DataGeneral
Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Managing Mass Storage Devices and DG/UX™ File Systems

093–701136–00

AVIION®
PRODUCT LINE

Managing Mass Storage Devices and DG/UX™ File Systems

093–701136–00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085–series) and/or Update Notice (078–series) supplied with the software.

Printed in the United States of America
Rev. 00, July 1994

Ordering No. 093–701136

Notice

Managing Mass Storage Devices and DG/UX™ File Systems
093–701136–00

Revision History:

Original Release – July 1994

Effective with:

DG/UX 5.4 Release 3.10

Preface

This manual explains the operations necessary for configuring and managing your mass-storage devices — disk and tape drives. It is assumed that the device has already been installed and power has been applied. If connected to a newly installed DG/UX™ system, the devices are configured automatically in the kernel. If you add a device after the system is operational, you must first configure the device before it is usable. See Chapter 1 for information on navigating this manual.

Who should read this manual?

The manual is intended primarily for a system administrator or the person who is responsible for configuring the mass-storage device and maintaining its operation in a DG/UX environment. In a large configuration, either one person may configure and maintain all devices, or the user of the workstation to which the device is attached may be responsible for the device's operation. In general, if you build and boot a kernel for your workstation, you probably will be responsible for any attached devices.

Although this manual does not require expert knowledge of UNIX®, some system administration experience would be helpful. Familiarity with commonly used UNIX commands and the file system layout of the UNIX operating system is recommended. *Using the DG/UX™ System* provides general information on these topics should you need to review them.

Using this book

If you are adding a disk device to an existing DG/UX configuration (other devices are already operational), you may read through the chapters in this book in a sequential order. If you are already knowledgeable of the operations you must perform on a device, you may go immediately to the relevant chapter. A descriptive summary of the book's topics follows:

Making your mass storage devices operational Chapter 1

Directs you to the procedures necessary for making your devices operational.

Identifying and configuring mass storage devices Chapter 2

Surveys the hardware, describes DG/UX device naming syntax, and tells how to configure devices.

Tape devices	Chapter 3
Identifies the DG/UX dump and copy commands and gives instructions for removing a tape device from a DG/UX configuration. A list of tape drive documentation is provided in the Preface.	
Disk resource planning	Chapter 4
Explains how to design your file system layout and to map your software needs onto disk devices.	
Organizing virtual disks into mirrors and caches	Chapter 5
Offers two advanced methods for disk storage: mirroring and caching.	
Soft formatting and registering disk drives	Chapter 6
Gives such instructions as a prerequisite for creating virtual disks.	
Creating virtual disks	Chapter 7
Provides procedures to create virtual disks, software mirrors, and caches on physical disks.	
Creating and mounting file systems	Chapter 8
Explains how to create a DG/UX file system, create a mount point, mount the file system, and optionally export it.	
Using DG/UX file systems	Chapter 9
Explains typical uses for DG/UX file systems: creating subdirectories, text files, data files, third-party applications and executable files.	
Maintaining disk devices	Chapter 10
Covers four categories of disk maintenance: physical, virtual, mirrors, and caches.	
Maintaining file systems	Chapter 11
Covers local and remote file system maintenance.	
File system checking	Chapter 12
Explains how to repair corrupt file systems.	
Retrieving information about files and file systems	Chapter 13
Shows how to display disk usage, check for security breaches, and find files.	

DG/UX mass storage device names Appendix A

Gives full descriptions of DG/UX device names for all mass-storage devices.

Virtual disk management (VDM) Appendix B

Gives a brief overview of Virtual Disk Management and its predecessor Logical Disk Management.

Completing the disk planning worksheets Appendix C

Contains blank copies of the disk planning worksheets.

Troubleshooting Appendix D

Briefly lists errors reported for SCSI tape drives and NVRAM boards.

Introduction to interfaces Appendix E

Introduces **sysadm** (stand-alone and stand-among forms) and the shell.

