and arguments in column 6 are the "Executable pathname" and "Command-line arguments" entries shown in the "Systems Daemons Table" menu shown in this step):

```
sdaemon -s NQS
```

Note

If you included the enable and start directives in the file you submitted to qmgr in step 3, you do not have to do steps 5 and 6.

5. If you did not embed batch queue and pipe queue `enable` and `start` directives in the directives file that you placed in the empty NQS database in step 3, you must enter the `qmgr` and submit one `enable` directive for each queue you want to enable. Example:

```
qmgr
enable queue little
enable queue medium
enable queue large
enable queue gateway
```

6. After the queues are enabled, you can start them by submitting either individual directives, such as `start queue little`, or the following single directive that starts all queues:

```
qmgr
start all_queues
```

Note

After you have configured and started NQS, you should make subsequent database changes interactively using the `qmgr` interface. After changes are made, you can use the `qmgr snap` command to save the current configuration. To update the standard `/etc/config/nqs_config` file, you can replace it with the snap file (see the procedure for obtaining an image of the NQS configuration, page 345).

Important

Always keep the `/nqe` directory and nmake file on your system even if you are not running NQE; NQS uses files in the `/nqe` directory.

Procedure: Adding and removing qmgr managers and operators

By default, `root` is already configured as an NQS manager on your system.

- To add other `qmgr` managers, use the following `qmgr` directive:

  ```
  add managers name:m
  ```

  Example:

  ```
  qmgr
  add manager mark:m
  quit
  ```

- To add other `qmgr` operators, use the following `qmgr` directive:

  ```
  add managers name:o
  ```

  Example:

  ```
  qmgr
  add manager jane:o
  quit
  ```

- To remove `qmgr` managers and operators, use the following `qmgr` directives:

  ```
  delete managers name:m

  delete managers name:o
  ```

  Example:

  ```
  qmgr
  delete manager mark:m
  delete manager jane:o
  quit
  ```

Procedure: Obtaining an image of the NQS configuration

The `snap` command of the `qmgr` utility captures an image of the current NQS configuration. The command places this information (consisting of a set of `qmgr` commands) into a specified file.

You can specify this file on the `snap` command itself, or you can specify a file by using the `set snapfile` command. You should place the `set snapfile =` *absolute-pathname* command in the `NQS.startup` file.

Note

It is important that you execute the `snap` command after you interactively modify the NQS configuration file. This gives you a record of the most recent changes to the file. You can then replace the `/etc/config/nqs_config` file with the `snap` file to update the standard NQS configuration file.

You must use `snap` only after interactive changes were made, so that you have a permanent record of the current NQS configuration. If you used the following steps to make changes your file is already updated; you do not have to use the `snap` command:

1. `vi` *file*
2. `qmgr` *file*

To obtain a snapshot of the NQS configuration, enter these commands:

```
qmgr
snap f = absolute-file-path
quit
```

Procedure: Shutting down NQS

You can shut down NQS without shutting down your entire system by using the `-k` option of the `sdaemon` command.

Example:

```
/etc/sdaemon -k NQS
```

To shut down NQS during a system shutdown, the following steps are involved:

1. The `qstop`(8) command is called from the system shutdown script. By default, `qstop` takes its input from `/etc/config/NQS.shutdown`.
2. The `qstop` command sends `stdout` and `stderr` to `/usr/lib/nqs/qstop.out`.
3. The `/etc/config/NQS.shutdown` script issues the `qmgr` commands `stop all` and `shutdown` (see the following example).

A sample NQS shutdown file follows:

```
cat /etc/config/NQS.shutdown

# USMID @(#)nqs/example/NQS.shutdown
#
#       (C) COPYRIGHT CRAY RESEARCH, INC.
#       UNPUBLISHED PROPRIETARY INFORMATION.
#       ALL RIGHTS RESERVED.
#
#
#   NQS.shutdown - Qmgr commands to shutdown NQS.
#
#   Example of site configurable script that resides in /etc/config;
#   /usr/bin/qstop invokes /etc/config/NQS.shutdown unless
#   -i option specified.  (The -i option specifies an alternative input file.)
#
stop
allset accounting off
#
#   The 60 second grace period means shutdown will take longer than 60
#   seconds. The nqsdaemon sends a SIGSHUTDN signal to the processes
#   of all running requests and then waits for the number of seconds
#   specified by the grace period. After the grace period, nqsdaemon
#   tries to checkpoint all running requests that are restartable;
#   i.e., QSUB option -nc NOT specified. nqsdaemon will send SIGKILL to
#   processes of requests that were not checkpointed, including those
#   for which the checkpoint failed. All checkpointed requests and
#   rerunnable requests (i.e., QSUB option -nr NOT specified), will be
#   requeued to be restarted or rerun when NQS is next initiated.
#
shutdown 60
```

When the `qmgr shutdown` command shuts down NQS, all processes that make up a restartable running batch request on the local host are suspended, and NQS saves an image of the request (using the `chkpnt(2)` system call) in a file in the checkpoint directory.

For a batch request to be considered restartable, it must meet the recoverability criteria described in subsection 14.6, page 350.

When NQS is restarted, checkpointed requests are recovered from the images in their respective restart files. They resume processing from the point of suspension before the shutdown. After a request is restarted successfully, the restart file is kept until the request completes (in which case the file is removed), or the request is checkpointed again (in which case, the file is overwritten).

NQS periodic checkpointing
14.5

NQS periodic checkpointing provides transparent automatic checkpointing of NQS requests. The checkpoint files restart NQS requests from the time the checkpoint file was created. Use the `qmgr set periodic_checkpoint` commands to checkpoint the requests at intervals that are site-controlled. See the `qmgr`(8) man page for more information. Periodic checkpointing initiates a checkpoint for all NQS requests based on the amount of CPU time that is used or the amount of wall-clock time that has elapsed. An NQS administrator can tailor the following periodic checkpoint options:

- Enable and disable periodic checkpointing for all NQS requests
- Enable and disable periodic checkpoint intervals based on the CPU time
- Enable and disable periodic checkpoint intervals based on the wall-clock time
- Exclude short running requests
- Exclude large memory requests
- Set an interval for periodic checkpoints based on the CPU and wall-clock time
- Set the maximum number of concurrent periodic checkpoints
- Define an interval to check eligible requests

Users can enable and disable periodic checkpointing during the life of a job by using the `qalter` command in jobs. For more information, see the `qalter`(1) man page.

Periodic checkpoint file restrictions
14.5.1

The periodic checkpoint/restart file cannot be a network file system (NFS) file, but a process with open NFS files can be checkpointed and restarted.

If a process is checkpointed with an open NFS file and that file is on a file system that was manually mounted, then that file system must also be manually mounted when the process is restarted or the restart will fail.

Periodic checkpoint modes
14.5.2

You can set your periodic checkpoint environment to one of the following modes:

- Compatibility mode
- User-initiated mode
- CPU time mode
- Wall-clock mode
- CPU and wall-clock mode

For more information about NQS periodic checkpointing, see *UNICOS System Administration*, publication SG–2113, which describes how to set each periodic checkpoint mode of operation by using the NQS `qmgr`(8) and `qalter`(1) commands.

Conditions for recovery of NQS requests
14.6

When the NQS daemon is instructed to shut down, it tries to create a checkpoint file for each currently running batch request.

When NQS is restarted (with a start-up script that contains the `qstart` command), all checkpointed jobs are recovered and continue running until completion. A user may specify *no recovery* for a batch request, causing NQS to rerun the request when started again. If the user also specifies *no rerun*, the job is aborted.

Users can restart NQS requests only if **all** of the following conditions are satisfied:

1. No open TCP/IP socket connections exist.
2. All files in use reside on the local host.
3. No tape devices are reserved.

For additional system-level reasons why NQS requests might not be able to be restarted, see the `restart`(2) system call man page.

NQS user commands
14.7

The NQS user commands are as follows:

Command	Description
`qalter`	Alters the attributes of one or more jobs.
`qchkpnt`	Creates a checkpoint image of a running request.
`qdel`	Deletes or signal NQS requests.
`qlimit`	Displays the batch limits and shell strategy of an NQS host.
`qmsg`	Writes a message to the `stdout` or `stderr` file of a running request and writes a message to the job log file of a queued or running request.
`qping`	Determines whether the local NQS daemon is running and responding to the requests.
`qstat`	Displays the status of NQS queues, requests, and complexes.
`qsub`	Submits a batch request to NQS.

The `qsub`(1) command submits a batch request to NQS. Following are options commonly used with this command:

Option	Description
`-a` *date–time*	Runs request after stated time.
`-eo`	Combines the job's standard error output (`stderr`) with the job's standard (`stdout`) output in the `stdout` file.
`-lt` *limit*	Specifies a CPU time limit, in seconds, for each process generated in the batch request.
`-me`	Causes NQS to send an electronic mail message to the owner of the job when the request completes execution.
`-mu` *username*	Causes NQS to send an electronic mail message to the specified user when the request completes execution.

Any option to qsub that begins with the letter l (signifying *limit*) allows users to specify a resource requirement for a batch job (for example, CPU time limit). If the user does not specify batch request resource limits, the job goes into a default queue, assuming one was configured (see subsection 14.4.7, page 327), and is subject to the resource limits set for that queue. The resource requirements for the default queue may not be sufficient for the successful completion of the user's job. Therefore, you should specify resource requirements for each batch job.

Users can submit jobs by using qsub in the following ways:

- Include #QSUB directives embedded at the beginning of the job file (see example 1 that follows)
- Specify options on the command line (see example 2 that follows)

Example 1: Using qsub, include #QSUB directives in the job file:

Job myjob1 has the following contents:

```
#QSUB -q gateway    # request is submitted to pipe queue
#QSUB -r myjob1     # request name is bigjob
#QSUB -lU 2         # request two tape drives
#QSUB -lM 50Mw      # request 50 Mwords of memory
#QSUB -lT 100       # request 100 CPU seconds for this job
cft77 crash.f       # compile a program
segldr crash.o      # load (link) the program
a.out               # execute the binary output of the load
```

You submit the job as follows:

```
unicos$ qsub myjob1
Request 10764.sn1101 submitted to queue: batch.
```

Example 2: Using qsub, specify options on the command line:

Job myjob2 has the following contents:

```
cft77 crash.f       # compile a program
segldr crash.o      # load (link) the program
a.out               # execute the binary output of the load
```

You submit the job as follows (gateway is the name of the pipe queue):

```
unicos$ qsub myjob2 -r myjob2 -lU 2 -lM 50Mw lT 100 myjob2 -q gateway
Request 10763.sn1101 submitted to queue: batch
```

NQS qstat -a command

14.8

The output of `qstat -a` shows some very valuable information about user jobs. The following information refers to the output, as follows:

```
--------------------------------
NQS 8.0.2 BATCH REQUEST SUMMARY
--------------------------------
IDENTIFIER    NAME      USER    QUEUE       JID  PRTY  REQMEM   REQTIM   ST
----------    ----      ----    -----       ---  ----  ------   ------   --
31099.sn414   testjob   tim     small       351    25      97       88  R02
31097.sn414   test      phil    medium      348    32     807      520  R04
31102.sn414   myjob1    phil    medium      350    32     807      949  R02
31117.sn414   myjob2    phil    medium          63     807     1000  Qqr
31098.sn414   test1     jan     large           63    9216     9000  Qqs
31003.sn414   another   mary    huge        108    63    5000    50000  S02
```

Heading	Description
IDENTIFIER	Request ID with appended name of host from which it was submitted; the ID (without the host name) is used by many NQS commands to refer to specific requests.
NAME	Request name of the job taken from `#QSUB -r` *name*; if not specified, the script name is used.
USER	Login name of the user who submitted the request.
QUEUE	Name of the queue in which the request currently resides.
JID	UNICOS job identification number; use this number with the `jstat` command.

Heading	Description
PRTY	Priority of the request when in the queue; the nice value of the request if running.
REQMEM	The number of kilowords of memory the request is currently consuming.
REQTIM	The number of seconds of CPU time remaining for the request.
ST	The state of the request; major and minor status. For additional information, see the qstat man page.

NQS user messages
14.9

The NQS user error messages have a group code of nqs and are located in the /usr/src/net/nqs/nqs.msg file. To display the explanation of a message, use the explain command (for more information, see the explain(1) man page). The following example shows some NQS error messages and the use of the explain command:

```
% qmsg -j -f msg_in 10
nqs-272 qmsg:  WARNING
  Error opening file </abc/u0/xyx/msg_in>, errno = <13>.
nqs-20 qmsg:  WARNING
  Errno <13> = <Permission denied>.
$ explain nqs272
Error opening file <file>, errno = <errno>.

NQS tried to use the open(2) system function to open file <file>,
but failed.  The reason for the failure is identified by <errno>.
% explain nqs20
Errno <errno> = <error_desc>.

The specified error <errno> has an associated description of
<error_desc>.
%
```

NQS user exits
14.10

The NQS user exits allow the system administrator to introduce local code at defined points (compatible from release to release) and allow users to customize NQS without having access to the source.

The user exits allow sites to tailor various functions of NQS, including queue destination selection, `qsub` option preprocessing, job startup and termination processing, and NQS startup and shutdown processing.

NQS user exits are used for the following functions:

- NQS daemon packet. Allows sites to add functionality when a packet arrives.
- NQS destination ordering. Allows sites to control the order of pipe queue destinations within NQS.
- NQS fair-share priority calculation for the *share_pri* value.
- NQS job selection. Allows sites to determine whether the request chosen by NQS should be started. Users can customize the order in which NQS starts requests.
- NQS job initiation. Allows a user exit before job initiation.
- NQS job termination. Allows a user exit after job termination.
- NQS `qmgr` command. Allows additional functionality when a `qmgr` command will be processed.
- `qsub` directives. Allows user exits before the first `#QSUB` directive, on each `#QSUB` directive, and after the last `#QSUB` directive.
- NQS startup. Allows users to perform processing during the NQS startup.
- NQS shutdown. Allows users to perform processing during the NQS shutdown.
- Job submission. NQS uses the new centralized user password identification and authentication (I&A) routines. The user exits that are a part of the new validation routines allows sites to implement their own NQS user identification and validation algorithms.

For additional information about NQS user exits, see *UNICOS System Administration*, publication SG–2113.

Tape Subsystem [15]

Warning

This manual does not contain any Trusted UNICOS information. If your site is running a Trusted UNICOS system, you must refer to *UNICOS System Administration*, publication SG–2113, and remain within the constraints outlined there, to maintain the Trusted UNICOS environment.

CRAY J90 and CRAY EL systems are designed with a tape subsystem that provides you with two methods of accessing your tape devices. One method uses the feature-rich `tpdaemon` tape daemon software provided as part of a standard UNICOS software release. This method offers features such as multivolume support, labeled tapes, tape silo access, and data buffering. The tape daemon also offers users a lot of tape device control, flexibility, and security. The second method is similar to the traditional UNIX approach of accessing tape devices by using character special files.

Use of the UNICOS tape daemon requires you to specify a tape daemon configuration file. The tape daemon creates and maintains special files in the UNICOS `/dev/tape` directory. Users can make tape device requests through the tape daemon, which actually manages the devices. The UNICOS `tpdaemon` is the way most users will probably want to access and manipulate tape devices.

As of the UNICOS 8.0.3 release, a new character special tape file interface is supported on CRAY J90 and CRAY EL systems. To create character special tape files, execute the `tpdaemon` command. The `tpdaemon` command creates a file for each device defined in the tape configuration file (`/etc/config/tapeconfig`). These files reside in the `/dev/tape` directory. Before terminating, the `tpdaemon` command creates a detached process that is used to assist the tape driver. If tape devices will be accessed using only a character special interface, you may terminate the process by using the `tpdstop` command. You can restart the tape daemon as long as all character special device files are closed.

Note

Unlike the CRAY EL character special tape file interface that uses the `/etc/mknod` command, the new character special tape file interface and the tape-daemon-assisted interface may operate concurrently. Devices for both interfaces are defined in the same configuration file and are defined identically; that is, the interface is not identified in the configuration file.

Note

This character special files interface will be supported until the UNICOS 9.0 release. As of the UNICOS 9.0 release, only the character special files interface that uses the tape daemon (`tpdaemon`) will be supported.

The type of interface used is identified when the device is opened. If a device file residing in the `/dev/tape` directory is opened, the character special interface is used. After it is opened, the tape daemon cannot access the device until it is closed. If a device will be accessed using the daemon-assisted interface, you must configure the device up by using the `tpconfig` command. A device is not accessible to the character special interface while the device is configured up.

For information about using the character special tape file interface that uses the `tpdaemon` command, see the UNICOS 8.0.3 online version of the *UNICOS Tape Subsystem User's Guide*, publication SG–2051. This publication can be accessed through the CrayDoc and Docview products.

CRAY EL systems also support a character special tape interface that uses the `mknod` UNICOS utility. This second method is similar to the traditional UNIX approach of accessing tape devices by using character special files. The character special file tape interface permits standard UNIX control over tape devices. To create these character special files in the UNICOS `/dev` directory, use the standard `mknod` UNICOS utility. The UNIX path to tapes does not provide tape configuration management, security, labels, multivolume support, or any of the tape daemon features. This path should never be available if the tape can be accessed through the tape daemon.

Users can use tapes with either UNIX commands or UNICOS `tpdaemon` commands. You should use tape daemon commands because you gain general security, configuration management, labeled tapes, and so on.

Warning

On CRAY EL systems, it is **not** recommended that a given tape drive be configured up in the tape daemon if it also has a UNIX character special file defined for it in the `/dev` directory. To avoid interference between the tape daemon and the standard UNIX tape access commands, you should define each tape device for **only one** of these two access paths:

- Tape daemon: Define a tape device for tape daemon usage by including it in your `/etc/config/tapeconfig` file (see subsection 15.4, page 368).
- Standard UNIX commands: Define a tape device for standard UNIX tape command access by making a UNIX character special file for it in the `/dev` directory (see subsection 15.3, page 362).

Interference between the tape daemon and the standard UNIX tape commands for a given device may cause severe problems, including data corruption, system hangs, or system panics.

On CRAY EL systems, after you have loaded your release tapes, you may want to delete, rename, or restrict permissions on the tape special files from the `/dev` directory, and thus limit your users to using only UNICOS `tpdaemon` commands to avoid confusion and possible usage contention.

This section provides information about creating character-special files by using the `/etc/mknod` command for CRAY EL systems. For information about using the character special tape file interface that uses the `tpdaemon` command, see the UNICOS 8.0.3 online version of the *UNICOS Tape Subsystem User's Guide*, publication SG–2051.

This section also describes how to configure tape devices accessible by the UNICOS tape daemon (`tpdaemon`) for CRAY J90 and CRAY EL systems.

Related tape subsystem documentation
15.1

The following documentation contains detailed information about the topic covered in this section:

- *UNICOS Administrator Commands Reference Manual,* publication SR–2022: `tar`(8), `tpapm`(8), `tpbmx`(8), `tpclr`(8), `tpconfig`(8), `tpdaemon`(8), `tpdev`(8), `tpdstop`(8), `tpfrls`(8), `tpgstat`(8), `tplabel`(8), `tpmls`(8), `tpmql`(8), `tpscr`(8), and `tpset`(8) man pages
- Tape Subsystem section in *UNICOS System Administration*, publication SG–2113
- *UNICOS Tape Subsystem User's Guide*, publication SG–2051
- *UNICOS User Commands Reference Manual*, publication SR–2011: `cpio`(1), `dd`(1), `mt`(1B), `rls`(1), `rsv`(1), `tpcatalog`(1), `tplist`(1), `tpmnt`(1), `tprst`(1), and `tpstat`(1) man pages

For information about cleaning and maintaining your tape device, see appendix D, page 431.