Related Data General manuals

Within this manual, we refer to the following manuals:

Tape drives

Installing, Operating, and Maintaining the CLARiiON™ Tape-Array Storage System – DG/UX™ or AOS/VS II Environment Describes how to install, operate, and maintain the CLARiiON deskside tape-array storage system, and how to operate the rackmount tape-array storage system. Explains how to make the physical tapes accessible to the operating system. Describes how to add and replace customer replaceable units (CRUs). Ordering Number — 014-002181-02

Installing, Operating, and Maintaining the CLARiiON™ Tape-Array Storage System Model 793 Describes how to install, operate, and maintain the CLARiiON deskside tape-array storage system, and how to operate the rackmount tape-array storage system. Explains how to make the physical tapes accessible to the operating system. Describes how to add and replace customer replaceable units (CRUs). Ordering Number — 014-002359-00

Installing and Operating the Cartridge Tape Drive: Models 6675, 6676, 6677, and 6756 Describes how to unpack, configure, install, operate, and maintain the named models. Ordering Number — 014-001953-02

Installing and Operating Model 6760 and 6761 Cartridge Tape Drives Describes how to install and operate the model 6760 and 6761 8mm tape drives. Ordering Number — 014-002169-01

Installing and Operating the Model 6762 Digital Audio Tape (DAT) Drive Describes how to unpack, configure, install, and operate the drive. Explains how to clean the tape path and how to solve operating problems. Ordering Number — 014-002138-00

Installing and Operating the Model 6691 Stand-Alone Cartridge Tape Drive Describes how to install and operate the model 6691 stand-alone cartridge tape drive. Ordering Number — 014-002158-00

Installing and Operating Your 150-Megabyte 1/4-Inch Cartridge Tape Describes how to unpack, configure, install, operate, and maintain the 150-megabyte QIC (quarter-inch cartridge) half-height tape drive. Ordering Number — 014-001699-03

Installing the 6586/6587 Magnetic Tape Streamer Unit Describes how to unpack, inspect, install, and power up the 6586/6587 reel-to-reel tape drive. Explains how to install and remove reel-to-reel tapes. Ordering Number — 014-001692-01

Installing and Operating Your Model 6590 Series Cartridge Tape Drive Describes how to unpack, install, power up, and maintain the 2.2-gigabyte stand-alone cartridge tape drive. Ordering Number — 014-001701-01

Installing and Operating Reel-to-Reel Tape Drives: Models 6855D, 6855R, 6856D, and 6856R Describes how to unpack, inspect, install, power up, run diagnostics on, operate, and clean the unit. Explains how to install, remove, and care for reel-to-reel tapes. Ordering Number — 014-002216-00

Installing and Operating Reel-to-Reel Tape Drives: Models 6588, 6589, and 6857 Describes how to unpack, inspect, install, power up, run diagnostics on, operate, and clean the unit. Explains how to install, remove, and care for reel-to-reel tapes. Ordering Number — 014-001759-04

Operating the Model 61001 and 61002 1/2-Inch Cartridge Tape Drives with Autoloader Describes how to operate the Model 61001 and 61002 1/2-Inch cartridge tape drives and how to clean the tape path using the cleaning cartridge. Ordering Number — 014-002362-00

Disk drives

The CLARiiON® Series 2000 Disk-Array Storage System with the DG/UX™ Operating System Describes how to install, operate, and maintain the CLARiiON® deskside disk-array storage system and how to operate and maintain the rackmount disk-array storage system. The CLARiiON storage system has a capacity of 20 disk

modules. Explains how to use the **GridMgr** (grid manager) and **sysadm** programs to plan, set up, and manage storage-system disk configurations. Explains how to generate a new DG/UX kernel to support storage-system disks and how to transfer control of disks using DG/UX. Describes how to add and replace customer replaceable units (CRUs). (The original version of this disk-array system was named H.A.D.A. II and the original revision of this manual used that name.) Ordering Number — 014-002168-03

Installing and Operating Your Model 6538/6539 Half-Height Winchester Disk Drive Describes how to unpack, install, power up, maintain, and operate the 179-megabyte half-height Winchester disk drive. Ordering Number — 014-001722-00

Installing and Operating the Model 6562 and 6563 Diskette Drives Describes how to unpack, install, power up, maintain, and operate the 1.44-megabyte 3-1/2 inch and 1.2-megabyte, 5-1/4 inch half-height diskette drives (with SCSI adapter board). Ordering Number — 014-001921-02

Installing the 1-Gigabyte Fixed Disk Drive Describes how to unpack, configure, mount, cable, and apply power to the drive. Covers both single-ended and differential SCSI configurations. Contains detailed technical specifications for the drive characteristics, power requirements, environmental tolerances, and physical dimensions. Ordering Number — 014-002014-01