The `/bin/file` command
15.2

The `/bin/file` command, among other things, causes major and minor device numbers, and device-specific parameters of character and block special devices to be printed in decimal.

The standard UNIX commands must address the tape devices through a path different from the one that the `tpdaemon` tape commands use. To see whether your tape devices are configured in that directory, change (`cd`) to the `/dev` directory and use the `/bin/file` command.

The following is an example of `/bin/file` output that shows `/dev` tape special files:

```
# /bin/file /dev/r*
/dev/rpe02:          character special (43/1)    0 8 65 0 2 0 0 0
/dev/rss000:         character special (43/7)    1 10 6 0 4 0 0 0
/dev/rpd03:          character special (43/5)    0 8 68 0 3 0 0 0
#
```

Note

The major number for all tape special files in the `/dev` directory will always be 43.

If the `/bin/file` command shows that the special file configuration information fields are all zeros (except for major and minor numbers), you must manually create the tape special files in the `/dev` directory for the standard UNIX commands.

The following example details the fields displayed by executing the `/bin/file` command on a tape device in the `/dev` directory:

```
# /bin/file /dev/rpe02
rpe02:          character special (43/1)     0 8 65 0 2 0 0 0
                                  :  :       : : :  : : : : :
                                  :  :       : : :  : : : : :
The fields are as follows:        :  :       : : :  : : : : :
                                  :  :       : : :  : : : : :
major number ....................:  :       : : :  : : : : :
minor number ....................:        : : : :  : : : : :
number of IOS device is attached to .....: : :  : : : : :
controller type ............................: :  : : : : :
device type ................................:  : : : : :
number of controller device is attached to .....: : : : :
device number ..................................: : : :
logical device or unit number ....................: : :
Rewind device ..................................: :
Density ........................................:
```

When a `/bin/file` is performed on the tape special files in the `/dev/tape` directory (which the UNICOS `tpdaemon` accesses), only the major and minor numbers are ever displayed. The UNICOS `tpdaemon` handles the rest of the configuration information in a different way than with UNIX tape configuration. Be aware that in the process of doing an installation or an upgrade, when loading `root (/)` and `/usr` from the release tape, you are in single-user mode and the `tpdaemon` is down. This means that no tape devices have been configured in the `/dev/tape` directory.

Following is an example of `/bin/file` output that shows `/dev/tape` (tape daemon's) special files:

```
# cd /dev/tape
# /bin/file r*
rpe02:          character special (11/1)    0 0 0 0 0 0 0 0
rpq01:          character special (11/0)    0 0 0 0 0 0 0 0
rss000:         character special (11/3)    0 0 0 0 0 0 0 0
```

Note

The major number for the tape special files in the `/dev/tape` directory will always be 11.

Creating character special files on CRAY EL systems by using the `/etc/mknod` command

15.3

CRAY EL systems can use the `/etc/mknod` command to create character special files. To add tape devices that the tape daemon will not control, you can configure the additional tape devices by creating additional character special files. These additional character special files establish a connection between an arbitrary path name and the actual specified device. Each special file resides under the UNICOS `/dev` directory. The following subsections describe creating character special files to support UNIX tape command access of devices on your CRAY EL system.

Note

This character special files interface will be supported until the UNICOS 9.0 release. As of the UNICOS 9.0 release, only the character special files interface that uses the tape daemon (`tpdaemon`) will be supported.

Components of character special files 15.3.1

Each character special file consists of a special file name and device information. The `/etc/mknod` UNICOS command is used to create these character special files. The following command line example creates a new character special file under the `/dev` directory. The special file name is defined first, followed by the device information:

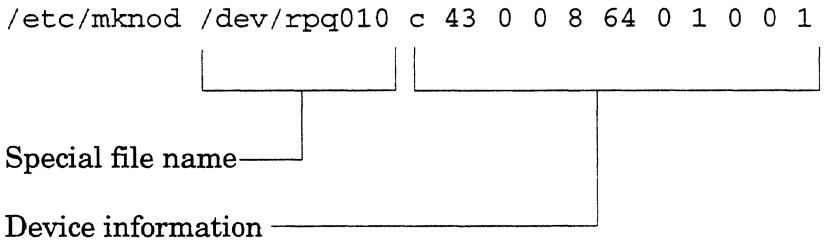


Figure 2 shows the components of special file names and device information.

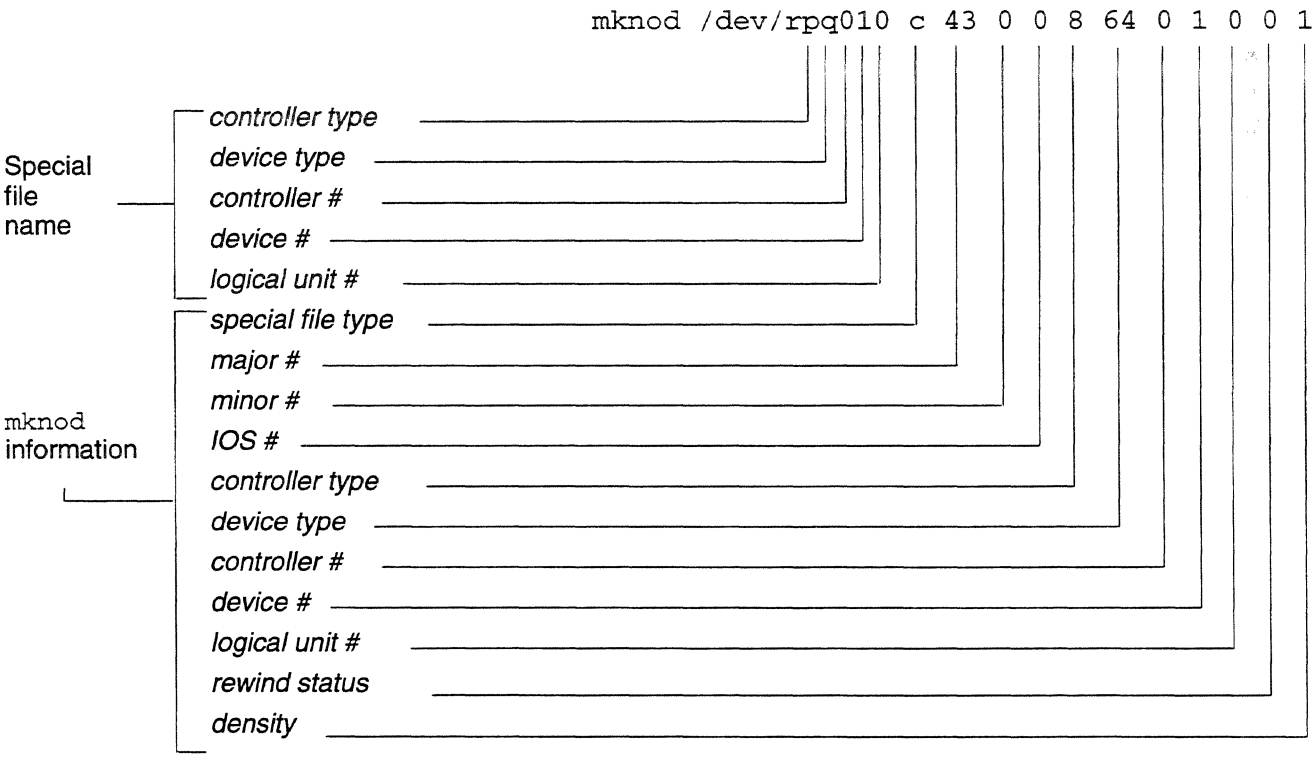


Figure 2. Special file name and `mknod` components

Tape special file naming convention 15.3.1.1

The special file name can be any name that is unique in the `/dev` directory. This subsection defines the naming conventions typically used:

Variable	Description
controller type	Governs the functions of the attached devices. Possible values are as follows:

Type	Description
`p`	I/O processor
`s`	Small Computer Systems Interface (SCSI) controller (SI1, SI2, or SI3)
`m`	PERTEC controller

device type — Defines device being used. Possible values are as follows:

Type	Description
`q`	Quarter-inch cartridge tape device
`d`	HP DDS-2 digital audio tape (DAT) device
`e`	EXABYTE tape device
`m`	STK 9914 9-track tape (SCSI)
`s`	STK square tape
`t`	9-track tape (PERTEC)

controller # — Defines controller being used. Possible values are as follows:

Type	Description
`0`	I/O processor
`0-7`	SCSI and PERTEC

device # — Defines device ID number. Possible values are as follows:

Type	Description
`0-6`	SCSI
`1-6`	I/O processor
`0-3`	PERTEC

Variable	Description
logical unit #	Defines the SCSI logical unit being used; optional value. This field does not apply to PERTEC tapes. Possible values are as follows:

Type	Description
0-7	SCSI

Tape special file mknod fields
15.3.1.2

The variables of the special file mknod fields are as follows:

Variable	Description
special file type	Specifies the special file type entry.

Type	Description
c	Creates character special file.

Variable	Description
major #	Specifies major device number.

Type	Description
43	Major number is always 43.

Variable	Description
minor #	Specifies minor device. Possible values are as follows:

Type	Description
0-31	This number can be any unique number between 0 and 31.

Variable	Description
IOS #	Specifies the IOS on which the drive resides. Possible values are 0 through 15.
controller type	Specifies the controller on which the drive resides. Possible values are as follows:

Type	Description
8	I/O processor
9	PERTEC controller
10	SCSI controller (SI1, SI2, or SI3)

Variable	Description
device type	Specifies device being used. Possible values are as follows: Type — Description 6 — STK square tape 4220 or STK 18-track square tape (SCSI). 64 — Quarter-inch cartridge tape device (SCSI) 65 — EXABYTE tape device (SCSI) 66 — 9-track round tape (PERTEC) 68 — HP DDS-2 digital audio tape (DAT) device (SCSI) and generic SCSI (i.e., `rsm`*xxx*); VTAPE interface: STK 9914 (SCSI) 1/2" round tape.
controller #	Defines controller being used. Possible values are as follows: Type — Description 0 — I/O processor 0–7 — SCSI (SI1, SI2, or SI3) and PERTEC
device #	Device ID number. Possible values are as follows: Type — Description 0–3 — PERTEC 0–6 — SCSI 1–6 — I/O processor
logical unit #	Specifies the device logical unit number for SCSI devices. Possible values are 0 through 7.
rewind status	Specifies whether the device is a rewind or no-rewind device. Possible values are as follows: Type — Description 0 — Rewind device 1 — No-rewind device

Variable	Description
density	Specifies whether the device (9-track only) will write tapes in high density or low density. Possible values are as follows:

Type	Description
0	High density (6250 b/i)
1	Low density (1600 b/i)

Sample character special files 15.3.2

Table 3 shows example command lines that create special character files for various devices and configurations.

Table 3. Example special character file command lines

Device	Configuration	Command line
QIC	—	`mknod /dev/rpq01 c 43 0 0 8 64 0 1 0 0 0` `mknod /dev/nrpq01 c 43 1 0 8 64 0 1 0 1 0`
DAT	—	`mknod /dev/rpd03 c 43 5 0 8 68 0 3 0 0 0` `mknod /dev/nrpd03 c 43 15 0 8 68 0 3 0 1 0`
EXABYTE	Attached to the IOP SCSI bus	`mknod /dev/rpe02 c 43 2 0 8 65 0 2 0 0 0` `mknod /dev/nrpe02 c 43 3 0 8 65 0 2 0 1 0`
EXABYTE	Attached to the VME SCSI controller	`mknod /dev/rse02 c 43 4 0 10 65 0 2 0 0 0` `mknod /dev/nrse02 c 43 5 0 10 65 0 2 0 1 0`

Two SCSI square tapes	One SCSI ID number and attached to the VME SCSI controller	`mknod /dev/rss000 c 43 6 0 10 6 0 0 0 0 0`
		`mknod /dev/rss001 c 43 7 0 10 6 0 0 1 0 0`
		`mknod /dev/nrss001 c 43 8 0 10 6 0 0 1 1 0`
		`mknod /dev/nrss000 c 43 9 0 10 6 0 0 0 1 0`

Two SCSI square tapes	Different SCSI ID numbers and attached to the VME SCSI controller	`mknod /dev/rss000 c 43 10 0 10 6 0 0 0 0 0`
		`mknod /dev/rss010 c 43 11 0 10 6 0 1 0 0 0`
		`mknod /dev/nrss000 c 43 12 0 10 6 0 0 0 1 0`
		`mknod /dev/nrss010 c 43 13 0 10 6 0 1 0 1 0`

UNICOS `tpdaemon` tape subsystem overview

15.4

This subsection describes configuring tape devices that the UNICOS `tpdaemon` (the UNICOS tape daemon) can access.

Tape configuration is run-time configurable. Changes to the configurations of devices (controllers, channels, drives, and so on) or tape parameters (AVR, tape groups, and so on) are made in the `/etc/config/tapeconfig` file. To make changes, either edit the file manually or use the menu system.

Note

To incorporate a new configuration, you must stop and restart the tape daemon.

Accessing tapes through `tpdaemon` offers features such as multivolume support, labeled tapes, tape silo access, and data buffering. The tape daemon also offers users a lot of tape device control, flexibility, and security.

Use of the UNICOS tape daemon requires you to specify a tape daemon configuration file. The tape daemon creates and maintains special files in the UNICOS `/dev/tape` directory. All user tape device requests can be made through the tape daemon, which actually manages the devices. The UNICOS `tpdaemon` is the way most users probably will want to access and manipulate tape devices.

Note

For information about using the character special tape file interface that uses the `tpdaemon` command, see the UNICOS 8.0.3 online version of the *UNICOS Tape Subsystem User's Guide*, publication SG–2051. You can access this publication through the CrayDoc and Docview products.

Tape daemon commands 15.4.1

This subsection provides a list of tape daemon commands. For detailed information about these commands, see the man pages.

Following is a list of the `sys` commands:

Command	Description
`rls`(1)	Releases reserved tape resources.
`rsv`(1)	Reserves tape resources.
`tpcatalog`(1)	Catalogs, recatalogs, or deletes a dataset in a front-end catalog.
`tplist`(1)	Lists contents of tape volume.
`tpmnt`(1)	Requests a tape mount for a tape file.
`tprst`(1)	Displays reserved tape status for current job ID.
`tpscr`(8)	Returns volumes to loader scratch pool.
`tpstat`(1)	Displays current tape status.

Following is a list of the `operator` commands:

Command	Description
`tpapm`(8)	Requests that tape daemon mount a volume on any available drive serviced by an autoloader.
`tpbmx`(8)	Displays operator information about tape devices.
`tpclr`(8)	Clears the tape drive.
`tpconfig`(8)	Configures tape devices up and down.
`tpdaemon`(8)	Starts tape daemon.
`tpdev`(8)	Displays current tape equipment configuration.
`tpdstop`(8)	Stops tape daemon.
`tpfrls`(8)	Forcibly releases tape reservation and associated devices.

Command	Description
`tpgstat`(8)	Displays user reservation status for all users.
`tplabel`(8)	Labels magnetic tape reel or cartridge.
`tpmls`(8)	Displays loader status.
`tpmql`(8)	Displays tape daemon mount request queue list.
`tpset`(8)	Sets features for the tape daemon, `tpdaemon`.

Tape and message daemon directories 15.4.2

The following spooling areas (directories) are used for the tape subsystem:

Directory	Description
`/usr/spool/msg`	Spooling area for message daemon. Contains log files, lock file for the message daemon, and the named pipe for the message daemon requests.
`/usr/spool/tape`	Spooling area for tape daemon. Contains trace files and the named pipe for tape daemon requests.

Procedure: Setting up the tape daemon

To set up the tape daemon for the first time, perform the following steps:

1. Ensure that all tape daemon programs and commands have the superuser ID, that is, that they are all owned by `root` (see subsection 15.4.1, page 369).

2. Configure user and operator logins so that they have appropriate tape usage privileges. By default, the `operator` login ID that comes with your system has 0 (no) tape privileges.

   In the user database (UDB), the default setting for all parameters is 0, which means a value of infinite for that parameter, except for tape privileges; a setting of 0 means that users cannot use tapes at all. A setting of `256` for tapes is equivalent to infinite tape privileges.

   The tape types for which privileges can be specified (for example, `type0`, `type1`, and so on, if using `etc/nu` to modify or create a UDB entry) will match the type and the order of the tape types listed in the `DEVICE_GROUPS` section of the `/etc/config/tapeconfig` tape configuration file. Whatever is listed first in the `DEVICE_GROUPS` section, such as `QIC`, will be the type referred to as `type 0` in the UDB.

   Give all users and operators who must access tapes appropriate UDB settings for your site. For example, if you have listed `QIC`, `EXB`, `TAPE`, and `CART` in your `DEVICE GROUPS` section of `/etc/config/tapeconfig` and you want operators to have access to all four device groups, but you want other users to access only the production tape types of `TAPE` and `CART`, operator logins must have nonzero values as settings for tape types 0, 1, 2, and 3, but users must have zero values for the first two types (`type0` and `type1`) and nonzero values for the last two types.

3. Configure user and operator logins to belong to the appropriate group(s) to use `tpdaemon` tape commands. A login must be a member of the appropriate group to use any command that belongs to that group. For more information, see the `/etc/group` file discussed in section 7, page 149.

   Any login that is a member of the `sys` group can use all tape-related commands in the `/bin` directory. Most of the rest of the tape-related commands are in the `/etc` directory. By default, the `operator` login that comes with your system already belongs to the `operator` group, but not to the `sys` group. Any login that is a member of the `operator` group can use the commands that are in the `/etc` directory. The tape daemon itself, `tpdaemon`, is in the `/usr/lib/tp` directory, and it is also in the `operator` group.

   To ensure that a login is a member of the same group as a command, use **one** of the following procedures:

   - Change the commands to match the users. Change the group ID of the command to match one of the groups to which the login belongs by using the `/bin/chgrp`(1) command (for more information, see section 7, page 149, the `chgrp`(1) man page, and *UNICOS System Administration,* publication SG–2113).

For example, you may want both your users and your operators to be able to use the `/bin/tpmnt` command (to mount tapes), but you may want only your operators to be able to use the `/etc/tpconfig` command (to configure tape devices, loaders, and so on, up and down). The `/etc/tpconfig` command is already in the `operator` group, and the `operator` login is already a member of that group. If users do not belong to the `operator` group, they cannot use the `/etc/tpconfig` command.

or

- Change the users to match the commands. Change the UDB settings for the user login that must use these commands so that the user belongs to the group shown for a command's group ID.

4. By default, `/etc/rc` includes calls to the `/etc/sdaemon` script. `/etc/sdaemon` reads the `/etc/config/daemons` file and start certain system daemons according to the information in this text file. You should ensure that your `/etc/config/daemons` file `start` parameter for the tape daemon (`tpdaemon`) and the message daemon (`msgdaemon`) is `YES` so that each is started automatically when the system is brought up. You can do this step either manually or by using the menu system.

   If you are using the menu system, select the `Configure System ==> System Daemons Configuration ==> System Daemons Table` menu, set the `tpdaemon` and the `msgdaemon` daemons to `YES`, and update the form file. Then activate your changes through the `System Daemons Configuration` menu. When you activate this change, the daemons will be started automatically each time you start UNICOS. A sample `System Daemons Table` menu screen follows:

```
Configure System
...System Daemons Configuration
......System Daemons Table
```

```
                              System Daemons Table
SYS1 msgdaemon   YES  /etc/msgdstop /usr/lib/msg/msdaemon
SYS2 tpdaemon    YES  /etc/tpdstop  /usr/lib/tp/tpdaemon    -rc
```

   If you are not using the menu system, edit the `/etc/config/daemons` file to enable the `tpdaemon` and the `msgdaemon` daemons, as shown in the following example. If you do not use the menu system, editing this file also will ensure that the daemons will be started automatically each time you start UNICOS.

```
# vi /etc/config/daemons
SYS1 msgdaemon   YES  /etc/msgdstop /usr/lib/msg/msdaemon
SYS2 tpdaemon    YES  /etc/tpdstop  /usr/lib/tp/tpdaemon    -rc
```

5. Define the physical devices and software operating parameters with which the tape daemon will operate. For more information about defining devices, see subsection 15.5, page 375. You can do this step either manually or by using the menu system.