Installing Your Model 6554/6555 Series Disk Drive Describes how to unpack, install, power up, and maintain the 662-megabyte full-height Winchester disk drive. Ordering Number — 014-001702-01

Installing Model 6811-B Single-Ended and Model 6660 Series Disk Drives Describes how to unpack, install, power up, and maintain the 332-megabyte half-height Winchester disk drive. Ordering Number — 014-001940-01

Installing the 3-1/2 Inch Disk Drive: Models 6662, 6796 and 6799 Describes how to install models 6662 and 6796 (single-ended) and 6799 (differential) disk drives. Ordering Number — 014-002139-04

Technical Notice: Installing the 3-1/2 Inch Disk Drive Model 6662, Version B Provides additional configuration and installation information for model 6662 version B 332-megabyte disk drive. Ordering Number — 014-002165-00

Installing the 3-1/2 Inch Disk Drives: Models 61000-S, 61000-D, 6802 and 6805 Describes how to install models 61000-S. 61000-D, 6802 and 6805 disk drives. Ordering Number — 014-002239-01

Installing the 3-1/2 Inch Disk Drives: Models 6841, 6842, and 6843 Describes how to install models 6841, 6842, and 6843 disk drives. Ordering Number — 014-002194-00

Installing the Model 6861 and 61000-W 3-1/2 Inch Disk Drives Describes how to install models 6861 and 61000-W 3-1/2 inch disk drives. Ordering Number — 014-002288-00

Installing and Configuring the Model 6627 Rewriteable Magneto-Optical Disk Drive Describes how to unpack, configure, install, and operate the disk drive. Explains how to handle and use magneto-optical cartridges. Ordering Number — 014-001942-02

Installing and Operating Your Model 6552 Series CD-ROM Drive Describes how to unpack, install, power up, and maintain the 600-megabyte half-height CD-ROM drive. Ordering Number — 014-001721-01

Installing and Operating the Model 6629 CD-ROM Drive Describes how to unpack, install, power up, and maintain the model 6629 CD-ROM drive. Ordering Number — 014-002141-00

Installing and Operating the Model 6690 Stand-Alone CD-ROM Drive Describes how to unpack, install, power up, and maintain the model 6690 CD-ROM drive. Ordering Number — 014-002162-00

DG/UX system

Analyzing DG/UX™ System Performance Tells how to analyze DG/UX system performance and fine-tune a system. Explains how the DG/UX system uses the CPU, virtual memory, file systems, and I/O devices. Ordering Number — 093-701129-02

Installing the DG/UX™ System Describes how to install the DG/UX operating system on AViiON hardware. Ordering Number — 093-701087-06

Managing the DG/UX™ System Discusses the concepts of DG/UX system management. Explains how to customize and manage a system using commands and the **sysadm** system management tool. Includes instructions for booting and shutting down the system, backing up and restoring files and file systems, and recovering from system failure. Tells how to manage users, system services and activity, application software, and accounting. Ordering Number — 093-701088-05

Achieving High Availability on AViiON® Systems Describes the hardware and software components of high-availability systems. Provides instructions for managing high-availability

features of the DG/UX system. Explains how to set up AViiON systems in failover configurations, including examples of typical configurations. Ordering Number — 093-701133-01

Using the DG/UX™ System Describes the DG/UX system and its major features, including the C and Bourne shells, typical user commands, the file system, and communications facilities such as **mailx**. Ordering Number — 069-701035-02

Using the DG/UX™ Editors Describes the text editors **vi** and **ed**, the batch editor **sed**, and the command line editor **editread**. Ordering Number — 069-701036-01

Third-party software

Installing OpStar™ 2.00 Optical Storage Software Describes how to install the OpStar 2.00 optical storage software. OpStar 2.00 is optical imaging software that lets users on DG/UX systems access large amounts of data stored on optical disk volumes. Ordering Number — 093-000859–00

Using OpStar™ 2.00 Optical Storage Software Describes how to set up and use the OpStar 2.00 software after it has been installed. OpStar is optical imaging software that allows users on DG/UX systems to access large amounts of data stored on optical disk volumes. Ordering Number — 093-000858-00

Legato NetWorker User's Guide Explains how to use the NetWorker backup software. Shows how to mount backup tape volumes, request backups, browse through backed-up files, and restore them to disk. Ordering Number — 069-100496-01

Legato NetWorker Administrator's Guide Explains the setup and maintenance procedures for the NetWorker backup software. Tells how to set up the NetWorker server, its clients, and backup devices. Describes how to set up backup schedules, groups, levels, and policies, and how to prepare backup tapes, perform routine daily tasks, and recover from disk failures. Ordering Number — 069-100495-01

Legato NetWorker User's Guide Explains how to use the NetWorker backup software from a Networker client. Shows how to mount backup tape volumes, request backups, browse through backed-up files, and restore them to disk. Ordering Number — 069-100496-01

Reader, please note:

Throughout this manual the following format conventions are used:

command [*optional*] ...

Where	Means
command	You must enter the DG/UX shell command (or its accepted abbreviation) in the exact case specified.
[*optional*]	You have the option of entering this argument. Do not type the brackets; they only identify the argument as an option.
...	Means you can repeat the preceding argument as many times as needed.

Additionally, certain symbols in command lines are used.

Symbol	Means
↵	Press the New Line, Carriage Return (CR), or Enter key on your terminal keyboard.
<CTRL–D>	Hold the Control key down and press the D key on your terminal keyboard.
%, $ or #	The UNIX® shell prompt; the latter is reserved for the superuser.

The following example shows the **sysadm** menu traversal convention:

```
System -> Kernel -> Auto Configure
```

A typical prompt and user reply follow:

```
System configuration file name: [sport] canyon ↵
```

An example of a shell command follows:

```
# admdevice -o configure 'sd(insc(1),2,0)' ↵
```

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

Telephone assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Customer Support Center (CSC) by calling 1–800–DG–HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining our users group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive *FOCUS* monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1–800–253–3902 or 1–508–443–3330.

End of Preface

Contents

Chapter 8 – Creating and mounting file systems

Chapter 9 – Using DG/UX file systems

Chapter 10 – Maintaining disk devices

Chapter 11 – Maintaining file systems

Chapter 12 – File system checking

Chapter 13 – Retrieving information about files and file systems

Appendix A – DG/UX mass storage device names

Tables

Table

Figures

Figure

1 Making your mass storage devices operational

It is assumed that your mass storage devices are installed physically and that they are connected to an AViiON computer. Power has been applied to these devices and they have the potential to operate. If your DG/UX operating system has been installed, the attached devices are configured automatically in the kernel. If you have disk drives, as part of DG/UX installation, you have had the chance to software format them. Even so, there's still plenty to do to make the devices operational.

This manual presents sparse information on tape drives because the tape drive's function is limited primarily to storing file system backups. The book explains the DG/UX device-naming conventions for a tape device and gives procedures to configure and deconfigure it. Furthermore, it lists a set of DG/UX commands that initiate the backup activity and software applications that manage the backup. Related documents and manual pages are provided for additional assistance. For tape drives, focus your attention on Chapters 2 and 3.

Primarily focusing on the disk drives, this manual gives instructions to make disks operational. Since the disk drive family is so diverse — Winchester hard disk, CD-ROM, magneto optical, DOS-formatted diskettes and DG/UX file systems mounted from remote disks, for example — you must carefully read and select the operations that are relevant to your device and your application, if you have one. Each chapter covers a class of operations to help you not only use the device, but to improve its performance. For a Winchester hard disk, you might proceed through the manual in a sequential order, shown as follows:

Table 1–1 Using this manual to manage a Winchester hard disk

Operation	Chapter
Configure it	2
Plan its resources	3 and 4
Soft format and register it	5
Create virtual disks on it	7
Create a DG/UX file system on it and mount it at a directory location	8
Use the DG/UX file system	9
Maintain the physical disks, virtual disks, mirrors, caches and file systems	10 and 11
Check DG/UX file systems, if corrupted	12
Retrieve information about files and file systems	13

Although for a DOS-formatted diskette which is under the control of the MS-DOS® operating system, you need only create a DOS file system and mount it at a directory location. For a DOS-formatted diskette, consult Chapters 2 and 8.

In general, start with Chapter 1 and proceed through each chapter sequentially. If, after you have some familiarity with the procedures and need to perform a specific operation, use the Table of Contents, Chapter 1 and the Index to help you locate the desired topic.