   If you are using the menu system, select the `Configure System ==> Tape Configuration` menu, import your default tape configuration, modify the necessary submenus, and update the related form files to configure your physical devices. A sample `Tape Configuration` menu screen follows:

```
Configure System
...Tape Configuration
```

```
                        Tape Configuration

     Configure tape loader(s) ==>
M-> Configure tape resource group(s) ==>
     Configure tape IOP(s)  ==>
        Configure channels ==>
        Configure banks ==>
     Select tape subsystem options ==>
     Import the tape configuration ...
     Activate the tape configuration ...
```

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

6. Any changes you made to your tape configuration will not take effect until the tape daemon is initialized with the `tpdaemon` command. If the tape daemon is already running, you can stop it with the `/etc/tpdstop` command. You can start the tape daemon with your configuration by either executing the `/usr/lib/tp/tpdaemon` command manually or letting it start automatically at system boot time (using `/etc/rc`).

Note

If you start the tape daemon manually, make sure that the `msgdaemon` is already running.

---

7. If a component was configured as `status = DOWN` in the `/etc/config/tapeconfig` file, that component will not be available to the tape daemon until the component's status is changed to `UP`. After the tape daemon is initialized, use can change a component's status by using the `/etc/tpconfig` command. Sites may want to use this command to change the availability of certain components (tape drives, loaders, and so on), depending on time of the day, peripheral maintenance, user requests, and so on. To configure tape hardware available for use, see the procedure for using the `tpconfig` command on page 403.

Tape configuration parameters
15.5

The tape configuration file defines all of the tape hardware that the system uses. The diagnostic devices are implicitly defined when the I/O processors (IOPs) and the channels are defined. You must not redefine them.

Note

When UNICOS publications provide information for Cray Research systems that have an I/O subsystem model E (IOS-E), this information also applies to CRAY J90 and CRAY EL systems, unless noted otherwise.

The tape configuration file consists of comments (optional) and statements. A comment begins with the number sign (#) and continues to the end of line. A statement consists of a name followed by a list of keyword parameters. There are four statements, two of these statements also consist of other statements. Statements must be in the order shown:

1. Loader statements (one per loader)
2. Device group statements (one per device group)
3. IOP statements (one per IOP); the IOP statement consists of the following two statements that define the IOP configuration:
   a. Channel statements (one per channel in the IOP)
   b. Bank statements (one per bank); the bank statement consists of two of the following three statements that define the bank configuration:
      - Slave statements (one per slave device) or Control Unit statements (one per control unit)
      - Device statements (one per device)
4. Options statement

The following is the list of statement syntax rules.

- The statement name and its parameters are separated by one or more white spaces (blank, tab, or new line characters).
- Adjacent parameters are separated by a comma.

- The end of the parameter list is indicated by the absence of a comma.
- Adjacent statements are separated by one or more white spaces.

The following is a list of keyword parameter syntax rules:

- The keyword is separated from its value by the equal sign (=).
- The value of a keyword may consist of keywords, numbers, character strings, and lists of keywords, numbers, and character strings.
- If the value of a keyword is a list, the list is enclosed within left and right parentheses. Adjacent elements of a list are separated by a comma. If the list consists of one element, you do not have to enclose it in parentheses. The elements of a list may be lists.
- Numbers may be specified in decimal, octal, and hexadecimal formats. These formats are the same as those used in the C programming language:

Decimal	The first digit is not 0 (for example, 1372).
Octal	The first digit is 0 (for example, 0563).
Hexadecimal	The first 2 characters are either 0x or 0X (for example, 0xf2).

- Character strings are series of characters. If any one of the special characters (white space, ", #, =, {, }, (,), ', \ ) is needed in the string, you must enclose the string in a pair of double quotation marks ("). Within a pair of double quotation marks, the sequence of characters \x (x is any character) will be replaced by x. This is the only way you can specify a " and a \ in a quoted string.
- Comments may appear between any symbols as described.

You may code the names of statements and keywords in a mixture of uppercase and lowercase letters. The values that the user specifies are case sensitive. The following specify the same thing:

```
Name = A
name = A
```

The following are different:

```
name = A
name = a
```

The following subsections describe the parameters for these statements. A sample `tapeconfig` configuration file follows.

```
LOADER
      name = Operator ,
      type = OPERATOR ,
      status = UP ,
      message_path_to_loader = MSGDAEMON ,
      message_class = NONE ,
      server = UNICOS ,
      scratch_volume_label_type = (NL,AL,SL) ,
      queue_time = 0 ,
      verify_non_label_vsn = YES ,
      message_route_masks = (UNICOS) ,
      mode = ATTENDED
LOADER
      name = wolfy ,
      type = STKACS ,
      status = UP ,
      message_path_to_loader = NETWORK ,
      message_class = TYPE_340 ,
      server = stkwolf ,
      scratch_volume_label_type = NONE ,
      queue_time = 0 ,
      verify_non_label_vsn = NO ,
      message_route_masks = (UNICOS) ,
      mode = ATTENDED
DEVICE_GROUP
      name = QIC ,
      avr = YES
```

```
DEVICE_GROUP
      name = EXB ,
      avr = YES
DEVICE_GROUP
      name = DAT ,
      avr = YES
DEVICE_GROUP
      name = CART ,
      avr = YES
DEVICE_GROUP
      name = TAPE ,
      avr = YES
DEVICE_GROUP
      name = WOLF ,
      avr = YES
IOP
      number = 0 ,
      cluster = 0
      {
            CHANNEL
                  address = 022 ,
                  status = UP
            BANK
                  number = 0
                  {
                        CONTROL_UNIT
                              protocol = IOP ,
                              status = UP ,
                              path = ((022, 0))
                        DEVICE
                              name = rpe02 ,
                              device_group_name = EXB ,
                              id = 020 ,
                              type = EXB ,
                              status = DOWN ,
                              loader = Operator
                        DEVICE
                              name = rpd03 ,
                              device_group_name = DAT ,
                              id = 030 ,
```

```
                              type = DAT ,
                              status = DOWN ,
                              loader = Operator
                        }
      }
IOP
      number = 0 ,
      cluster = 4
      {
            CHANNEL
                  address = 023 ,
                  status = UP
            BANK
                  number = 1
                  {
                        CONTROL_UNIT
                              protocol = SI2 ,
                              status = UP ,
                              path = ((023, 0))
                        DEVICE
                              name = rsm010 ,
                              device_group_name = TAPE ,
                              id = 00 ,
                              type = VTAPE ,
                              status = DOWN ,
                              loader = Operator
                  }
      }
IOP
      number = 0 ,
      cluster = 2
      {
            CHANNEL
                  address = 024 ,
                  status = UP
            CHANNEL
                  address = 025 ,
                  status = UP
            BANK
                  number = 2
```

```
        {
              CONTROL_UNIT
                    protocol = SI2 ,
                    status = UP ,
                    path = ((024, 0))
              DEVICE
                    name = rss000 ,
                    device_group_name = CART ,
                    id = 040 ,
                    type = 3480 ,
                    status = DOWN ,
                    loader = Operator
        }
BANK
        number = 3
        {
              CONTROL_UNIT
                    protocol = SI2 ,
                    status = UP ,
                    path = ((025, 1))
              DEVICE
                    name = B00 ,
                    device_group_name = WOLF ,
                    id = 00 ,
                    type = 3480 ,
                    status = DOWN ,
                    vendor_address = (0,0,9,0),
                    loader = wolfy
              DEVICE
                    name = B01 ,
                    device_group_name = WOLF ,
                    id = 01 ,
                    type = 3480 ,
                    status = DOWN ,
                    vendor_address = (0,0,9,1),
                    loader = wolfy
                    loader = wolfy
              DEVICE
                    name = B02 ,
                    device_group_name = WOLF ,
```

```
                              id = 02 ,
                              type = 3480 ,
                              status = DOWN ,
                              vendor_address = (0,0,9,2),
                              loader = wolfy
                        DEVICE
                              name = B03 ,
                              device_group_name = WOLF ,
                              id = 03 ,
                              type = 3480 ,
                              status = DOWN ,
                              vendor_address = (0,0,9,3),
                              loader = wolfy
                  }
      }
OPTIONS
      avr_at_startup = YES ,
      max_number_of_device_groups = 8 ,
      max_blocksize = 4194303 ,
      max_number_of_tape_users = 100 ,
      tape_daemon_trace_file_prefix = /usr/spool/tape/trace ,
      tape_daemon_trace_file_owner = 0 ,
      tape_daemon_trace_file_group_id = 9 ,
      tape_daemon_trace_file_mode = 0640 ,
      tape_daemon_trace_file_size = 409600 ,
      tape_daemon_trace_savefile_prefix = /usr/spool/tape/save/trace ,
      blp_ring_status = UNRESTRICTED ,
      system_code = CRI/UNICOS ,
      reselect_cart = YES ,
      device_group_name = WOLF ,
      retention_period_days = 0 ,
      ring_status = (IN,OUT) ,
      blocksize = 32768 ,
      file_status = OLD ,
      label_type = AL ,
      scratch_volume_vsn = ?????? ,
      scratch_volume_retries = 0 ,
      number_of_autoloader_retries = 10 ,
      ask_blp = NO ,
      ask_label_switch = YES ,
```

```
ask_vsn = NO ,
verify_scratch_vsn = NO ,
operator_message_destination = (UNICOS) ,
mainframe_job_origin = C1 ,
scratch_volume_action = FREE ,
operator_message_type = USCP_TYPE_1 ,
loader_device_assignment_order = DEVICE_LIST ,
check_vsn = NO ,
check_file_id = NO ,
check_protection = NO ,
check_expiration_date = NO ,
message_daemon_pipename = /usr/spool/msg/msg.requests ,
cray_reel_librarian = NO ,
cray_reel_librarian_mandatory = NO ,
cray_reel_librarian_scratch_vsn = ?CRL?? ,
cray_reel_librarian_operator_select_scratch = NO ,
tape_daemon_dump_mask = (MLT,LBL,DPW,SM,LDRREQ,CSI,ACC,EDT,FUNC)
tcp_daemon_frontend_id
tcp_daemon_socket_port_number
```

The following subsections explain the statement parameters you must define; for explanations of additional fields you also may want to define, see the `tpconfig`(8) man page.

LOADER *statement* 15.5.1

The `LOADER` statement identifies the loaders in the tape configuration file. `LOADER` statements **must** appear before the device information. The `LOADER` definitions **must** define at least a manual loader.

Note

For information about the following topics that pertain to the autoloader, see the UNICOS Tape Subsystem section in *UNICOS System Administration,* publication SG–2113: Organizing devices in attended or unattended mode, naming and numbering device groups, and accessing tape cartridges.

The LOADER statement has the following format:

```
LOADER keyword-parameter-list
```

Table 4 describes the tape configuration file parameters for the LOADER statement.

Table 4. LOADER statement parameters

Parameter	Value	Description
name	Character string	Specifies the name of the loader. No default.
type	OPERATOR	The operator loads the drive.
	STKACS	STK 4400 or WOLFCREEK autoloader is used.
	EMASS	EMASS autoloader is used. No default.
status	UP	Loader is up when the tape daemon starts.
	DOWN	Loader is down when the tape daemon starts.
		No default.
message_path_to_loader	MSGDAEMON	Use message daemon to send message to loader.
	NETWORK	Use TCP/IP network protocol to send message to loader.
		No default.
message_class	NONE	No message class.
	TYPE_340	USCP message class; USCP is not supported on CRAY J90 or CRAY EL systems.
		No default.

Table 4. LOADER statement parameters (continued)

Parameter	Value	Description
server	Character string	Specifies the server name.
	Server list	List of servers.
		No default.
scratch_volume label_type	List of AL, NL, or SL	Scratch label to be used.
	NONE	No scratch labels can be used.
		No default.
queue_time	0	Wait up to 24 hours for the best drive to mount a tape.
	Nonzero	Number of seconds to wait for the best drive to mount a tape before asking the operator to import or export the requested tape.
		No default.
verify_non_ label_vsn	YES	Verify nonlabel VSN.
	NO	Do not verify nonlabel VSN.
		No default.
message_ route_masks	UNICOS	Route mount messages to UNICOS console
	SERVER	Route mount messages to server.
		No default.
mode	ATTENDED	Request operator intervention.
	UNATTENDED	Assume negative response for operator intervention.
		No default.
return_host	Character string	Specifies the name of the Cray Research host that serves as the return address for the server.
		Optional.

DEVICE_GROUP *statement*
15.5.2

The tape daemon enforces a resource limit for each user based on the entry for that user in the user database (UDB). The UDB limit applies to resource group numbers rather than resource or device names. This necessitates a way of mapping the devices configured in the system to the appropriate resource numbers.

Like other Cray Research systems, CRAY J90 and CRAY EL systems support the following:

- 9-track, or round, tapes (TAPE) (CRAY J90 systems support only SCSI version)
- 18-track, or square, tapes (CART)

CRAY J90 and CRAY EL systems also support additional tape types, including the following:

- Digital Audio Tapes (DAT)
- EXABYTE 8-mm cassette tapes (EXB) (CRAY EL systems only)
- Quarter-inch cartridge (QIC) quick or streaming tapes (CRAY EL systems only)

CRAY J90 and CRAY EL systems use a standard naming convention for tape device names; for the naming convention, see the special file name portion of Figure 2, page 363.

The order of DEVICE_GROUP statements determines the default set of resource names (device groups). The first device group encountered represents resource group number 0. The second device group represents resource group number 1, and so on.

Changing the order by using the DEVICE_GROUP statements gives you the flexibility to change the configuration while maintaining a consistent naming convention for device groups that are mapped to the limits set for each user in the UDB.

The DEVICE_GROUP statement has the following format:

```
DEVICE_GROUP keyword–parameter–list
```

Table 5 describes the configuration file parameters for the DEVICE_GROUP statement.

Table 5. DEVICE_GROUP statement parameters

Parameter	Value	Description
name	Character string	Specifies the device group name. No default.
avr	ON	Starts automatic volume recognition (AVR) for this group.
	OFF	Does not start AVR for this group. Default is the avr_at_startup option in the options statement.

IOP *statement*
15.5.3

The IOP statement specifies IOP characteristics. You should include one IOP statement for each IOP that contains tapes. This statement has the following format:

```
IOP keyword–parameter–list { iop–configuration }
```

The *iop–configuration* consists of a series of channel statements and bank statements, which are described in Table 7 and Table 8, respectively.

Table 6 describes the tape configuration file parameters for the IOP statement.

Table 6. IOP statement parameters

Parameter	Value	Description
number	0	Specifies the IOP number; must be set to 0. No default.
cluster	*Number*	Specifies the IOS number on CRAY J90 and CRAY EL systems. No default.
{ *iop–configuration* }		Specifies a series of CHANNEL statements and BANK statements. The braces are part of the syntax and must be coded.

CHANNEL *statement* 15.5.3.1

A CHANNEL statement is one of two statements that constitute the *iop–configuration* component of the IOP statement. The CHANNEL statement specifies channel characteristics for an IOP. You should include one CHANNEL statement for each tape controller (or SI2 controller port) on this IOP.

This statement has the following format:

```
CHANNEL keyword–parameter–list
```

Table 7 describes the tape configuration file parameters for the IOP CHANNEL statement parameters.

Table 7. CHANNEL statement parameters

Parameter	Value	Description
address	20 - 37 octal	Set the address parameter for each controller to a unique number between 20 and 37 (octal) inclusive. This number can be arbitrary, as long as it is within this range and is unique for each controller in the system.
status	UP	The channel status should always be set to UP. No default.

BANK *statement*
15.5.3.2

A BANK statement is the second of two statements that constitute the *iop–configuration* component of the IOP statement.

The BANK statement specifies the bank characteristics of an IOP. The BANK statement contains definitions of devices and the controller to which they are associated.

This statement has the following format:

```
BANK keyword–parameter { bank–configuration }
```

Table 8 describes the tape configuration file parameters for the IOP BANK statement.

Table 8. BANK statement parameters

Parameter	Value	Description
number	0	Must always be set to 0 on CRAY J90 and CRAY EL systems. By default, a bank number will be assigned to a bank.
{ *bank-configuration* }		Specifies a series of slave statements or control unit statements followed by a series of device statements. The braces are part of the syntax and must be coded.

SLAVE statement
15.5.3.2.1

On CRAY J90 and CRAY EL systems, the SLAVE statement is used only for ER90 devices. The SLAVE statement is one of three statements that constitute the bank-configuration component of the BANK statement. This statement has the following format:

```
SLAVE keyword–parameter–list
```

Table 9 describes the tape configuration file parameters for the IOP SLAVE statement.

Table 9. SLAVE statement parameters

Parameter	Value	Description
status	UP	The slave is up when the tape daemon is started.
	DOWN	The slave is down when the tape daemon is started.
		No default.

Table 9. SLAVE statement parameters (continued)

Parameter	Value	Description
`path`	(*channel-number, slave-address*)	Specifies a list of IPI channel address and slave address pairs enclosed in parentheses. The parentheses are part of the syntax and must be coded. The channel number is the channel that is connected to the port address in the slave. (ER90/IPI only) No default.
`reset_timeout`	*Number*	Specifies the reset timeout in seconds. (ER90/IPI only) No default.
`channel_pairs`†	(*input-channel: output-channel,* [*input-channel: output-channel*])	Specifies a list of ER90/HIPPI channel pairs that may be used to issue requests and send data to the slave. The parentheses are part of the syntax and must be coded. If two channel pairs are specified, the channel pairs must be unique. The channel pairs must have been defined in a channel statement. (ER90/IPI only) No default.
`i_field`†	*Address*	Specifies the ER90/HIPPI `i_field` address used to route requests from a HIPPI switch to a slave. (ER90/IPI only) No default.

† Deferred implementation

CONTROL_UNIT statement 15.5.3.2.2

The CONTROL_UNIT statement is one of three statements that constitute the bank-configuration component of the BANK statement. This statement has the following format:

```
CONTROL_UNIT keyword–parameter–list
```

Table 10 describes the tape configuration file parameters for the IOP CONTROL_UNIT statement.

Table 10. CONTROL_UNIT statement parameters

Parameter	Value	Description
status	UP	The control unit is up when the tape daemon is started.
	DOWN	The control unit is down when the tape daemon is started. No default.
protocol	IOP	IOP = Controller for the devices attached to the IOP onboard SCSI bus.
	356X SI1 SI2 SI3	356X, SI1, SI2, or SI3 = Controller for SCSI tape devices attached to the VMEbus, using the SI1, SI2, or SI3 device. 356X and SI1 are not supported on CRAY J90 systems.
	TM3000	TM3000 = Controller for the PERTEC 9-track round tape devices; supported only on CRAY EL systems.
path	((x,y))	x is the address specified in the CHANNEL statement. y is the Control Unit number in hexadecimal; this value must match the number physically set on the controller hardware. To get this value, either look at the address setting on the controller hardware itself or retrieve it from the output of the IOS load, reload, or whatmic command. The number is specified as the ctlr value in the output from these commands. If the ctlr value is IOP in this output, you should set y to 0. Following is an example excerpt taken from output of the whatmic command: `IOS 2 H, ctlr 0 (VME SCSI SI-3) - detected` No default.