Given the requirements of your device and the application that you may be using, you must select the operations that apply to you. For example, if you have a database application that requires disk storage, you must create virtual disks but not a file system. File system support is provided by the application. Or, if you are using a diskette device, you may not want to create virtual disks, but only a single DG/UX file system. Or, if you have the OpStar Optical Disk Jukebox Package which supports the 12-inch WORM jukebox and 5–1/4-inch rewritable and WORM jukebox subsystems, after you configure the devices in the kernel, you must go to a manual entitled *Using OpStar™ Optical Storage Software* for all disk and data management details. As a final example, if you have a CLARiiON disk-array storage system, after you configure the device, you must go to a separate manual entitled *Configuring and Managing a CLARiiON® Disk-Array Storage System* for remaining details.

Disk failover

Disk failover involves setting up two systems in a dual-ported configuration or a dual-initiator configuration where they share a common SCSI bus or disk array. Disk subsystems that support dual porting can also provide failover. In these configurations, the two systems provide alternate paths to the same devices and data. The disk-failover feature provides a simple way of transferring control of a disk or disk array from one system to the other. Disk failover thus provides not only a means of restoring database and application access quickly after a failure but also of balancing the I/O or CPU load shared by two normally functioning systems.

These topics are covered at length in the manual *Achieving High Availability on AViiON® Systems.*

End of Chapter

2 Identifying and configuring mass storage devices

A mass storage device is a hardware medium used to store data in the form of databases, work directories, tools packages, temporary file space, user home directories, software packages, extra swap space, and memory contents. Before you actually can use a mass storage device to do meaningful work, you must first perform a series of steps to make the device usable. This book aims to help you understand concepts that underlie the use of mass storage devices and it specifies the steps you must perform to make them operational.

This chapter, in particular, gives an overview of the types of mass storage devices you may be using, explains how you and the DG/UX system assign device names, and tells you how to configure the mass storage device into the DG/UX system so that it is recognized throughout the configuration.

IMPORTANT: Henceforth, all references to mass storage device simply will be device.

Surveying the hardware

This section describes the hardware attributes of a device.

Device controllers

A device controller, which can be integrated into the computer (AViiON server or workstation) or on a independent board in the computer, is the logic circuitry and firmware that connects the device to the computer. Over the years, many computer manufacturers have shifted from their own proprietary interfaces to third-party, industry-standard interfaces. At present, Data General supports three standardized controller interfaces: SCSI, ESDI and SMD.

SCSI and **SCSI–2**

Acronym for Small Computer Systems Interface, pronounced "scuzzy," SCSI is a standard interface to disk and tape controllers for small devices. Providing faster transfer speeds than the SCSI interface, SCSI-2 is an improved definition of the SCSI interface. Either resides either directly on the computer's CPU board or is an independent board in the computer. A SCSI interface supports up to seven devices attached to the computer in a serial configuration. A SCSI-2 interface (wide-SCSI 16-bit bus addressing) supports up to 15 devices. See Appendix A for more information on narrow and wide addressing.

The SCSI bus comes in two electrically different types: single-ended and differential. Single-ended SCSI has one wire per SCSI signal, which is driven to a positive voltage level to indicate logical 1 and 0 volts to indicate logical 0. Differential SCSI uses two wires per signal, and the wires are always driven to opposite polarities. The differential SCSI bus uses more wires but has better noise immunity and can be used over much longer distances than single-ended. A single-ended bus has a maximum cable length of 6 meters, while differential has a maximum length of 25 meters.

All single-ended devices can connect only to a single-ended bus; likewise, all differential devices can connect only to a differential bus. No intermixing is allowed.

Data General supports these types of SCSI interfaces:

cisc Ciprico VME SCSI adapter.

dgsc Data General VME SCSI–2 adapter. When used with a disk-array storage system, it is called a SCSI-2 adapter.

hada High-availability 30-module disk-array I/O processor.

insc Integrated SCSI adapter.

ncsc NCR integrated SCSI-2 adapter.

ESDI Acronym for Enhanced Small Device Interface, pronounced "ezdy," it can accommodate up to four disk-only devices (tapes excluded). Data General supports the **cied** (Ciprico VME ESDI controller) and **cird** (Ciprico VME ESDI or SMD controller).

SMD Abbreviation for Storage Module Disk, the SMD controller, which resides on an independent board, can accommodate up to four disk devices. Data General supports the **cimd** (Ciprico VME SMD controller) and **cird** (Ciprico VME ESDI or SMD controller).

Disk drives

Often called a "physical device," a disk drive is the most commonly used mass storage device. This family of devices is composed of two primary types: online (magnetic) and near time (optical). While

both magnetic and optical media satisfy requirements for storage and retrieval, magnetic devices offer speed and availability, whereas slower optical devices provide capacity.

Because new disk models are introduced with each release, supported device models are not listed here. Refer to the **sd**(7) manual page for an up-to-date list of supported disk drive models and raw capacities.

Fixed Winchester

Also called a hard disk, a sealed, nonremovable disk unit. Such a disk has multiple disk platters, stacked one above the other on a spindle. Data is stored magnetically in tracks on both sides of each platter. Each platter is divided into pie-shaped pieces, called sectors, each of which stores 512 bytes of data. When a disk reads or writes data, the disk transfers the data in sectors. The raw capacity of a hard disk ranges from 322 Mbytes to 4 Gbytes, depending on the model.

The most popular magnetic device, the Winchester offers the fastest access to data, with reasonable performance. A disk is a good choice for storing the operating system, users' home directories, source trees, database packages, and other applications.

Disk array

A group of Winchester disks grouped together on a dedicated controller, providing higher capacity, performance and reliability than a single disk. The disk-array storage system provides a compact, high-capacity source of disk backup for your computer system. It offers up to 40 Gbytes of disk storage, with as many as 20 disk drives. The function of the disk array is identical to the fixed Winchester drive.

CD–ROM

Acronym for Compact Disk–Read Only Memory, the raw capacity of a 5.25-inch CD-ROM is approximately 600 Mbytes.

CD-ROMs are the most popular permanent optical read-only storage medium for system software; multi-volume libraries such as encyclopedias, on-line documentation, and graphical images; video data; and audio data.

Diskette

Unit that reads and writes a removable magnetic medium. Inside the drive, the disk spins at high speed while a read/write head

reads and writes data. Diskettes, also known as floppies, come in standard sizes and capacities: 3.5-inch (720 Kbytes or 1.44 Mbytes) or 5.25-inch (720 Kbytes, 360 Kbytes, or 1.2 Mbytes). A diskette can be write-protected.

Diskettes are best used to transport readable and writable data between a personal computer and the DG/UX operating system.

Write Once, Ready Many Disk (WORM)

A WORM is a removable medium (similar to a CD-ROM) offered in two sizes and capacities: 5.25-inch 650 Mbyte or 12-inch 6.5 Gbyte. You can write data on a WORM only once. The medium can never be erased.

Having a larger capacity than magnetic media, WORMs are suited for applications that do not require fast, frequent access to the data. WORMs and CDs share a common purpose with the exception that WORMs are higher capacity devices and are writable.

Rewritable Magneto-Optical Disk (MO)

Also referred to as magneto-optical or optical disk, which uses a removable, rewritable 5.25-inch (900 Mbyte raw capacity) or 12-inch (2.4 Gbyte raw capacity) disk cartridge. Each cartridge has two sides that are identified by the letter A or B, which you can write protect by setting a switch.

Best suited for archival purposes because of its relatively slow transfer rate, an MO typically stores data that is accessed infrequently.

You may use the OpStar™ optical storage software to access the data stored on optical disk volumes for both magneto optical and WORM platters.

Tape drives

The tape device family comprises several tape drive types. Known as off-line devices, tapes drives are used primarily as archival devices.

Because new tape drive models often are introduced with each release, supported device models are not listed here. Refer to the **st**(7) manual page for an up-to-date list of supported tape drive models and raw capacities.

Helical scan tape

Helical scan tape capacities usually are expressed in "native" (uncompressed) capacities: 8 mm — 2 Gbytes or 5 Gbytes and

4 mm — 2 Gbytes or 4 Gbytes. A 4 mm tape is also known as a Digital Audio Tape (DAT). Using a tape in compressed mode effectively doubles storage capacity.

Many helical scan drives also implement a data compression function inside the drive, raising the effective capacity of a tape used in the drive. A data compression rate of 2:1 is typical for most operations, although your actual capacity may vary.