DEVICE statement
15.5.3.2.3

The DEVICE statement is one of three statements that constitute the bank-configuration component of the BANK statement.

This statement has the following format:

```
DEVICE keyword–parameter–list
```

Table 11 describes the tape configuration file parameters for the IOP DEVICE statement.

Table 11. DEVICE statement parameters

Parameter	Value	Description
name	Character string	Specifies the name of the device. No default.
device_group_name	Character string	Specifies the name of a device group defined by a DEVICE_GROUP statement. No default.
status	UP	Specifies initial status as up.
	DOWN	Specifies initial status as down.
		No default. Never place devices in any state but DOWN in the tapeconfig file.
id		Field of 3 octal digits, *xyz*, defined as follows: *x* = Always 0 *y* = SCSI identifier *z* = Logical unit number (LUN) The SCSI identifier, *y*, is set to 0 for non-SCSI devices, such as devices on a TM3000 type of controller. For SCSI devices, you may determine this identifier directly from the hardware or from the output of the IOS load, reload, or whatmic commands. The output from these commands refers to this SCSI identifier as the unit, as shown in the output of the IOS reload command. You may also determine the Logical Unit Number, z, from either the hardware or the lun value given in the output from the load, reload, or whatmic command.

Table 11. DEVICE statement parameters (continued)

Parameter	Value	Description
type	DAT	Specifies Digital Audio Tapes.
	ER90	Specifies ER90 (EMASS) device type; not supported on CRAY J90 systems.
	EXB	Specifies EXABYTE 8-mm cassette tapes; not supported on CRAY J90 systems.
	IDRC	Specifies IDRC device type.
	QIC	Specifies quarter-inch cartridge tapes; supported on CRAY EL systems only.
	VTAPE	Specifies 9914 SCSI device.
	3480	Specifies 3480 device type
	3490	Specifies 3490E device type
	3490E	Specifies PERTEC 9-track round reel tapes; not supported on J90 systems.
	9914	No default.
loader	Character string	Specifies the loader name defined in a loader statement.
		No default.
vendor_address	Character string	Specifies the vendor address of the drive in an autoloader.
		The format for an STK drive is as follows: *acs#,lsm#,panel#,drive#.*
		The format for an EMASS drive is as follows: *drive#.*
		No default.
facility_address	number	Specifies the ER90 facility address.
		No default.
short_timeout	number	Specifies the ER90 short timeout (in seconds).
		No default.

Table 11. DEVICE statement parameters (continued)

Parameter	Value	Description
long_timeout	number	Specifies the ER90 long timeout (in seconds). No default.

OPTIONS *statement* 15.5.3.3

The options in force when the tape daemon is built are specified in the /usr/include/tapereq.h file. You can specify most of these options in the options section of the tape configuration file. List options following the OPTIONS statement. This statement has the following format:

```
OPTIONS keyword–parameter–list
```

You can specify the following options and their corresponding values to override the values with which the tape daemon was built. The options that you can specify in the tape configuration file after the OPTIONS statement are similar to the options in the tapereq.h file, but not identical. Values are often given in a different form in the two files (for example, the value of the ask_blp keyword is expressed as 0 or 1 in tapereq.h, but it is expressed as YES or NO in the tape configuration file.

Table 12 describes the tape configuration file parameters for the OPTIONS statement.

Table 12. OPTIONS statement parameters

Parameter	Value	Description
ask_blp	YES NO	Seeks permission from the operator to use bypass label processing.
ask_label_switch	YES NO	Asks operator for permission to switch label type. Does not ask operator for permission to switch label type. Default: YES

Table 12. OPTIONS statement parameters (continued)

Parameter	Value	Description
ask_vsn	YES	Asks operator for the volume serial number (VSN) when a nonlabel tape is mounted.
	NO	Does not ask operator for the VSN when a nonlabel tape is mounted.
		Default: YES
avr_at_startup	YES	Starts automatic volume recognition (AVR) when tape daemon is started.
	NO	Does not start AVR when tape daemon is started.
		Default: YES
blocksize	*Number*	Specifies the block size to use when no block size is specified by the user in the tpmnt(1) command (-b option). Default: 32768
max_blocksize	*Number*	Specifies the maximum block size (in bytes) that the tape system can read or write. Applies only to the IOS model D.
		Default: Default: 4194303
blp_ring_status	UNRESTRICTED	The user can use the in and out ring status option on the tpmnt command when the user requests bypass label processing.
	OUT	The user can use only the out ring status option on the tpmnt command when the user requests bypass label processing.
		Default: UNRESTRICTED
check_file_id	YES	Verifies the file ID on a labeled tape when the file is opened.
	NO	Does not verify the file ID on a labeled tape when the file is opened.
		Default: YES

Table 12. OPTIONS statement parameters (continued)

Parameter	Value	Description
check_protection	YES NO	Checks the protection flag on the header label. Does not check the protection flag on the header label. Default: YES
check_vsn	YES NO	Verifies the VSN on a labeled tape. Does not verify the VSN on a labeled tape.
check_expiration_date	YES NO	Checks the expiration date on the header label of a labeled tape; if not expired, asks operator. Does not check the expiration date on the header label of a labeled tape. Default: YES
device_group_name	*Character string*	Specifies the default device group name if it is not specified on the tpmnt -g command. No default.
max_number_of_device_groups	*Number*	Specifies the maximum number of device groups. Default: 8
tape_daemon_trace_dump_mask	ALL (option1, option2,...)	Specifies the various tables that the tape daemon should dump in its trace files. The options are TDT, FIT, MLT, LBL, DPW, BMX, WKA, MSG, REQ, REP, SM, LDERREQ, AVR, CSI, SL, DEM, STAT, ACC, FS, EDT, and FUNC. Default: (MLT, LBL, DPE, SM, LDRREQ, CSI, ACC, EDT, FUNC)
file_status	NEW OLD	Specifies the file status if it is not specified on the tpmnt command. Default: OLD

Table 12. OPTIONS statement parameters (continued)

Parameter	Value	Description
label_type	AL SL NL	Specifies the label type if it is not specified on the tpmnt command. Default: AL
loader_device_ assignment_order	DEVICE_LIST ROUND_ROBIN	Specifies the method with which the loader assigns devices. Default: ROUND_ROBIN
operator_message_ destination	List of UNICOS SERVER FRONTEND	Specifies where operator messages are sent. Default: UNICOS
operator_message_type	USCP_TYPE_1 USCP_TYPE_3	Specifies the operator message type. Default: USCP_TYPE_1
message_daemon_ pipename	*Character string*	Specifies the message daemon pipe name. Default: /usr/spool/msg/msg.requests
mainframe_job_origin	*Character string*	Specifies the mainframe ID of the job if it is not specified. No default.
ring_status	IN OUT (IN,OUT)	Accepts only ring in status if ring option (-r) is not specified on the tpmnt command. Accepts only ring out status if ring option (-r) is not specified on the tpmnt command. Accepts either ring in or ring out status if ring option (-r) is not specified on the tpmnt command. Default: (IN,OUT)
reselect_cart	YES NO	Reselects another device at end-of-volume for cartridge type devices, which includes 3480, 3490, and 3490E devices. Default: NO
retention_period_days	*Number*	Specifies the retention period (in days). Default: 0

Table 12. OPTIONS statement parameters (continued)

Parameter	Value	Description
tape_daemon_trace_ save_file_prefix	*Character string*	Specifies the prefix to the tape daemon save files. Default: /usr/spool/tape/save/trace
scratch_volume_action	FREE KEEP	Specifies the action to perform for scratch tapes when they are released. Default: FREE
scratch_volume_retries	*Number*	Specifies the number of retries to get a scratch volume out of the autoloader scratch pool. Default: 3
scratch_volume_vsn	*Character string*	Specifies the scratch tape VSN. Default: ??????
number_of_autoloader_ retries	*Number*	Specifies the number of times to try to send a request to the autoloader before informing the operator of an error. Default: 10
system_code	*Character string*	Specifies the system code to put on tape labels. Default: CRI/UNICOS
maximum_number_of_ tape_users	*Number*	Specifies the maximum number of tape users. Default: 100
tape_daemon_trace_ file_group_id	*Number*	Specifies the group ID of the tape daemon trace files. Default: 9
tape_daemon_trace_ file_mode	*Number*	Specifies the file mode of the tape daemon trace files. Default: 0640
tape_daemon_trace_ file_owner	*Number*	Specifies the owner ID of the tape daemon trace files. Default: 0

Table 12. OPTIONS statement parameters (continued)

Parameter	Value	Description
tape_daemon_trace_file_prefix	*Character string*	Specifies the tape daemon trace file prefix. Default: /usr/spool/tape/trace
tape_daemon_trace_file_size_bytes	*Number*	Specifies the size (in bytes) of the tape daemon trace files. Default: 409600
tcp_daemon_frontend_id	*Character string*	Specifies the front-end ID of the TCP daemon. No default.
tcp_daemon_socket_port_number	*Number*	Specifies the socket port number of the TCP daemon. Default: 1167
uscp_pipename	*Character string*	Specifies the UNICOS station call processor (USCP) pipe name. No default.
cray_reel_librarian	YES NO	Enables the Cray/REELlibrarian. Default: NO
cray_reel_librarian_mandatory	YES NO	Specifies whether the Cray/REELlibrarian is mandatory. Default: NO
cray_reel_librarian_operator_select_scratch	YES NO	Specifies whether the Cray/REELlibrarian should use the automatic pool allocation to satisfy a scratch request.
cray_reel_librarian_scratch_vsn	YES NO	Specifies the scratch VSN that the Cray/REELlibrarian will use to tell the operator that a scratch volume is needed.
cray_reel_librarian_verify_scratch_vsn	YES NO	Indicates whether the operator should verify the scratch mounts by the Cray/REELlibrarian before continuing; NO = Does not ask all of the time; only when the vsn is needed and cannot be found. YES = When labels are not read, asks operator to enter vsn.

Table 12. OPTIONS statement parameters (continued)

Parameter	Value	Description
user_exit_mask	(list of user exits enabled)	Enables the use of the listed user exits. If no user exits are required, this entry is not needed. See the User exits subsection in *UNICOS System Administration*, publication SG–2113.

Note

You should not alter the following parameters on your CRAY J90 or CRAY EL system. They relate to front-end station functionality that CRAY J90 and CRAY EL systems do not support. These options are set to disable this functionality; by default, therefore, you do not have to alter them.

Table 13. OPTIONS that CRAY J90 and CRAY EL systems do not support

Parameter	Value	Description
servicing_frontend_id	*Character string*	Specifies the servicing front-end ID to use when the –m option is omitted on the tpmnt command. CRAY J90 and CRAY EL systems do not support any front-end servicing. Default: " "
servicing_frontend_ mandatory	YES NO	If YES, the servicing front-end ID specified by the servicing_frontend_id parameter is used regardless of the -m option on the tpmnt command. CRAY J90 and CRAY EL systems do not support any front-end servicing. Default: NO
secure_frontend	YES NO	Specifies the security of the front end. Default: NO

Table 13. OPTIONS that CRAY J90 and CRAY EL systems do not support (continued)

Parameter	Value	Description
servicing_frontend_at_ startup	YES NO	Starts servicing the front end when the tape daemon is started. CRAY J90 and CRAY EL systems do not support any front-end servicing. Default: NO
servicing_frontend_ protocol	USCP	Uses USCP protocol to talk to front ends. CRAY J90 and CRAY EL systems do not support USCP or front-end servicing.
	TCP	Uses TCP protocol to talk to front ends. No default.
operator_message_ frontend_id	*Character string*	Specifies the front-end ID for operator messages. Default: " "
stop_hippi_eiop	YES NO	Stops the HIPPI EIOP driver before setting up the HIPPI EIOP configuration. Default: YES
tcp_daemon_pipename	*Character string*	Specifies the pipe name of the TCP daemon. No default.
tcp_daemon_childname	*Character string*	Specifies the child name of the TCP daemon. No default.

User exit options allow users to add special routines to communicate with the tape daemon without having access to the source. User exits allow a system process to examine and modify a structure associated with a tape file. To implement user exits, you must modify and recompile the `tpuex.c` file. You also must switch the user exit to be on or off within the tape configuration file, which you can do either manually or through using the menu system. There are several user exit options; for additional information, see *UNICOS System Administration*, publication SG–2113.

Procedure: Configuring tape hardware available for use by using the `tpconfig` command

If you configured any tape hardware as not available, you must use this procedure to configure your tape hardware as "up and available" to the user.

Editing the `/etc/config/tapeconfig` file serves two main purposes. When the `tpdaemon` reads that file, the daemon is informed of the tape hardware that is configured on the system and whether that hardware has been configured to be `up` or `on` (that is, available to users). If any pieces of the tape hardware have been configured as `OFF` or `down`, after the `tpdaemon` has been brought up, you must use the `/etc/tpconfig` command to configure that hardware as available for use. The following procedures provide examples of how to use the `tpconfig` command to configure a controller and a tape device to be up and available.

Using the `tpconfig` command to configure a controller available for use

This procedure describes the steps involved in using the `/etc/tpconfig` command to configure a tape controller up and available for use. This step is necessary if, in your `/etc/config/tapeconfig` file, the fourth field for any given controller was defined as `OFF`. In the following sample `/etc/config/tapeconfig` file excerpt, all of the controllers have been configured as `OFF`. This means that when `tpdaemon` is started, the tape software subsystem will know these controllers exist, but the controllers will not be enabled.

```
-CONTROLLERS
tp_cntlr_IOS0 IOP       0       OFF     0       0       020     0
tp_cntlr_TAPE1 SI2      1       OFF     2       0       021     0
tp_cntlr_TAPE2 SI2      2       OFF     3       0       022     0
tp_cntlr_CART1 SI2      3       OFF     3       0       023     0
```

1. Determine the appropriate values for the `/etc/tpconfig` command by using the `/etc/tpdev` command.

   To determine the values that you need for the `/etc/tpconfig` command parameters, you can either look in the `/etc/config/tapeconfig` file or examine the display output of the `/etc/tpdev` command. A sample `tpdev` display follows.

```
sn5111# /etc/tpdev
  dev name  dev grp   bnk  did  ios  iop  dvst  cu   cust  ch   chst  loader
  ========  ========  ===  ===  ===  ===  ====  ===  ====  ===  ====  ========
     rpe02  EXB         0    0    0    0  down    0  down  20   up    Operator
     rsm00  TAPE        1    0    2    0  down    0  down  21   up    Operator
     rsm10  TAPE        2    0    3    0  down    0  down  22   up    Operator
    rss000  CART        3    0    3    0  down    0  down  23   up    Operator
    rss010  CART        3   10    3    0  down    0  down  23   up    Operator
```

2. Configure the controller up by using the `/etc/tpconfig` command.

   `/etc/tpconfig -u` *channel/control_unit* `-i` *ios* `-p` *iop* `up`

   The following is an example, based on the preceding `tpdev` display, of how to configure a square (`CART`) type controller up and available to users:

```
sn5111# /etc/tpconfig -u 23/0 -i 3 -p 0 up
```

3. Confirm that the controller is now configured up and available for use.

   Now, when you examine the `tpdev` display, the "cust" (control unit status) column has changed from `down` to `up` for both of the square (`CART`) tape drives attached to that controller.

```
sn5111# /etc/tpdev
  dev name  dev grp   bnk  did  ios  iop  dvst  cu   cust  ch   chst  loader
  ========  ========  ===  ===  ===  ===  ====  ===  ====  ===  ====  ========
     rpe02  EXB         0    0    0    0  down    0  down  20   up    Operator
     rsm00  TAPE        1    0    2    0  down    0  down  21   up    Operator
     rsm10  TAPE        2    0    3    0  down    0  down  22   up    Operator
    rss000  CART        3    0    3    0  down    0  up    23   up    Operator
    rss010  CART        3   10    3    0  down    0  up    23   up    Operator
```

Using the `tpconfig` command to configure a tape device available for use

This procedure describes the steps involved in using the `/etc/tpconfig` command to configure a tape device up and available for use. This step is necessary if, in your `/etc/config/tapeconfig` file, the value for the sixth field for any given device is defined as an initial state of `down`. In the following excerpt from a sample `/etc/config/tapeconfig` file, all tape devices have been configured `down`. This means that when the `tpdaemon` is started, the tape software subsystem will know these devices exist, but the devices will not be enabled.

```
-DEVICES
rpe02    EXB     0       020     EXB     DOWN     Operator
rsm00    TAPE    1       000     9914    DOWN     Operator
rsm10    TAPE    2       000     9914    DOWN     Operator
rss000   CART    3       000     3480    DOWN     Operator
rss010   CART    3       010     3480    DOWN     Operator
```

1. Determine which devices are down.

   Two of the more useful commands for determining the status of a tape device are the `tpstat` command and the `tpdev` command. Sample output from the `tpstat` command follows. Sample output from the `tpdev` command is shown in step 2.

   The output of the `tpstat` command generates a status display that will show which devices are currently `up` and available for use or `down` and unavailable. A sample follows. Notice that all the devices are in a `down` state.

```
sn5111# /bin/tpstat
userid   jobid dgn      a stat dvn      bx i rl ivsn   evsn   blks  NQSid
              EXB      + down rpe02    03 0
              TAPE     + down rsm00    04 2
              TAPE     + down rsm10    05 3
              CART     + down rss000   06 3
              CART     + down rss010   07 3
```

2. Determine the appropriate values for the `/etc/tpconfig` command by using the `/etc/tpdev` command.

   To determine the values that you will need for the `/etc/tpconfig` command parameters, you can either look in the `/etc/config/tapeconfig` file or examine the display output of the `/etc/tpdev` command. A sample `tpdev` display follows. The controllers ("cust") for the `CART` tape devices have already been configured up.

Note

Before the device itself is configured up, the controller for a device must always be configured as up.

```
sn5111# /etc/tpdev
  dev name   dev grp    bnk  did  ios  iop  dvst  cu   cust  ch   chst  loader
  ========   ========   ===  ===  ===  ===  ====  ===  ====  ===  ====  ========
     rpe02   EXB          0    0    0    0  down    0  down  20   up    Operator
     rsm00   TAPE         1    0    2    0  down    0  down  21   up    Operator
     rsm10   TAPE         2    0    3    0  down    0  down  22   up    Operator
    rss000   CART         3    0    3    0  down    0  up    23   up    Operator
    rss010   CART         3   10    3    0  down    0  up    23   up    Operator
```

3. Configure the tape device up by using the `/etc/tpconfig` command, as follows:

   `/etc/tpconfig` *device_name up*

   The following is an example, based on the preceding `tpdev` display, of how to configure the square (`CART`) tape device `rss000` up and available to users. You may want to put the commands in this step into scripts so that your operators can simply execute them.

```
sn5111# /etc/tpconfig rss000 up
```

4. Confirm that the tape device is now configured up and available for use.

   Now when the `tpstat` command display is examined, the status for `rss000` is `idle`, that is, available for use, instead of `down` (unavailable).

```
sn5111# /bin/tpstat
userid   jobid dgn      a stat dvn      bx i rl ivsn   evsn   blks  NQSid
               EXB      + down rpe02    03 0
               TAPE     + down rsm00    04 2
               TAPE     + down rsm10    05 3
               CART     + idle rss000   06 3
               CART     + down rss010   07 3
```

Also, in the following output from the `tpstat` command, note that the device status (`dvst`) is `up`, (available for use) instead of `down` (unavailable).

```
sn5111# tpdev
  dev name  dev grp   bnk  did  ios  iop  dvst  cu   cust  ch   chst  loader
  ========  ========  ===  ===  ===  ===  ====  ===  ====  ===  ====  ========
     rpe02  EXB         0    0    0    0  down    0  down  20   up    Operator
     rsm00  TAPE        1    0    2    0  down    0  down  21   up    Operator
     rsm10  TAPE        2    0    3    0  down    0  down  22   up    Operator
    rss000  CART        3    0    3    0  up      0  up    23   up    Operator
    rss010  CART        3   10    3    0  down    0  up    23   up    Operator
```

Operator display utility: /usr/lib/msg/oper

15.6

The following commands reside in `/usr/lib/msg`:

`oper` built-in command	Description
`ref` *time*	Sets the refresh rate (in seconds) for the refresh display to a specified length of time. The minimum refresh rate is 5 seconds.
`redraw`	Redraws screen.
`snap` [*file*]	Sends a copy of the screen to *file*.
`exit`	Exits from `oper`.
`?`	Help.
-	Displays previous page.
+ or space bar	Displays next page.
`infd`	Displays informative messages. See the `infd`(8) and `msgd`(8) man pages for more information.
`msgd`	Displays action messages from users, such as tape mount messages. See the `msgd`(8) man page for more information.
`rep` *msgnumber* [*reply_string*]	Allows an operator to respond to an action message, such as a tape mount message.
`oper`	Starts a refreshing display of operator messages being handled by the `msgdaemon`. These typically include tape mount requests.
`operator`	Anyone with a special *operator* group ID. The default `group` is `operator`.