Tape array

A group of 4 mm helical scan tapes on a dedicated controller, a tape array subsystem provides greater capacity, performance and reliability than a single tape. Such a tape can still retrieve host data if a single tape drive or tape medium fails. The tape-array storage system provides a compact, high-capacity source of tape backup for your computer system or disk-array storage system. It offers up to 70 Gbytes of tape storage, with as many as seven tape drives.

QIC Cartridge

Unit that reads and writes removable, pre-formatted cartridge tapes as follows: QIC-24 (60 Mbytes), QIC-120 (125 Mbytes), QIC-150 (150 Mbytes), QIC-320 (320 Mbytes) and QIC-525 (525 Mbytes). Most cartridge tape devices read and write in a 512-byte fixed record size.

Reel–to–reel (9-Track)

The drives can use both 1.0 mil and 1.5 mil tape in reel sizes from 15.2 cm to 26.7 cm densities (6 in. to 10.5 in), and operate using any tape certified for 1600 cpi. The unformatted capacities of reel tapes are: 800 bpi — 20 Mbytes, 1600 bpi — 40 Mbytes, and 6250 bpi — 140 Mbytes.

Magnetic tape streamer unit

The unit can support tape reels from 6 inches to 10.5 inches, although it works best with 7-inch through 10.5-inch reels.

Tape autoloader

IBM 3480-compatible, 18-track, cartridge tape drives. Each drive contains a control panel, power supply, an integrated tape controller, and a SCSI interface board. You can change the drive from a single-ended to a differential SCSI interface by a selection from the front panel.

The drive comes with either a right-hand or left-hand mounted autoloader. The autoloader uses a magazine that holds from 1 to 10 IBM 3480-style, 1/2-inch cartridge tapes. The autoloader moves the magazine up or down to select, load, and unload each cartridge tape.

Each tape stores about 250 Mbytes of data; and each fully loaded magazine, about 2 Gbytes. With an optional ICRC data-compression board installed, storage capacities increase from 2 to 2.5 times, depending on the bit pattern of the stored data.

Memory

Although memory does not constitute a device, per se, it does supplement storage in the form of a special memory board or built-in random-access memory, both resident in the computer. Memory file systems provide the fastest access and least capacity of any of the disk or tape storage options.

Nonvolatile Random Access Memory (NVRAM)

Also known as Battery Backed Up Random Access Memory (BBURAM), its high speed and stability offer enhanced performance. The combination of nonvolatility through each NVRAM module's on-board lithium batteries and advanced management offered through the DG/UX system assures that data written to the NVRAM remains secure through a system power failure. Its high speed and stable storage provide a performance accelerator to network and database management system environment. Each NVRAM module contains 2 Mbytes of storage.

A common use of NVRAM is to serve as a front-end software buffer (or cache) to expedite file system access over the network. Use of NVRAM is not recommended for large databases whose write operations may overrun the cache, causing slowdowns.

Random access memory disk

Also called a RAM disk or a memory file system that resides entirely in memory. It provides very fast I/O performance for file systems that can fit in memory. Such a file system's volatility presents a major disadvantage; whenever you delete or unmount the file system or the system goes down, all contents of the memory file system are lost.

A typical use of the memory file system is for temporary space used by applications.

Device drivers, and standard and nonstandard devices

The DG/UX system communicates with a device through a device driver, which must be configured in the kernel. Whenever you configure a new device, its device driver code gets incorporated into your version of the DG/UX system. The SCSI disk driver, named **sd**, and the SCSI tape driver, named **st**, work with most industry-standard SCSI disk and tape devices.

Any device you purchased from Data General for use with AViiON computers is a standard device. Its device ID (SCSI ID) has been preset through a jumper or switch setting at the factory. (The documentation you used to physically install the device explained the importance of the SCSI ID). There are a limited set of standard device names, which are listed in **/usr/etc/probedevtab**. To find out the names of the standard devices, search this file for device names that begin with **sd** and **st**. A major advantage of having a standard device is that you can refer to it by its short name rather than its cumbersome internally represented long name. A later section, "Anatomy of a DG/UX device name," discusses device short and long naming conventions.

A nonstandard device is one that you add to your configuration for which there are no remaining standard device names. Or you might rejumper a standard device to a nonstandard setting. For example, you may add to your configuration a device controller supplied by a third-party vendor that requires a specific standard address setting. Should this setting duplicate the standard address of an existing device controller in the configuration, you must rejumper the existing device controller to a nonstandard address in long name format. To determine a nonstandard controller's address, follow the instructions outlined in *Programming in the DG/UX™ Kernel Environment*.

Finally, you may try to add nonsupported devices to a configuration. A nonsupported device is one you purchased from a third-party vendor. Such a device may be recognized by the DG/UX system, but there are no guarantees. Furthermore, such a device may not be compatible with the device driver provided by Data General. In this event, you will have to write a compatible device driver to accommodate the device. See *Programming in the DG/UX™ Kernel Environment* for details.

IMPORTANT: Be sure to keep track of the device IDs that you may have rejumpered in case you have any problems that may require help from Data General.

Why do I need to know a device's name?

You need to know a device's name under these circumstances:

- To specify the boot device for the DG/UX system and other bootable software.
- To configure any device.
- To recognize devices that are configured into the kernel and listed in the system file.
- To format a device.
- To identify a device that may be experiencing hardware failures.
- To deconfigure and physically remove a device from the configuration.
- To recognize the names of configured devices that are created in the **/dev** directory after the kernel is booted.

It is a good practice to know the names of devices that compose your DG/UX configuration. You may wish to affix an identifying label to each device.

There are two types of device names: DG/UX and node. You use the DG/UX name to refer to the device before the DG/UX system has booted; such as when you are configuring or formatting the device. Each time the kernel boots, the system automatically translates each DG/UX device name into a node name, which is listed in the **/dev** directory.

Anatomy of a DG/UX device name

A device's DG/UX name reflects the simplicity or complexity of your DG/UX configuration. The following example illustrates a simple DG/UX device name and a definition of each field in the name.

Short name

An example of a DG/UX short name follows:

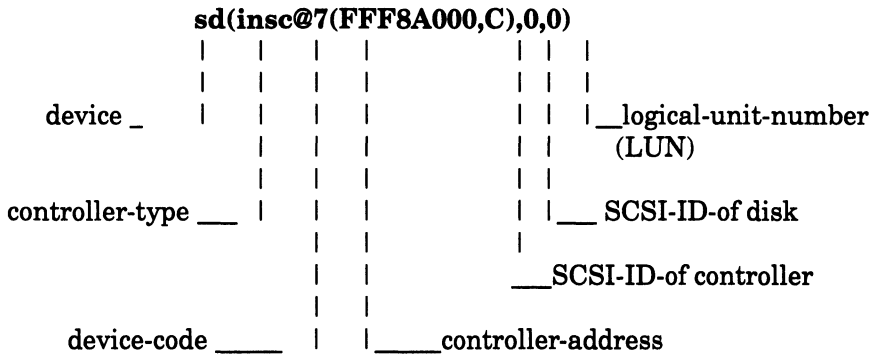
```
                 sd(insc(0),0,0)
                  |    |   |  | |___ logical unit number (LUN)
                  |    |   |  |      (not required for LUN 0)
                  |    |   |  |
      device ____ |    |   |  |__SCSI-ID number of disk drive
                       |   |
controller-type_____ |   |__controller-number
```

This device name **sd(insc(0),0,0)** identifies a disk drive on the first integrated SCSI controller (insc) which has controller number 0. The disk drive is set to SCSI ID 0. Since the disk is at LUN 0, the LUN is not required, although it is shown for clarity. You could identify the disk by the name **sd(insc( ),0)** or **sd(insc( ))**. Both the first controller number, the first SCSI ID number, and the logical unit number are 0.

Because this device is standard, you can refer to it by its short name. However, each short form is internally represented by a long form equivalent. Nonstandard devices and device nodes are represented exclusively in the long name form.

Long name

An example of a DG/UX long name follows:



The long name identifies a disk having the SCSI ID of 0 on the integrated SCSI controller whose device code is 7, controller address is FFF8A000 and controller SCSI-ID is C. Since the disk is at LUN 0, it needn't be further qualified by a logical unit number, although 0 is shown for clarity. The short name for this device can be expressed as **sd(insc( ))**, **sd(),0)** or **sd(0),0)**.

For a complete list of supported devices and their DG/UX long and short names, see Appendix A.

Device nodes

Device nodes are created automatically in the **/dev** directory each time you boot the kernel. Any facility that needs to access