The operator display provided by the `/usr/lib/msg/oper` command can be run from any terminal defined in `/usr/lib/terminfo`; but it requires at least 80 columns and 24 lines. The three lines at the bottom are used for input and for running commands that do not display information on the screen. The rest of the screen is used as a refreshable display to show messages and to run other display commands.

The following is an example of the `oper` display. The last line, which starts with >, is the command entry line; the `snap /tmp/mlmdraw` is the last command that was given to the `oper` utility. All lines above the command entry line are in the refreshable display area.

A sample `oper` display follows:

```
Command: msgd  Page: 1 [delay 10] Fri Feb  4 12:32:50 1994


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


   2     12:32    TM046 - mount tape VSN(unknown) on device rss000 for mlm

                    3843,  () or reply cancel / device name



Enter '?' for help.

> snap /tmp/mlmdraw
```

Default operator commands file, `oper.rc`

15.7

The `oper.rc` file is the default file for operator commands. As administrator, you also can place this file in the user's `home` directory (as `.operrc`).

A sample `/usr/lib/oper.rc` file follows.

```
1 #
2 #  System default "oper" configuration file
3 #  ****************************************
4 #
5 #  Commands to be executed as refreshing displays
6 #
7  cal
8  cat
9  df
10 egrep
11 env
12 fgrep
13 find
14 finger
15 grep
16 infd
17 ls
18 man
19 msgd
20 news
21 ps
22 qstat
23 tprst
24 tpstat
25 tpgstat
26 who
27 whom
28 #
29 #  Commands that require full control of the screen
30 #    must be preceded with ":"
31 #
32 :csh
33 :dis
34 :ed
35 :edit
36 :ex
37 :ksh
38 :mem
39 :more
40 :oper
41 :pg
42 :qmgr
43 :screen
44 :sh
45 :vedit
46 :vi
47 :view
```

Autoloader support
15.8

An autoloader is a tape subsystem that automatically mounts and unmounts cartridge tapes. Two classes of autoloaders exist:

- Stacker devices; allows tapes to be preloaded and used sequentially.
- Random-access devices; allows specified volumes to be mounted on specified devices.

Stacker devices
15.8.1

UNICOS supports the StorageTek 4480 Scratch Loader.

Random-access devices
15.8.2

UNICOS supports the following random-access devices:

- StorageTek 4400 Automated Cartridge System (ACS).
- EMASS ER90 helical scan tape drive and its associated autoloader.

Procedure: Installing an autoloader

To use the Sun (UNIX) version of the autoloader, you must perform the following steps:

1. Build your system with TCP/IP turned on in the `/etc/config/config.mh` file; that is, `/etc/config/config.mh` must contain the following lines:

   ```
   #define CONFIG_TCP 1
   #define CONFIG_RPC 1
   ```

   By default, this is already defined for you on CRAY J90 and CRAY EL systems.

2. Define the UNIX storage server host name in the the following places:

   a. Local `/etc/hosts` file.

   b. `Server_Name` parameter of the loader definition in the `/etc/config/tapeconfig` file.

3. Because the communication protocol between the network daemon (`stknet`) and the storage server uses the TCP remote procedure call (RPC) service, this must be indicated in **one** of the following places:

   a. `/usr/ACSSS/rc.acsss` file.

   b. UNIX storage server host.

   This is done by setting the `CSI_UDP_RPCSERVICE` and `CSI_TCP_RPCSERVICE` parameters to `TRUE`.

4. Define the vendor address for each affected device in the `/etc/config/tapeconfig` file, by either using the menu system or doing it manually.

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

   ```
   Configure System
   ...Tape Configuration
   ......Configure Tape Bank(s)
   .........Configure Tape Device(s)
   ```

   In the `Configure Tape Device(s)` menu, define the `vendor loader address` as *a, b, c, d,* or *e,* which have the following meaning:

   a = ACS *b* = LSM *c* = panel *d* = drive *e* = robot

   If you are not using the menu system, edit the `/etc/config/tapeconfig` file to add the following statement under each affected `DEVICE` statement:

   `vendor_address =` *(a, b, c, d, e)*

   with *a, b, c, d,* and *e* defined as above.

UNICOS Menu System Overview [A]

This appendix includes information about the following:

- Accessing and initiating the menu system
- Selecting software components to maintain through the menu system
- Using menu prompts
- Using menu keys
- Using menu definition files
- Sample process of using a menu
- Restoring a configuration
- Viewing the `/etc/install/install.log` log file

This appendix provides a brief description of the UNICOS Installation and Configuration Menu System (referred to as the menu system in this guide), which, with system start-up scripts, provide a uniform way to configure and start the various system and network utilities.

Caution

When you are upgrading your Cray Research products software and you want to run the menu system but have never run it before at your site, you must import and activate all parameters. To perform these functions, refer to the Utilities submenu of the main menu. This allows you to maintain your existing configuration files instead of using default configuration files included with a Cray Research product release.

Accessing and initiating the menu system
A.1

The files and scripts that compose the menu system are grouped in the `/etc/install` directory. To access and initiate the menu system, enter the following command lines:

```
cd /etc/install
./install
```

To execute the `./install` script, you must have super-user permission.

Note

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

Selecting components to maintain by using the menu system
A.2

You can use the menu system to maintain all or selected portions of your system configuration files. Select the following menu and set the components you want to maintain by using the menu system to `YES` (the menu system will access the configuration files for these components):

```
UNICOS Installation / Configuration Menu System
      Configure System ==>
            Configurator Automation Options ==>
```

If you elect to maintain a component manually, set that component to `NO`. All related menus for that component will be disabled; files cannot be imported into or activated from the menu system.

Caution

If you change any settings in the following file, you must rebuild your kernel:

```
UNICOS Installation / Configuration Menu System
    Configure System ==>
       Major Software Configuration ==>
```

You also should verify that the components you have set to `YES` in the `Configurator Automation Options ==>` menu also are set to `on` in the following menu:

```
UNICOS Installation / Configuration Menu System
      Configure System ==>
           Major Software Configuration ==>
```

Caution

You should **always** import, modify, and/or activate a configuration from the same menu. Although a component's menus may be disabled, if you execute the `Import ...` and `Activate...` lines of the `Configure System` menu, you will import and activate that component's default configuration along with all other default configurations of components that are set to `on` in the `Major Software Configuration ==>` menu.

To import an existing configuration file into the menu system, execute the `Import ...` line of a menu. After you have imported a file, you can modify the configuration within the menu system. When you have the configuration you want, execute the `Activate...` line of the menu. The activation process writes the configuration into the menu system's internal database files to the new `root (/)`.

Menu prompts
A.3

On the left side of a menu display you will be prompted with the following character strings:

`M ->` Means pressing a carriage return will display a submenu. Each menu display corresponds to a `/etc/install/`*xxx*`.mnu` file written in menu specification language (MSL). Over 30 menu displays exist.

`E ->` Means pressing a carriage return will take the horizontal fields to the right of the prompt and display them vertically as a selection list. These values are stored in a `.cfg` file in the `/etc/install/cfdb` directory.

`S ->` Means pressing a carriage return will move the prompt to the far right column, allowing the user to edit the value or tab through a list of valid selections. These values are stored in `.sav` files in the `/etc/install` directory.

`A ->` Means pressing a carriage return will invoke the action described, such as loading a tape or invoking a build.

`N/A` Means the current selection is not applicable because of previous selections.

Menu keys
A.4

To access information regarding a specific menu, use the keys provided at the bottom of each menu system screen.

Example:

```
                        Disk Configuration

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


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

Key	Description
`^? Commands`	Displays a list of currently active command keys that you can enter from the tool.
`H Help`	Accesses the help screen for this particular menu. It gives you explanations for each line.
`Q Quit`	Allows you to get out of the menu system from this screen.
`V ViewDoc`	Allows you to get information from a man page or Docview from within the menu system.
`W WhereAmI`	Displays the list of menus that you have traversed.

For information about the menu prompts, maneuvering keys, and function keys, see the *UNICOS Installation Menu System Reference Card*, publication SQ–2411. To change the value of a `selection` item in a menu, use one of the input keys. Input keys are set to emulate the functions of one of the text editors, which you choose in the `Preferences menu`.

For detailed information about the menu system, see the *UNICOS Installation and Configuration Tool Reference Manual*, publication SR–3090.

Menu definition files

A.5

There are menu definition files that use the menu specification language (MSL) to define the menus displayed by the menu system program (`inmenu`). Menu definition files are identified by the file-name suffix `.mnu`. The following are the basic directories, files, and scripts that the menu system uses:

Path/File	Description
`/etc/config`	Directory that contains the start-up configuration files that the menu system maintains.
`/etc/install`	Directory that contains all menu system interface scripts (`*.sh` and `*.mnu`) and some of the current menu system values (`*.sav`).
`/etc/install/listings`	Directory used by the menu to hold listings that reflect errors.
`/etc/install/cfdb`	Directory of form files (`.cfg`) that the menu system modifies and uses to replace the corresponding system files (`hosts`, `ldchlist`, and so on).
`/etc/install/editions`	`cpio` archives of menu system changes (all `.sav`, `cfg`, and `/dev` changes).

Sample process of using a menu

A.6

To use the menu system to make a change to the `/etc/fstab` file, you must traverse these menus:

```
Configure System ==>
      File System (fstab) Configuration==>
            Standard File System Configuration==>
```

Make the changes for `/etc/fstab` on this menu. When you have completed your changes, exit this submenu. You are prompted as follows:

```
Do you want to update the form file?  y/n
```

Answering `yes` updates **only** `/etc/install/cfdb/fsmf.cfg`; answering `no` eliminates any changes you made on this menu.

If you want the change dispersed into the real system source, you must execute the following action entry:

```
Activate the configuration
```

Activating the configuration overwrites `/etc/fstab` with the contents of the form file `/etc/install/cfdb/fsmf.cfg`. The menu system displays which files it will update, in this case `/etc/fstab`, and prompts you with the following:

```
continue? y/n
```

This question means that you must give permission for copying the `fsmf.cfg` file to `fstab`. If you answer `yes`, it will copy the file; if you answer `no`, it will not copy the file. This is your last chance to back out.

A `cpio` archive file also is created each time you activate a configuration and a file is actually updated. The `cpio` archive file is a snapshot of all menu system settings; therefore, you can use it at a later date to restore the menu system to a specific state.

The `cpio` archive file contains the `.sav` and `.cfg` files, and all special files in the `/dev` directory. The `cpio` archive is in `/etc/install/editions` and can be restored in `/tmp`, where you can examine or copy all or some files to the appropriate directory.

Restoring a configuration
A.7

Occasionally, you may need to restore a prior iteration of the menu system's configuration files. You can use the menu system for configuration management. Use the following menu sequence to do the following:

- List each stored configuration edition
- Compare two configuration editions
- Extract either a complete edition or individual configuration files within an edition
- Compress the files
- Convert pre-UNICOS 8.0 editions
- Print the listing of available configuration editions
- Store a complete edition of the current system configuration (a snapshot of the system configuration files in their current state)

```
UNICOS Installation / Configuration Menu System
M->   Utilities ==>
M->         Configuration editions utility ==>
```

Viewing the /etc/install/install.log log file
A.8

The menu system provides a log file, `/etc/install/install.log`, which you can use to monitor actions, including any errors and problems. To examine this file within the menu system, use the following menu sequence:

```
UNICOS Installation / Configuration Menu System
      Utilities ==>
            Inspect the installation log . . .
```

To view this file from outside the menu system, enter the following command:

```
$ more /etc/install/install.log
```

Frequently Used Commands [B]

This appendix provides a brief description of several frequently used system administration commands, scripts, and files. For more details about these commands, see the online man page for the command or consult one of the following man page manuals:

- *UNICOS User Commands Reference Manual*, publication SR–2011
- *UNICOS System Calls Reference Manual*, publication SR–2012
- *UNICOS File Formats and Special Files Reference Manual*, publication SR–2014
- *UNICOS Administrator Commands Reference Manual*, publication SR–2022
- *CRAY IOS-V Commands Reference Manual*, publication SR–2170
- *CRAY EL Series IOS Commands Reference Manual*, publication SR–2408

Commands available from the IOS console

B.1

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

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

Command/Script/File	When to use
/bin/dflawr /bin/dflaww /bin/dformat /bin/dslip /bin/dsurf /bin/dverify	Commands that aid in disk flaw handling.
/bin/ed	Command to use when editing files on the CRAY EL IOS SCSI disk or the CRAY J90 system console.
/bin/enstat	A command to give Ethernet addresses attached to that IOS.
/bin/mfdump	A command that dumps data from the mainframe if a system crash or hang occurs.
/bin/mt	A command that manipulates tape devices without using the UNICOS tape daemon (tpdaemon).
/bin/reload	A command that reboots the IOS. If the IOS is not running (that is, if the boot prompt is displayed), use the load command instead.
/bin/whatmic	A command that displays information about the microcode in use on certain IOS peripherals.
/config	IOS configuration file.
df	A command that displays the amount of free space left on the IOS SCSI disk or on the CRAY J90 system console disk. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
iosdump	A command that dumps data from the IOS if a system crash occurs; this command is available from the IOS software and the IOS PROM. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
load	A PROM command that boots the IOS (available only at the boot prompt; see also /bin/reload). This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
rcmd	A command that initiates execution of another specified command on a slave IOS. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.

Command/Script/File	When to use
`/sys/param`	UNICOS configuration file (`/ios-param` is a copy of this file). A CRAY J90 IOS-V is case sensitive, so this file must be referenced in all lowercase. A CRAY EL IOS converts all characters to uppercase for the user, so this file may be referenced in either uppercase or lowercase on a CRAY EL IOS.
`time`	A command to check or change the IOS's date and time clock. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
`/tmp/scb.log`	For CRAY EL systems. A detailed log file that contains messages generated by the last execution of the `scb` (scan chain builder) command.
`version`	A command to display the version number of the running system. If entered at the IOS prompt, the IOS software version is displayed; if entered at the boot prompt, the IOS PROM version is displayed. This command is built into the IOS kernel; no leading path name is used to invoke or specify it.
CONTROL-a	Terminal key sequence used to toggle between IOS and UNICOS consoles. CONTROL-a toggles between the IOS and UNICOS prompts. When going from the UNICOS prompt, the prompt changes to the IOS prompt after you press CONTROL-a. When going from the IOS prompt, the prompt does not change until you press RETURN.

Commands available from the UNICOS console

B.2

The following commands and scripts are available from the UNICOS console:

Command/Script/File	When to use
`/bin/mkdir`	A command that creates a subdirectory.
`/bin/tpstat`	A command that displays the current status of tape devices under control of the tape daemon (`tpdaemon`).

Command/Script/File	When to use
`/etc/brc`	A command that resets the `mnttab` file so a file system can be mounted.
`/etc/bcheckrc`	A command that checks file systems to be mounted during setup.
`/etc/chown`	A command that changes the ownership of a file.
`/etc/config/rcoptions`	A command used to alter the `/etc/rc` script.
`/etc/config/tapeconfig`	A file used to configure the UNICOS tape daemon (`tpdaemon`).
`/etc/coredd`	A command that copies raw core dump files to a regular UNICOS file in a separate file system.
`/etc/cpdmp`	A command that copies the dump from the dump directory to a file for further processing.
`/etc/crash`	A command used to analyze a dump file.
`/etc/csaboots`	A command that writes the boot record to the `/etc/casinfo` file.
`/etc/df`	A command used to check the amount of disk space available ("disk free").
`/etc/dump`	A command used to perform full or incremental backups.
`/etc/errpt`	A command used to display errors reported in the system's error file.
`/etc/fsck`	A command used to verify the consistency of a file system.
`/etc/init`	A command that signals the `init` process to change to a different run level.
`/etc/nu`	An interactive command used to add users. This command prompts you for account information such as the user password, login ID, and so on.
`/etc/passwd`	A command used to change a password.
`/etc/rc`	A script run automatically at start-up time that resets the system to multiuser mode.
`/etc/shutdown`	A command that puts UNICOS into single-user mode.

Command/Script/File	When to use
`/etc/setdev`	A command run automatically at start-up time that removes and remakes disk special files.
`/etc/udbgen`	A command that alters the user database.
`/usr/bin/chgrp`	A command used to change the group ownership of a file.
`/usr/bin/chmod`	A command used to change permissions on a file or directory.
`/usr/bin/du`	A command used to check disk usage statistics.
`/usr/bin/kill`	A command used to terminate a process.
`/usr/bin/mkdir`	A command used to create a directory in the current directory.
`/usr/bin/ps`	A command used to check status of active processes.
`/usr/bin/rmdir`	A command used to remove a specified directory.
`/usr/bin/who`	A command used to list information about logged-on users.
`/usr/lib/acct/startup`	A command that enables you to track per-process usage.
`/usr/lib/acct/ckpacct`	A command that checks the size of the accounting data files.
`/usr/lib/acct/ckdacct`	A command that checks the size of daemon accounting files.
`/usr/lib/acct/csarun`	A command that produces data file and accounting reports.
CONTROL-a	Terminal key sequence used to toggle between IOS and UNICOS consoles. CONTROL-a toggles between the IOS and UNICOS prompts. When going from the UNICOS prompt, the prompt changes to the IOS prompt after you press CONTROL-a. When going from the IOS prompt, the prompt does not change until you press RETURN.

File Version Numbers [C]

In the course of normal system administration, occasions exist when it is important to make a new version of a file, but it is also important to keep an old file. A sequence of operations often used to replace a real/production file with a new one and keep an old version of a file is as follows:

```
cp file.REAL file.NEW
edit file.NEW
cp file REAL file.OLD
mv file.NEW file.REAL
```

The preceding sequence can lead to problems; if a small error was made in the generation of the new file and a subsequent version is made, reusing the sequence will cause the loss of the previous real file. Because this is a well-known problem, you should use one of the following two sequences. To correct the error, use the following command lines:

```
cp file.REAL file.NEW
edit file.NEW
cp file.NEW file.REAL
```

To start again but to keep a copy of the broken new file, use the following command lines:

```
cp file.OLD file.NEW
cp file.REAL file.OLD2
edit file.REAL
```

A better strategy is to dispense with the `.OLD` file naming convention and use the following sequences. The first time you want to alter a file, use the following sequence:

```
cp file.REAL file.000
cp file.000 file.001
edit file.001
```

Each time you are ready to go live with the latest version of a file, copy the highest number file to *file* `.REAL`.

This method of file version numbering has three main advantages over the `.OLD` file naming scheme:

- You can quickly see a version history.
- You can make as many versions of the file as you like without losing the real file.
- You have a back-up copy of the real file in case it gets damaged in production.

Each time you make a new version, you can add a comment in the file's history file, as follows:

```
echo "file.00x version comment" >> file.HISTORY
```

Cleaning Tape Units [D]

This appendix contains information about standard use and care procedures for the tape hardware on CRAY J90 and CRAY EL systems.

Note

For complete information, see the documentation provided with your tape subsystem.

Cleaning the EXB-8500

D.1

EXABYTE tapes are supported only on CRAY EL systems. You should clean the EXB-8500's heads and tape path on a regular basis. The only cleaning material authorized for use with the EXB-8500 is an EXABYTE or EXABYTE-approved 8-mm cleaning cartridge. The cleaning cartridge is sent with each CRAY J90 or CRAY EL system that has an EXABYTE tape unit. The cleaning cartridge should be located along with the nine blank tapes sent with the system.

If you need another EXABYTE cleaning cartridge, you can order it from Logistics under P/N 90277500. This part number is for the cleaning cartridge alone; it does not include the nine blank tapes.

Caution

Using cloth swabs, cotton swabs, cleaning agents, or cleaning cartridges not approved by EXABYTE Corporation will void the warranty on the EXB-8500.

To determine how often to clean the EXB-8500, use the following guidelines:

- When using the EXB-8500 to read and write data in EXB-8500 format, clean the tape heads and tape path once a month or after 60 Gbytes of data transfer, whichever occurs first. For planning purposes, approximately 2 Gbytes of data are transferred per hour of operation in EXB-8500 mode.
- When using the EXB-8500 to read and write data in EXB-8200 format, clean the tape heads and tape path once a month or after 30 Gbytes of data transfer, whichever occurs first. For planning purposes, approximately 1 Gbyte of data is transferred per hour of operation in EXB-8200 mode.

To use the cleaning cartridge, follow these steps:

1. Apply power to the EXB-8500. When the power-on self-test is complete, press the unload button and remove any data cartridge in the EXB-8500. Leave the door open.
2. Place the cleaning cartridge in the EXB-8500 and close the door. The EXB-8500 performs the remainder of the cleaning cycle automatically. When the cleaning cycle is complete, the cleaning cartridge is unloaded and ejected from the EXB-8500. The average cleaning cycle is 15 seconds.
3. Record the date the cleaning was performed on the cleaning cartridge label.
4. Store the cleaning cartridge for future use.

Note

If the cleaning cartridge is ejected from the EXB-8500 without performing a cleaning cycle (that is, before 15 seconds), the cleaning cartridge has reached the end of its useful life, and you should discard it.

Caution

To prevent contamination of the EXB-8500, do not use the cleaning cartridge for more than the number of cleaning cycles specified on the cartridge label.

You should use the cleaning cartridge after 60 Gbytes of data have been transferred or once a month of normal usage.

Cleaning the Digital Audio Tape (DAT)

D.2

When the right Clean/Attention light on the Digital Audio Tape (DAT) flashes amber, you should clean the tape heads.

To clean the heads, use Cleaning Cartridge 90334800, which you can order from Logistics (or you can use the HP 92283K Cleaning Cartridge). The Cray cleaning cartridge package includes two tapes.

To use the cleaning cartridge, follow these steps:

1. Insert the cleaning cartridge into the drive. The drive automatically takes the cartridge, loads, it, and cleans the heads.
2. After about 30 seconds, the drive ejects the cartridge.

   If the cartridge is ejected after only about 14 seconds, this means the cartridge has reached the end of its useful life, and no cleaning has occurred. Discard the cartridge, and repeat the cleaning operation with a new cleaning cartridge.
3. Take the cartridge out of the drive, and write the date on the label on the cartridge. A cartridge usually has a life of 25 cleaning cycles.

If the Cleaning Needed signal reappears, the cartridge is nearing the end of its useful life; copy the data on the cartridge onto a new one and discard the old cartridge. After you have cleaned the heads successfully, the Cleaning Needed signal will be cleared.

Rather than waiting for the Cleaning Needed signal to appear on the front panel, you should clean the heads according to the following table:

Tapes used per day	<1	1	2	3	4	5
Cleaning interval	weekly	weekly	twice weekly	twice weekly	daily	daily

Cleaning the QIC tape (Anaconda 2750)

D.3

QIC tapes are supported only on CRAY EL systems. When reinserting a cartridge into the drive, you should be aware that when the tape is unloaded, it is partially ejected from the drive. Before you reinsert it, you must physically remove the tape from the drive.

Use the Cleaning Cartridge Kit for 1/4" Data Cartridge Tapes. This kit includes a fluid that you saturate on the pads of the cleaning cartridge before inserting the cartridge in the QIC device (for complete information about the amount of fluid to use, see the directions).

You should clean the head assembly after 8 hours of normal use, after an initial pass with a new cartridge, or if excessive errors occur.

Note

Users may encounter problems when loading the same QIC cartridge after it is accessed (write or read) and then unloaded. When a tape is unloaded, it is only partially ejected. To reinsert the same data tape, you must completely remove the tape from the clutches of the drive before you reinsert it (grasp the cartridge, pull out slightly, then use light pressure to reinsert).

Cleaning the 3480 (StorageTek 4220)

D.4

This subsystem displays a message on the LED when cleaning is required (the word CLEAN appears on the display). After the host processor unloads the current data cartridge, insert the cleaning cartridge (shipped with the system). The device recognizes the cleaning cartridge and initiates the cleaning cycle. The cleaning cartridge is unloaded after the 15-second cleaning cycle completes.

You should replace the cleaning cartridge after 500 uses.

Cleaning the 9-track tape (StorageTek 9914)

D.5

The StorageTek 9914 streamer requires no preventative maintenance, but it does require routine cleaning. You should clean the heads daily (if the system is used continuously). To clean, pull the device fully out from its rack, release the three thumbscrew fasteners, and raise the tape path cover to its fullest. Use a cleaning material and solvent as recommended in the subsystem documentation.

Disk Capacities and Transfer Rates [E]

This appendix contains information about disk device capacities and transfer rates.

Disk devices for formatted drives (CRAY EL systems)

E.1

The numbers shown in Table 14 for disk devices are for formatted drives on CRAY EL systems only.

Table 14. Disk devices for formatted drives (CRAY EL systems only)†

Device type	DR1 (removable ESDI) and DD-3 (ESDI)	DDAS2 (10 DD-3 disk drives)
Bytes/sector (bytes/block)	4,096	4,096
Words/sector (words/block)	512	512
Sectors/track (blocks/track)	10	75
Tracks/cylinder	15	N/A for striped device
Cylinders/spindle (cylinders/ device)	2,228	N/A for striped device
Bytes/device	1,368,883,200 1.3 Gbytes	10,248,192,000
Words/device	171,104,000	1,281,024,000

† 4096 bytes = 512 words = 1 block = 1 sector = 1 click

Table 14. Disk devices for formatted drives (CRAY EL systems only)† (continued)

Device type	DR1 (removable ESDI) and DD-3 (ESDI)	DDAS2 (10 DD-3 disk drives)
Sectors/device (blocks/device)	334,200	2,502,000
Transfer rate:		
Words/sec	250,000	1,687,500
Bytes/sec	2.0 Mbyte/s	13.5 Mbyte/s

File storage devices (CRAY EL systems)

E.2

You may define and mount one or more file systems on disk devices. The devices shown in Table 15 are the main file storage devices on CRAY EL systems.

Table 15. Main file storage devices on CRAY EL systems†

Device type	DD-4 (IPI-Sabre-7)	DDRAM (central memory)
Bytes/sector (bytes/block)	4,096	4,096
Words/sector (words/block)	512	512
Sectors/track (blocks/track)	28 ††	N/A for central memory
Tracks/cylinder	9	N/A for central memory

† 4096 bytes = 512 words = 1 block = 1 sector = 1 click

†† Does not include the additional sector per track, which is a "slip" or "spare" sector

Table 15. Main file storage devices on CRAY EL systems† (continued)

Device type	DD-4 (IPI-Sabre-7)	DDRAM (central memory)
Cylinder/spindle (cylinders/disk	2,593	N/A for central memory
Bytes/device	2,676,473,856	Depends on system memory size
Words/device	334,559,232	Depends on system memory size
Sectors/device (blocks/device)	653,436	Depends on system memory size
Transfer rate:		
Words/second	1,162,500	
Bytes/second	9.3 Mbyte/s	

† 4096 bytes = 512 words = 1 block = 1 sector = 1 click

†† Does not include the additional sector per track, which is a "slip" or "spare" sector

DD-5I disk drives (CRAY J90 and CRAY EL systems) E.3

The DD-5I disk drive, supported on and CRAY J90 and CRAY EL systems, is a high-performance, two-head, parallel enhanced IPI-2 drive with a peak unformatted transfer rate of 12.4 Mbyte/s and a peak formatted transfer rate of 9.5 Mbyte/s. Its formatted capacity is 2.75 Gbytes. The disk drive is zoned bit recorded, which means that bits are stored on the drive at different densities, depending on the location or "zone." This increases capacity, but it causes varying data rates, from 6 to 8.5 Mbyte/s sustained.

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

Reliability of the DD-5I disk drive is exceptionally high with a mean time between failures (MTBF) of 300,000 hours of power-on operation. No preventive maintenance is required. Table 16 shows DD-5I specifications.

Table 16. DD-5I specifications (CRAY J90 and CRAY EL systems)

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

DD-5S disk drives (CRAY J90 and CRAY EL)

E.4

Table 17 shows the specifications for DD-5S (SCSI) disk drives.

Table 17. DD-5S specifications (CRAY J90 and CRAY EL systems)

Unformatted capacity	3.5 Gbytes
Formatted capacity	2.98 Gbytes
Formatted capacity (in blocks)	781,000 (4-Kbyte blocks)
Number of disk platters	11
Data surfaces	21
Interface	SCSI-2 Fast Wide
Transfer rate (peak)	6 Mbyte/s
Transfer rate (sustained)	3.2 Mbyte/s – 5 Mbyte/s
Average access time	11.5 ms
Maximum access time	23.5 ms
Minimum access time	1.7 ms
Average latency time	5.55 ms
Disk speed	5,400 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	2,738

DD-6S disk drives (CRAY J90 systems)

E.5

Table 18 shows the specifications for DD-6S (SCSI) disk drives.

Table 18. DD-6S specifications (CRAY J90 systems only)

Unformatted capacity	10.8 Gbytes
Formatted capacity	9.11 Gbytes
Formatted capacity (in blocks)	2,389,000 (4-Kbyte blocks)
Number of disk platters	14
Data surfaces	27
Interface	SCSI-2 Fast Wide

Table 18. DD-6S specifications (CRAY J90 systems only) (continued)

Transfer rate (peak)	7.2 Mbyte/s
Transfer rate (sustained)	4.2 Mbyte/s – 6.2 Mbyte/s
Average access time	11.5 ms
Maximum access time	24 ms
Minimum access time	1.7 ms
Average latency time	5.55 ms
Disk speed	5,400 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	4,925

DD-7S disk drives (CRAY J90 systems)

E.6

Table 19 shows the specifications for DD-7S (SCSI) disk drives.

Table 19. DD-7S specifications (CRAY J90 systems only)

Unformatted capacity	5.06 Gbytes
Formatted capacity	4.2 Gbytes
Formatted capacity (in blocks)	1,102,000 (4-Kbyte blocks)
Number of disk platters	10
Data surfaces	21
Interface	SCSI-2 Fast Wide
Transfer rate (peak)	7.25 Mbyte/s
Transfer rate (sustained)	7.25 Mbyte/s
Average access time	8.5 ms
Maximum access time	19 ms
Minimum access time	0.9 ms
Average latency time	4.17 ms
Disk speed	7,200 r/min
Bytes per track	Varies by zone
Bytes per cylinder	Varies by zone
Cylinders	3,711

Disk Flaw Handling [F]

This appendix describes how to prepare CRAY EL ESDI and IPI drives for using the disk flaw handling utilities and shows examples of using the `dsurf` and `dslip` utilities for disk flaw handling.

After you have prepared the drives for disk flaw handling, the software will save all flaw information automatically. You then can use this flaw information on subsequent formats (if desired), so that no flawing information is lost.

The following utilities support the disk flaw handling capability:

- The `dsurf` utility provides a flexible method of performing disk surface analysis. It can perform read-only tests or pattern write and read tests.
- The `dformat` utility options `-l` *level*, `-s` *serial number*, and `-f` *file* use the saved flaw information. The `dformat` utility does not perform the `dverify` utility for IPI drives.
- The `dslip` utility `dslip C:` syntax allows blocks to be slipped on a CRAY EL IOS SCSI disk drive.
- The `dflawr` utility `-r` option reads the sector IDs on IPI and ESDI drives to report the sectors and tracks that have actually been slipped and mapped on the drive. Additional related `dflawr` utility options are `-f` *file*, `-l`, and `-s`.
- The `dflaww` utility options `E` (for ESDI DD3 disk drives), `I` (for IPI DD4 disk drives), `B` (for IPI DD5I disk drives), `S` (for SCSI DD5S disk drives), and `-f` *file* are used for disk flaw handling.

Note

The disk autoflawing feature automatically flaws unusable disk areas. It flaws all recovered read and write errors, as well as all write hard errors. A `syslog` entry then documents flaws that have been identified and handled. To enable autoflawing, you must add the flawing task, `/dev/flawd`, to the `/config` file on the CRAY EL IOS SCSI disk drive. This will autoflaw all disks on the system.

The following terms are used in this appendix:

Term	Description
Growth error table (GET)	This table shows all known flaws on the drive. For the ESDI DD-3, IPI DD-4 and DD-5I, and SCSI DD-5S disks, this table includes the OEM defect list, along with all flaws slipped and mapped by the user.
OEM defect list	This list shows all flaws found by the drive manufacturer; this list is stored on the disk permanently.

Preparing disks for flaw handling
F.1

This subsection describes how to prepare ESDI and IPI drives for using the disk flaw handling utilities.

DD-2 and DD-3 ESDI disk drives for CRAY EL systems
F.1.1

To prepare DD-2 and DD-3 ESDI disk drives for using the disk flaw handling utilities, you must do one of the following:

- Format the drive by using the `dformat` utility and the `-l 2` option, as follows:

  ```
  dformat E00 -l 2 -s <serial number of drive>
  ```

- Run the `dflawr` utility by using the `-r` option, and write the GET read from the drive to a file, as follows:

  ```
  dflawr E00 -r -s <serial number>
  ```

Then run the `dflaww` utility to write the GET to the disk, as follows:

```
dflaww E00
```

Note

When running the `dflawr` utility with the `-r` option on ESDI disk drives, the flaw block numbers found in the GET list may not always match the OEM list.

DD-4 IPI disk drives for CRAY EL systems
F.1.2

To prepare DD-4 IPI disk drives for using the disk flaw handling utilities, you must do one of the following:

- Format the drive by using the `dformat` utility and the `-l 2` option, as follows:

  ```
  dformat I00 -l 2 -s <serial number of drive>
  ```

- Run the `dflawr` utility by using the `-r` option, and write the GET read from the drive to a file, as follows:

  ```
  dflawr I00 -r -s <serial number>
  ```

  Then run the `dflaww` utility to write the GET to the disk, as follows:

  ```
  dflaww I00
  ```

Examples
F.2

This subsection shows examples of using the `dsurf` and `dslip` utilities for disk flaw handling.

dsurf *utility*
F.2.1

The dsurf utility has the following syntax:

```
dsurf Bcd [-adfirvw] [-l level] [-n blocks]
[-p passes] [-s start] [-t count]

dsurf C: [-adfirv] [-n blocks] [-p passes]
[-s start]

dsurf Ecd [-adfirvw] [-l level] [-n blocks]
[-p passes] [-s start] [-t count]

dsurf Icd [-adfirvw] [-l level] [-n blocks]
[-p passes] [-s start] [-t count]

dsurf Scd [-adfirvw] [-l level] [-n blocks]
[-p passes] [-s start] [-t count]
```

Option	Description
B	Indicates a buffered IPI drive.
C:	Indicates the CRAY EL IOS SCSI disk drive.
E	Indicates an ESDI drive.
I	Indicates an IPI drive.
S	Indicates a SCSI drive.
-a	Asks before flawing; default is to flaw silently.
-d	Debug on; does not flaw errors.
-f	Runs test until one pass completes without an error.
-i	Inhibits recheck on flawed errors.
-l	Test level: 0 Read test (default) 1 Eight-pattern write/read 2 Four-pattern random write/read
-n	Number of blocks to test; default is entire disk.
-p	Number of passes to run; default is 1.

Option	Description
`-r`	Does not flaw errors; default is to flaw.
`-s`	Starts block; default is 0.
`-t`	Reads/writes I/O size in sectors; default is one track.
`-v`	Specifies verbose mode.
`-w`	Allows writing without prompting; default is to prompt before writing.

The following example performs a read of the entire ESDI disk, slipping or mapping any bad sectors found:

```
IOS>dsurf E00
```

The following example performs an eight-pattern write/read on the entire IPI disk and runs until no errors are found in a pass:

```
IOS>dsurf I00 -vfl 1
```

The following example performs a read-only test starting at block 32,500 and reading until the end of the ESDI disk. The test does five passes and prompts the user to verify before slipping or mapping:

```
IOS>dsurf E00 -avp 5 -s 32500
```

The following example performs a read-only test of the entire CRAY EL IOS SCSI disk, slipping or mapping any defects found:

```
IOS>dsurf C:
```

`dslip` *utility*

F.2.2

The `dslip` utility has the following syntax:

```
dslip Bcd sector

dslip C: sector

dslip Ecd sector

dslip Icd sector

dslip Scd sector
```

The following example slips block 32,456 on the IPI disk drive:

```
IOS>dslip I01 32456
```

The following example slips block 3456 on the CRAY EL IOS SCSI disk:

```
IOS>dslip C: 3456
```

Note

The block number given by the `dflawr` utility for the CRAY EL IOS SCSI disk drive is a physical block number, which may not be the same as the logical block number given to the `dslip` utility.

Logical Device Cache Process [G]

Logical device cache is an optional feature that you may enable to reduce disk I/O wait time from a user's perspective. On CRAY J90 and CRAY EL systems, you define the logical device cache in central memory (DDRAM).

When a process issues a read request of data on a file system, the action taken to access the data depends on whether the data is currently in the UNICOS system buffer cache or logical device cache. The process is described as follows:

1. If the data is found in the system buffer cache (central memory), it is copied to the user area. If the requested data is not found, step 2 is taken.
2. If ldcache (logical device cache) has been allocated for the file system, the ldcache area is searched for the sector of data. If found, it is read into the system buffer cache and then copied to the user's process space. If the desired data is not found, step 3 is taken.
3. If ldcache is allocated for the file system, the sector is read from disk and cached into the ldcache area. The sector is then read from the ldcache device into the system buffer (central memory) cache and then copied to the user area.

Note

The system buffer cache may be bypassed if the data is a multiple of 512 words, begins on a word boundary, and the file system address of the data is on a block boundary.

The system buffer cache writes only to the ldcache area. When system buffers age and require reassignment, the system buffers are written to ldcache and the ldcache segment is marked as dirty. Dirty segments in ldcache are then written to disk when the segment is needed for a different part of the file system, when the ldcache area is flushed to disk by `ldsync`(8), or when the system periodically flushes the ldcache area to disk.

Setting up ldcache by using `/etc/ldcache`

G.1

The cache for a logical device is specified as a number of units and a count of 4096-byte blocks per unit. The system administrator easily configures the relationship between the number of cache units and the cache unit size. The `/tmp` and `root` (`/`) file systems are excellent candidates for logical device cache. If you have more ldcache area available, distribute the remaining area to other heavily used file systems. To be effective, the ldcache hit rate should be above 97% for ldcaching; however, the main concern is the ratio of logical reads to physical reads.

The `/etc/ldcache` command assigns groups of blocks, called *units*, of an ldcache device (central memory) to a specific file system. To set the number of blocks in an ldcache unit, use the `ldcache -s` command. Choose the size that is used in the `mkfs` command to build that file system. This makes reads and writes to that physical device much faster.

If a striped file system is cached, multiply the number of blocks per cylinder for the physical device type by the number of devices in the stripe group. Larger unit sizes are good for sequential I/O, but they may cause excessive I/O when the I/O is random.

Ensure that the number of blocks assigned for ldcache for all file systems added together does not exceed the total number of blocks available on your logical cache device. To calculate this figure, use the following steps:

1. For each file system being ldcached, multiply the number of blocks in an ldcache unit by the total number of ldcache units allocated for that file system.
2. Add all such totals together.
3. Subtract that sum from the total number of blocks available on the ldcache device for ldcaching.

Assigning ldcache
G.2

When assigning logical device cache, be sure to include the type. The MEM type is used when assigning central memory-based logical device cache. The LDCHCORE value defines the number of blocks of core memory to be used for logical device cache. The configuration specification language (CSL) NLDCH value defines the number of cache headers that will be configured. This sets the total number of logical device cache units that can be active at one time. You must use both the CSL LDCHCORE and NLDCH statements in conjunction to define central memory-based logical device cache.

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

`-l` *dev*	Full path name or minor device number of logical device.
`-n` *units*	Number of cache units to assign. If 0, the logical device caching is released.
`-s` *size*	Size (in 4-Kbyte blocks) of each cache unit. For best performance, set *size* as a multiple of tracks per cylinder related to the logical device and the file system used.
`-t` *type*	Type of memory for cache (MEM).

An example of releasing a logical device cache follows:

```
# /etc/ldcache -l /dev/dsk/user_a -n 0
```

An example of assigning a logical device cache follows:

```
# /etc/ldcache -l /dev/dsk/source-tree -s 27 -t MEM -n 500
```

You also can assign a logical device cache by creating an `/etc/config/ldchlist` file, which contains logical device cache configuration information used by `/etc/rc`. During multiuser startup, the `/etc/rc` script checks for the existence of an `/etc/config/ldchlist` file. If the file exists, `/etc/rc` will configure ldcaching according to the entries and values in the `/etc/config/ldchlist` file. There are four fields per line, separated by space; the first field is the logical device, the second field is the cache type (MEM), the third field is the number of

cache units, and the fourth field is the size in 4-Kbyte blocks of each unit (usually a track size). The following is an example:

```
/dev/dsk/root MEM 300 27

/dev/dsk/usr  MEM 300 27

/dev/dsk/tmp  MEM 300 27

/dev/dsk/home MEM 300 27
```

The third field multiplied by the fourth field is the total cache area (in blocks) allocated for that file system. The total of the third column is the number of NLDCH that you must define in the UNICOS `config` file.

An example of displaying the ldcache hit rate follows:

```
# /etc/ldcache
T unit size    reads     writes     hits     misses   rate   name
- ---- ----  ---------- --------  --------  -------  -----  -----
B  300   27    16727    30354      34865     1799     95.09  /dev/dsk/root
B  300   27    1729      4703       1399      254     84.63  /dev/dsk/home
B  250   27    6702     20794       6191      263     95.93  /dev/dsk/tmp
#
M  200   10      47        11         27        4     87.10  /dev/dsk/src

# ldcache -b
Cache to user    Cache to disk     Cache/disk ratio
Reads  Writes     Reads  Writes    Read  Write  Total    Name
-----  ------    -----  ------    ----  -----  -----    ----
839155 334505    65016  28772    12.9  11.6  12.5    /dev/dsk/root_b
301039  26871    25616   1628    11.8  16.5  12.0    /dev/dsk/usr_b
 68947  74725    13424  13824     5.1   5.4   5.3    /dev/dsk/spool
   183   1678    18416   1743     0.0   1.0   0.1    /dev/dsk/usr_tmp

# ldcache -l /dev/dsk/tmp
 /dev/dsk/tmp Fri Sep 24 14:52:12 1993
```

A hit rate of under 97% probably indicates that the file system is not a good candidate for ldcaching or that you should enlarge the size of that file system's ldcache area if possible. In the preceding display, you should examine the file system usage and

ldcache configuration aspects of the `/dev/dsk/home` and `/dev/dsk/src` file systems. You also should examine the ratio of logical reads to physical reads, as shown in the preceding display of the `ldcache -b` example.

An example of displaying ldcache statistics for an individual file system follows:

```
                          Read data      Write data
                          ---------      ----------
Blocks transferred:             689            1296
Avg request length:           1 blks          1 blks
Lst transfer rate:     0.008192 Mbs    0.061236 Mbs
Max transfer rate:     0.135680 Mbs    0.208438 Mbs
Cache hits:                     597             677
Cache misses:                     0              73
Cache hit rate:          1000.000000       90.266667
```

Flushing data by using `/etc/ldsync` G.3

You can use the `/etc/ldsync` command to flush data from all logical device caches to disk. Only data that has been written to a logical device cache, but not to disk, is affected. The `/etc/ldsync` command does not flush data in the system buffers to disk. During normal operation, the UNICOS system periodically flushes data from the ldcache area to disk; the `/bin/sync` command does this action.

During a normal UNICOS shutdown, all logical device cache data is flushed to disk. At shutdown time it is important that all `ldcache` is removed from all file systems. To check that all `ldcache` is removed, use `/etc/ldcache`. The command should print just a header, as in the following example:

```
ksh# /etc/ldcache

T  Unit  Size  Reads  Writes  Hits  Misses Rate  Name

ksh#
```

For additional information about when to execute the `/etc/ldsync` command when shutting down the UNICOS system, see the procedure in section 3, page 21.

CRAY J90/CRAY EL Software Differences [H]

To assist system administrators who may administer CRAY EL systems, this appendix includes the following information:

- Main software components differences between CRAY J90 systems and CRAY EL systems
- Feature differences between a CRAY J90 IOS (IOS-V) and a CRAY EL IOS
- Peripherals supported on CRAY EL systems but not supported on CRAY J90 systems
- Subsection H.4: Differences between CRAY J90 IOS-V commands and CRAY EL IOS commands

Main software components differences between CRAY J90 systems and CRAY EL systems

H.1

Table 20 indicates main software components differences between CRAY J90 systems and CRAY EL systems. In addition, because there may be differences in these areas from either the CRAY J90 or CRAY EL systems to other Cray Research CRAY Y-MP and/or CRAY C90 systems, these differences are also included in table 1.

Table 20. Main software components differences between CRAY J90 systems and CRAY EL systems

Software component	CRAY J90 systems	CRAY EL systems	Other CRAY Y-MP systems	CRAY C90 systems
Sun SPARCstation 5 and Sun Solaris software that has CD-ROM drive as system console	Yes	No	No	No
WYSE system console interface	No	Yes	No	No
OWS/OWS-E software	No	No	Yes	Yes
IOS-E software	No, CRAY J90-specific IOS and related software; however, it uses the IOS-E packet structure. Also see Table 2.	No, CRAY EL-specific IOS and related software; however, it uses the IOS-E packet structure. Also see Table 2.	Yes	Yes
Character-special tapes interface using `mknod` command	No	Yes	No	No
Network monitor	No	No	Yes	Yes
Hardware performance monitor (`hpm`) and related Perfview, Perftrace, and `perfdmp` tools	Yes	No	Yes	Yes

Table 20. Main software components differences between CRAY J90 systems and CRAY EL systems
(continued)

Software component	CRAY J90 systems	CRAY EL systems	Other CRAY Y-MP systems	CRAY C90 systems
Cray Ada programming environment	No	CRAY EL98 systems only	Yes	Yes, in Y-MP mode only
Pascal programming environment	No	CRAY EL98 systems only	Yes	Yes
Alternative path support for devices	No	No	Yes	Yes
USCP	No	No	Yes	Yes
SSD solid-state storage device and related software	No	No	Yes	Yes
Source code	No	No	Yes	Yes

Feature differences between a CRAY J90 IOS-V and a CRAY EL IOS

H.2

System administrator should note the following feature differences between a CRAY J90 IOS (IOS-V) and a CRAY EL IOS.

Table 21. Differences between a CRAY J90 IOS-V and a CRAY EL IOS

Feature	CRAY J90 systems	CRAY EL systems
System console	X Windows based/GUI interface; Sun SPARCstation 5. Also has command-line capability for remote support.	Command line/ASCII based; WYSE terminal.
Installation procedures	X Windows based/GUI interface. Combination CD and DAT tape media. Also has command-line capability for remote support.	Command line/ASCII based; tape media only.
File system on system console	Yes	No
Private Ethernet-attached console	Yes	No
Slave IOSs boot in parallel with the master IOS	Yes	No
IOS case sensitive	Yes	No, EL IOS converts all characters to uppercase.
`systat` command provides status of IOS network	Yes	No
SCSI disk attached to master IOS	No	Yes
Configuration maintenance	Changes made to IOS configuration files done on CRAY J90 system console by using Sun tools (that is, `vi` editor).	Changes made to IOS configuration files done on CRAY EL system console by using IOS tools (that is, `ed` editor).
Netblazer connection	Connected to Ethernet	Special interface board in Netblazer connected to IOSnet

Table 21. Differences between a CRAY J90 IOS-V and a CRAY EL IOS (continued)

Feature	CRAY J90 systems	CRAY EL systems
Modem connection	Connected to SPARCstation 5 serial port	A/B switch on serial port
Maximum number of peripheral controller boards per IOS	Four	Eight

Peripherals supported on CRAY EL systems but not supported on CRAY J90 systems

H.3

The following peripherals are supported on a CRAY EL system but are not supported on a CRAY J90 systems. Note: There are peripherals supported on both CRAY EL and CRAY J90 systems and also peripherals supported on CRAY J90 systems but not supported on CRAY EL systems that are not listed below.

- BMR
- DAS-2
- DD-3/DC-3
- DD-4/DC-4
- EMASS ER90
- EX-2
- FI-1
- HY-1
- IOBB-15
- IOBB-25
- IOBB-30
- QIC
- SI-1
- TC-2/TD-2
- TD-3

CRAY J90 IOS-V/CRAY EL IOS commands differences
H.4

The following subsections describe the differences between the CRAY EL IOS version 11.*x* and the CRAY J90 series IOS-V, version 1.*x*. Commands not listed in this subsection maintain the same functionality for the CRAY EL IOS and the CRAY J90 IOS-V.

CRAY J90 series IOS-V new commands
H.4.1

The following are commands supported on a CRAY J90 IOS-V only:

Command	Definition
`bbql`	Executes a quick-look IOBB diagnostic after power on
`bg`	Places in the background an IOS command that is suspended
`bootstruct`	Displays the boot environment of the IOS
`cc1test`	Executes a diagnostic test for the I/O channel card
`cc2test`	Executes a data transfer test from central memory to the I/O buffer board and back to central memory
`cc1ql`	Executes a quick-look I/O buffer board to and from I/O channel card diagnostic if first load after power on
`cc2ql`	Executes a quick-look central memory to I/O buffer board to central memory diagnostic if first IOS load after power on
`dd5sql`	Executes a quick-look SCSI drive diagnostic if first IOS load after power on
`dd5stest`	Executes acontroller comprehensive test and disk confidence test on any CRAY J90 supported SCSI disk(s)
`fg`	Places in the foreground an IOS command that is suspended or running in the background
`j90install`	Maintains and installs software on a CRAY J90 console, IOS-V, and mainframe

Command	Definition
`jbs`	Performs a boundary scan interconnect test on CRAY J90 series systems
`jcon`	Performs a remote login onto a CRAY J90 series mainframe
`jconfig`	Builds/edits the CRAY J90 configuration
`mm1test`	Executes a confidence test on the IOP RAM/CACHE memory
`rlogin`	Invokes a remote login
`tp1test`	Executes a confidence test on tape handlers

New command options H.4.2

The following commands contain new or changed options for a CRAY J90 IOS-V:

Command	Change
`ds`	*cpu* (now accepts CPU number from 0 through 15)
`load`	`-q` (quick load of IOS kernel to enable IOP disk or IOP file system maintenance to be performed)

Renamed commands H.4.3

The following commands are supported on a CRAY EL IOS under the old name and supported on a CRAY J90 IOS-V under the new name with no change in functionality:

Old name	New name
`eddtest`	`dd5stest`
`iob2test`	`bb2test`

Unsupported command options H.4.4

The following command options are supported on a CRAY EL IOS but are not supported on a CRAY J90 IOS-V:

Command	Unsupported options or subcommands
`crash`	subcommands: `das`, `esdi`, `ireq`, `pertec`, `stb PC A6_register`, `istat`, `itrace`, `s3560`, `sil`, `treq`
`dm`	`ll`, `lu`, `lr`, `ru`
`dflawr`, `dflaww`, `dformat`, `dstat`, `lm`	`D` (DAS drive), `E` (ESDI drive)
`dslip`, `dsurf`	`E` (ESDI drive)

Unsupported commands H.4.5

The following commands are supported on a CRAY EL IOS but are not supported on a CRAY J90 IOS-V:

- `act_shr`
- `bscan`
- `cdsktest`
- `clk`
- `comp`
- `cscan`
- `df`
- `ec`
- `eddql`
- `frff`
- `iosid`
- `ldf`
- `mkfs`
- `mount`
- `sc`
- `scb`
- `sync`
- `umount`

Power Up and Down Procedures [I]

This appendix includes the following procedures:

- How to power up and power down a CRAY Y-MP EL and CRAY EL98 system mainframe cabinet
- How to power up and power down a CRAY EL92 and CRAY EL94 system mainframe cabinet

See your CRAY J90 system hardware installation guide for power up and power down procedures for the CRAY J90 system.

Powering up/down a CRAY Y-MP EL or CRAY EL98 system

I.1

This subsection contains procedures for powering up and powering down CRAY Y-MP EL or a CRAY EL98 systems.

Powering up a CRAY Y-MP EL or CRAY EL98 system

I.1.1

Perform the following power-up procedure at the circuit breaker panel on the rear of the CRAY Y-MP EL and CRAY EL98 system mainframe cabinet.

DANGER

Ensure that system EMI shields and exterior panels are installed properly. If high voltage areas are exposed, serious injury may occur.

1. Ensure that the AC power plug is connected to power. One AC plug exists for each system cabinet.
2. Ensure that the AC Power Loss LED is illuminated on the control panel. See Figure 3, page 466.

Note

If the system has been powered down for 72 hours or more, the LED will not illuminate because the batteries for the LED have discharged. The batteries require approximately 36 hours to fully charge.

3. Ensure that the emergency power-off (EPO) button, located on the control panel, is extended outward by turning it counterclockwise 1/8 of a turn to the stop. If it is in, it will now pop out. See Figure 4, page 467.
4. Ensure that all individual component's power on buttons are enabled at this time. The locations of power-on buttons are as follows:
   a. The CPU and memory (MEM) buttons are located above the CPU and memory boards on the CPU card cage.
   b. The IOS ENABLE/INHIBIT button is located above the IOS VME card cage.
   c. The individual peripheral equipment drawers each have a power supply enable button located on the front of the drawer.
5. Move the circuit breaker on the back of each cabinet to the ON position (1), with the highest cabinet number being turned on first. For example, in a four-cabinet system, cabinet 4 would be powered on first, and the mainframe cabinet would be powered on last. Power on each cabinet within 2 seconds of each other. See Figure 5, page 467.
6. Visually inspect to ensure that no sparking or smoking is occurring among the components after you have powered up the system.
7. Ensure that the Capacitor Bank Ready LED for each cabinet is illuminated on the control panel. See Figure 3, page 466. The Capacitor Bank Ready LED for each cabinet takes approximately 1 minute to illuminate.
8. Wait for the System Ready LED to illuminate. Ensure that no yellow warning LEDs are illuminated. If there are, contact your service support personnel.

9. Ensure that the system console has been powered on and, if your system has an MWS (maintenance workstation-model EL), ensure the MWS-EL also is powered on.

For procedures to start (boot) the system, see section 3, page 21.

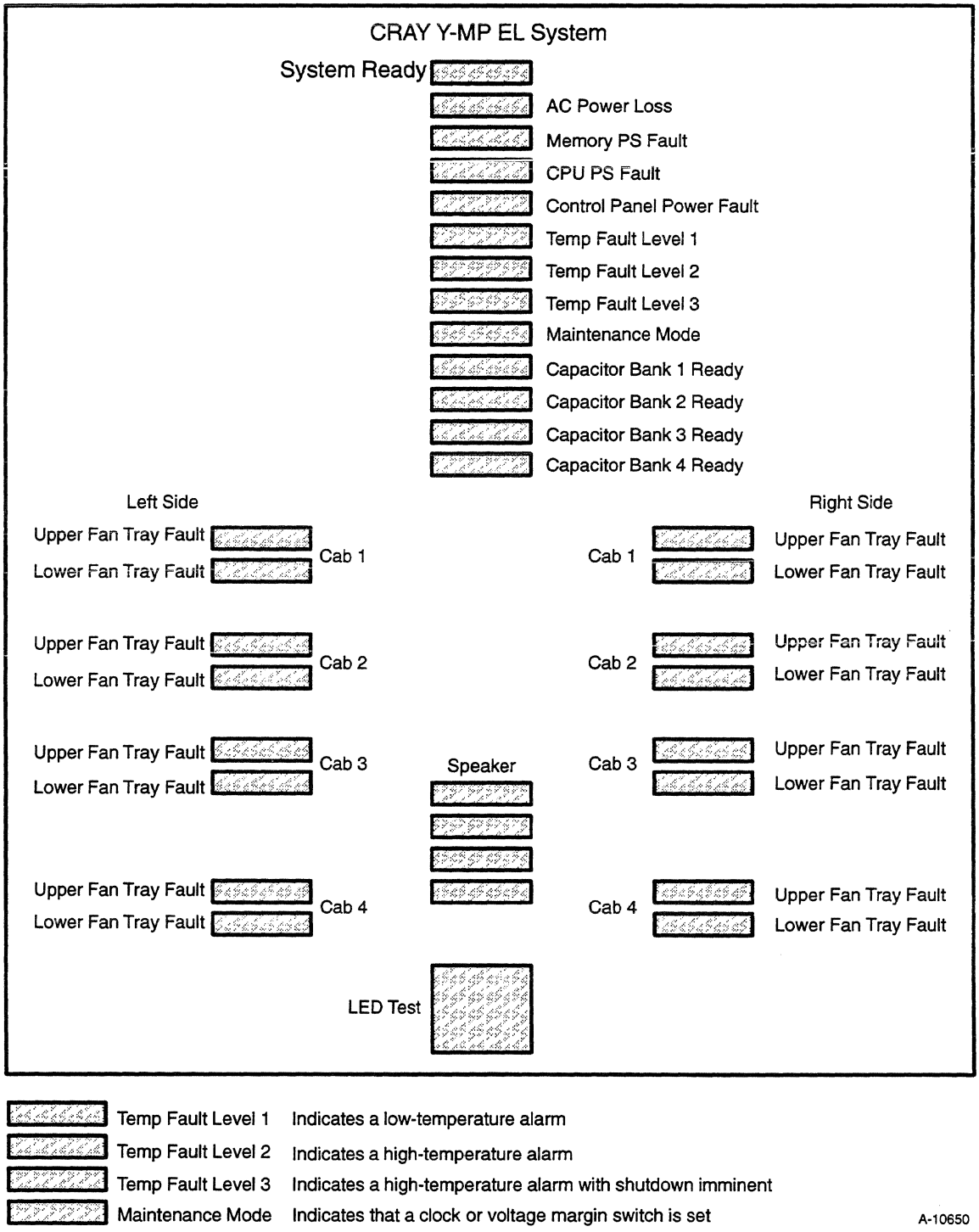


Figure 3. CRAY Y-MP EL control panel LEDs

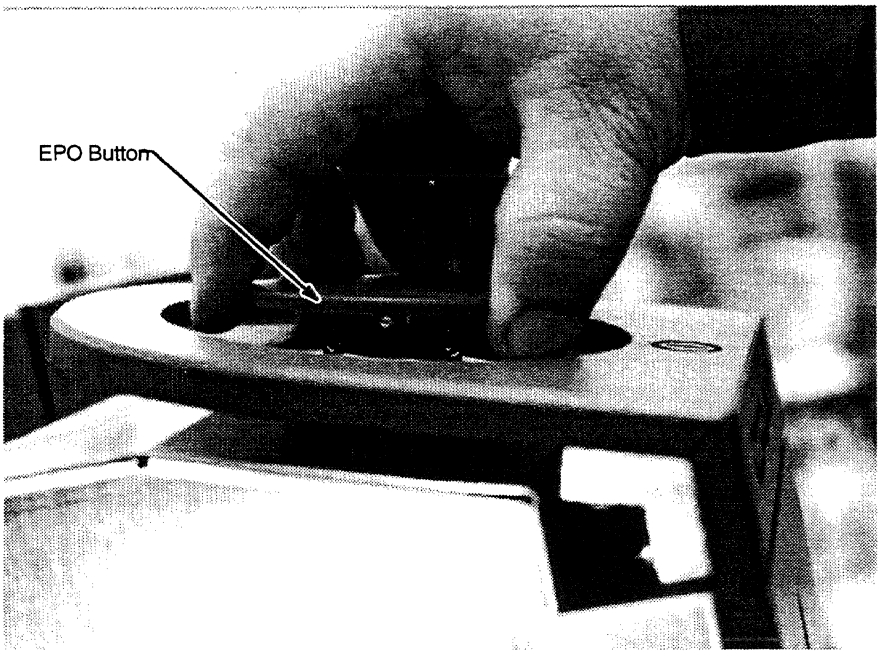


Figure 4. EPO button

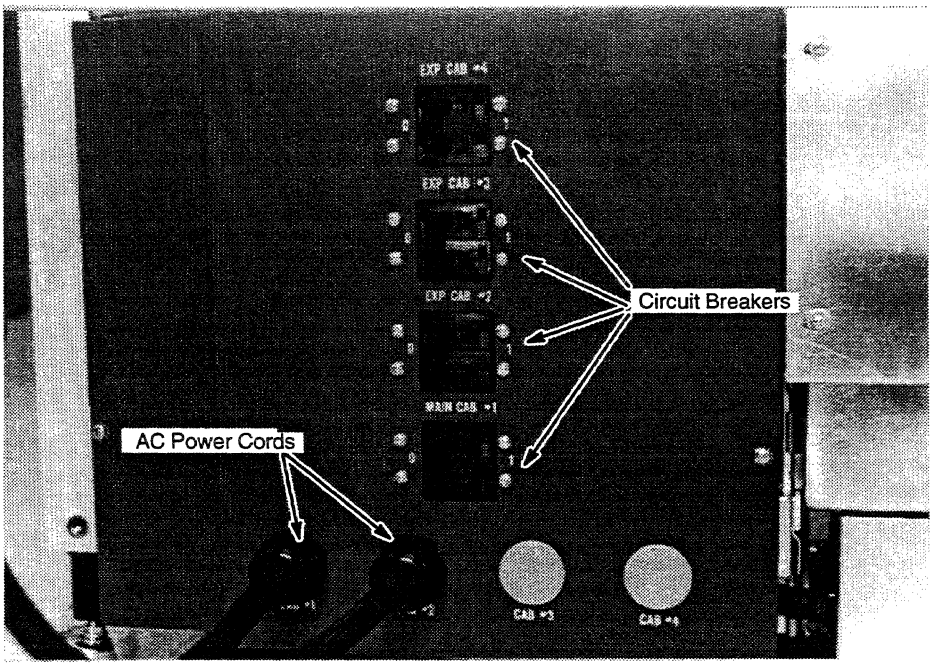


Figure 5. CRAY Y-MP EL and CRAY EL98 systems circuit breakers

Powering down a CRAY Y-MP EL or a CRAY EL98 system
I.1.2

To power down a CRAY Y-MP EL or CRAY EL98 system, follow these steps:

1. Ensure that the UNICOS system has been shut down properly before you start to power down the system. (See section 3, page 21.)

Caution

When you perform hardware maintenance on CRAY Y-MP EL or CRAY EL98 systems, press the EPO button when powering down the system (see Figure 4). This ensures that all safety features are working correctly.

---

2. Press the System Off button located behind the control panel door, as shown in Figure 6.
3. Ensure that the AC Power Loss LED is illuminated.

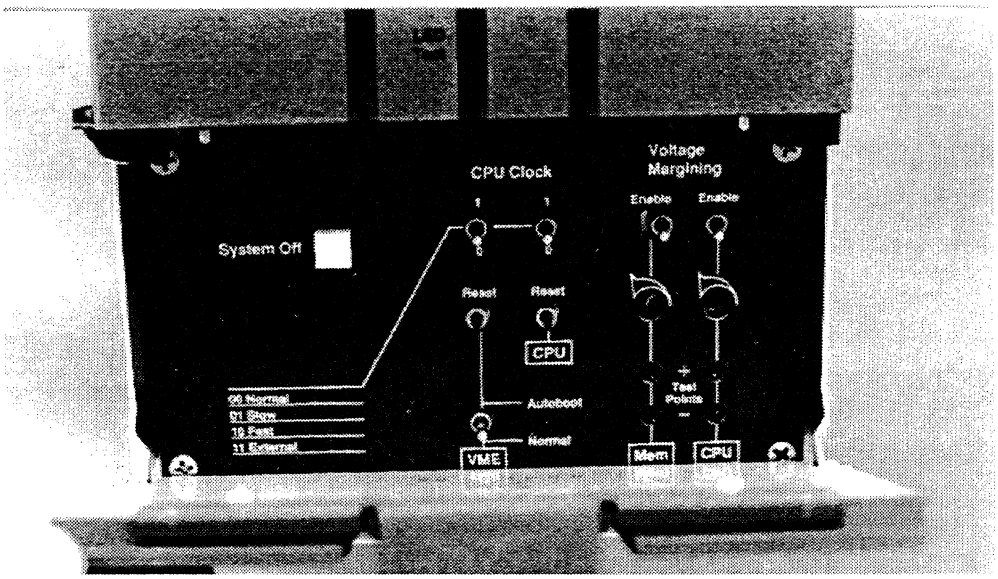


Figure 6. CRAY Y-MP EL and CRAY EL98 systems control panel door

Powering up/down a CRAY EL92 or CRAY EL94 system
15.9

This subsection contains procedures for powering up and powering down a CRAY EL92 or a CRAY EL94 system.

Powering up a CRAY EL92 or CRAY EL94 system
I.1.3

Perform the following power-up procedure at the circuit breaker panel, which is located at the front of the CRAY EL92 and CRAY EL94 systems.

1. Ensure that the AC power plug is connected to power.
2. Ensure that the Autoboot switch on the control panel is disabled (set to Normal) for the initial installation (see Figure 7). When the Autoboot switch is disabled (Normal), the IOS kernel is not automatically loaded, and automated confidence testing (ACT) is not invoked.

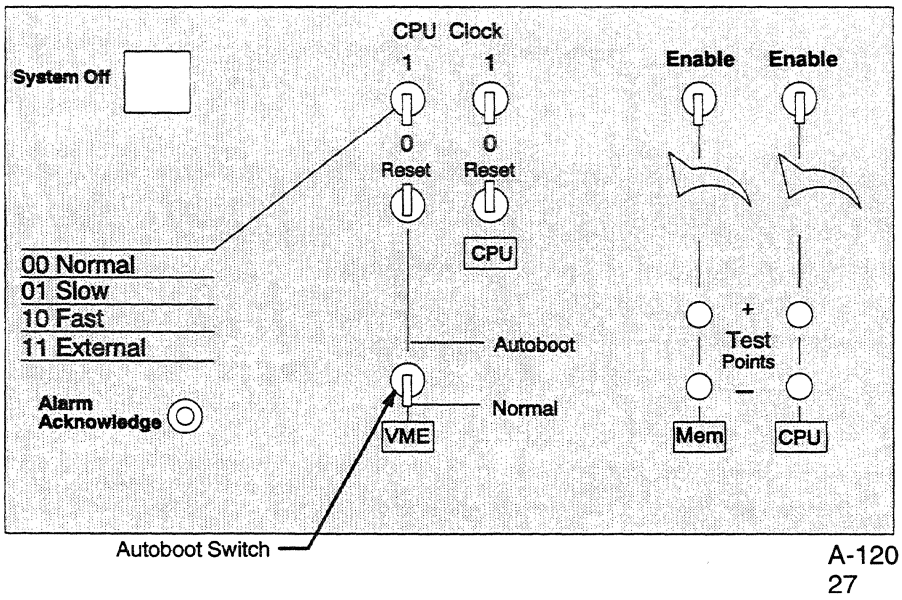


Figure 7. Autoboot switch

3. Turn on the system console.
4. Move the circuit breaker on the front of the system (see Figure 8) to the ON position (1).

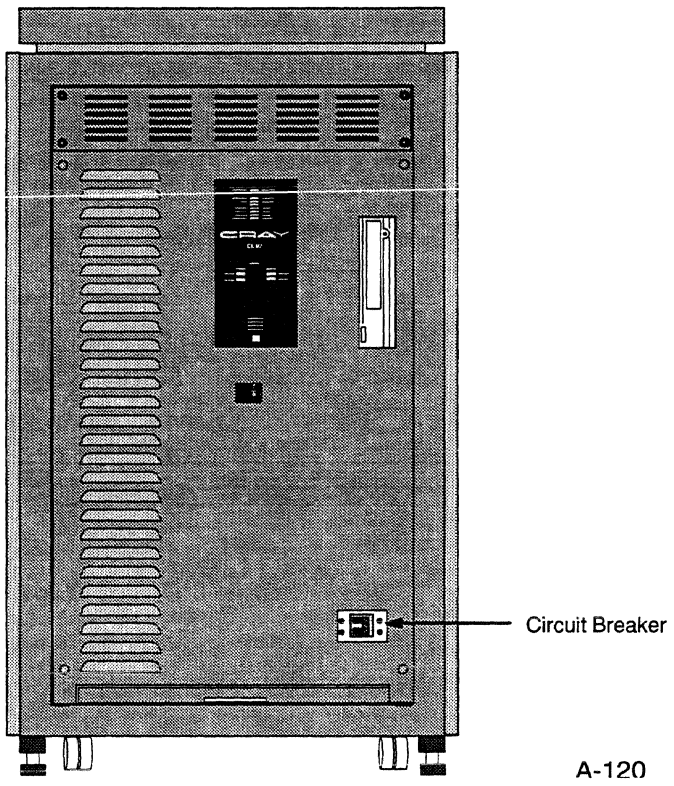


Figure 8. Circuit breaker

Powering on the system initiates a PROM-based self-test of the VME CPU. After the PROM self-test passes, the PROM code loads the `/diag/scripts/powerup` script. The `powerup` script invokes a basic checkout of the IOBB to verify that the software drivers can be loaded into the IOBB.

After the `powerup` script completes, the PROM code determines whether the Autoboot switch is enabled (set to Autoboot). If the Autoboot switch were enabled, the IOS kernel would be loaded, and ACT would be invoked automatically. ACT output would appear on the system console screen.

5. Ensure that the fans are operating.

6. Ensure that only the System Ready LED is illuminated on the control panel (after about 1 minute). See Figure 9. If any of the other LEDs are illuminated, contact Cray Research Hardware Product Support.

7. From the system console, enter `load` at the `BOOT>` prompt; doing so boots the IOS kernel and invokes ACT.

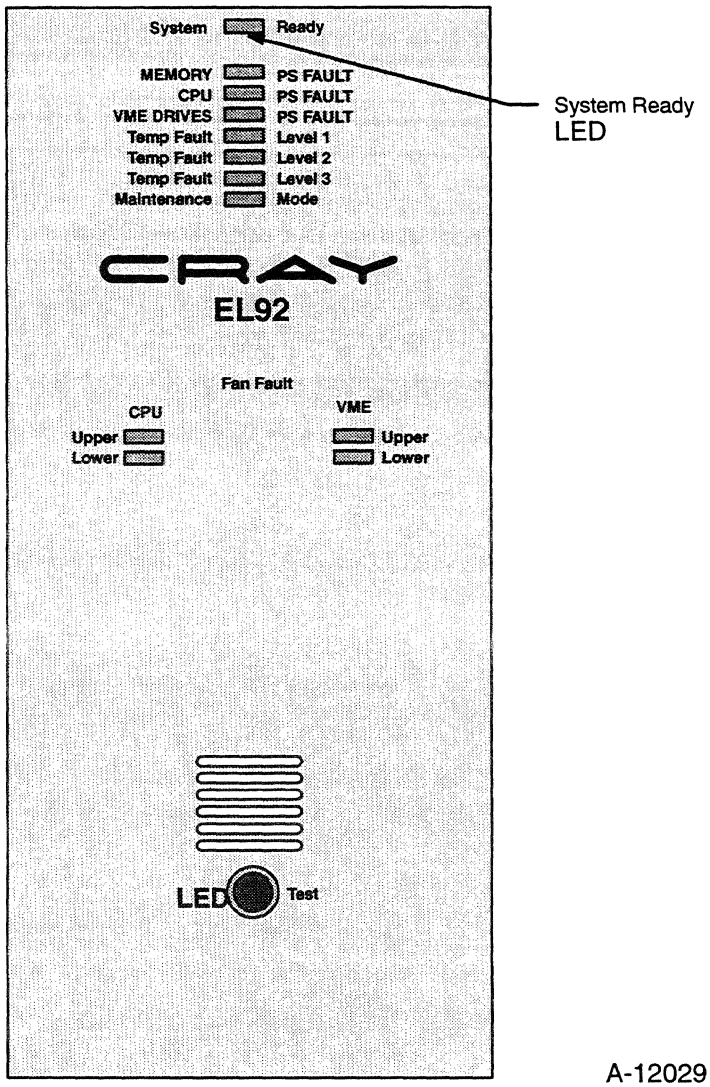


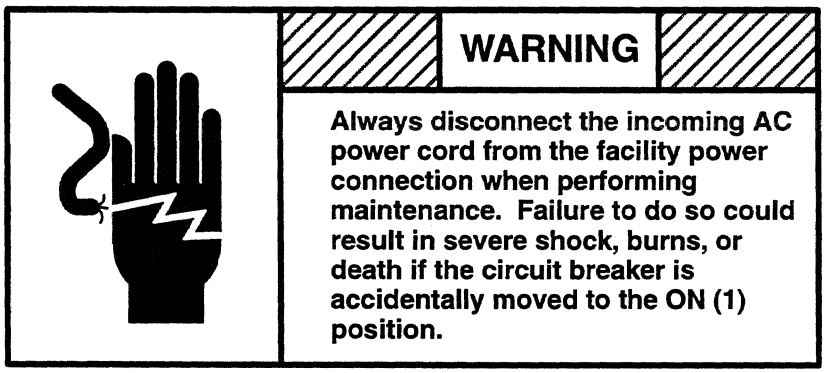
Figure 9. System Ready LED

8. Verify that ACT does not detect any hardware errors. If hardware errors are displayed, power down the system as described in subsection I.1.4, replace the defective component reported by ACT, and power up the system as described in subsection I.1.3. For troubleshooting information, see *CRAY EL Series Troubleshooting Using ACT*, publication HDM–060–0.

Powering down a CRAY EL92 or CRAY EL94 system I.1.4

To power down a CRAY EL92 or a CRAY EL94 system, follow these steps:

1. Ensure that all customer jobs are complete before you start to power down the system.
2. Ensure that you have permission to execute the necessary commands for this procedure.
3. While logged into the system, shut down the UNICOS operating system. (The time required to shut down the UNICOS system varies, depending on the system processes running.) Enter the `/etc/shutdown` command.
4. When the system is in single-user mode, enter `sync` three times.
5. Return to the `IOS>` prompt by pressing CONTROL-a.
6. Master clear and halt all CPU activity by entering the `mc` command.
7. Sync the disks and halt the IOS kernel by entering the `reset` command.
8. Wait until the `boot` prompt displays on the screen, and then move the circuit breaker on the front panel to the OFF (0) position (see Figure 8).



9. Unplug the incoming AC power cord from the facility power connection, **unless** you are removing or replacing the VME boards, CPU memory boards, disk drives, or the tape drive.

IOS-V Command Shell Overview [J]

The IOS-V command shell is an interactive shell based upon the common Korn shell functionality. The shell contains job control, command history, and the ability to execute scripts or simple ASCII files containing lists of commands.

The job control functionality supported by the IOS-V command shell includes the following, which are described in this appendix: CONTROL-C, CONTROL-Z, CONTROL-X, foregrounding, backgrounding, the `kill` command, and the shell history mechanism.

CONTROL-C functionality

J.1

Entering CONTROL-C causes the command shell to kill the current command that is running in the foreground, which is any command that is currently running and has not been explicitly placed in the background. A command can be placed in the background by placing the & symbol as the last character on the command line when the command is initiated, or by issuing a CONTROL-Z while the command is running and then entering the `bg` command at the IOS prompt. Entering CONTROL-C when no command is running in the foreground has no effect.

CONTROL-Z functionality

J.2

Entering CONTROL-Z causes the command shell to suspend the current command that is running in the foreground, immediately display the shell prompt, and wait for further console input. At this point, you can take one of the following three actions, which will affect the suspended command:

- Entering the `fg` command starts the suspended command in the foreground exactly as it had been running previously. For additional information about the foregrounding command, see the `fg`(8) man page.

- Entering the `bg` command starts the command running again but places it in the background, effectively allowing the interactive shell to run in parallel with it. This action is useful if you are doing something like formatting a disk and want to perform other actions while the `dformat` utility is running in the background. For additional information about the backgrounding command, see the `bg`(8) man page.
- Entering the `kill` command takes the command ID as an argument and attempts to kill the suspended or backgrounded command. For additional information about the `kill` command, see the `kill`(8) man page.

To identify the commands that are currently running in the background, enter the `jobs` command.

CONTROL-X functionality
J.3

Entering CONTROL-X restarts the interactive shell. You should enter CONTROL-X when a command that is running in the foreground is hung and entering CONTROL-C is ineffective in terminating the command. When the IOS you are logged into appears to be hung, enter CONTROL-C to ensure that the command shell is not making the IOS appear to be unresponsive.

Shell history
J.4

A shell history mechanism is supported in the same way that it is implemented in the Korn shell. Entering escape-K activates the history mechanism. You can navigate forward and backward through commands entered previously by typing the j or k keys, respectively. You can edit the command line by using standard `vi` commands and then re-execute the command.

Index

A

B

C

D

E

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Reader's Comment Form

UNICOS Basic Administration Guide for CRAY J90 and CRAY EL Series SG–2416 8.0.3.2

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name _____
Title _____
Company _____
Telephone _____
Today's date _____

Address _____
City _____
State/Country _____
Zip code _____
Electronic mail address _____

Cut along this line

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

ATTN: Software Information Services
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold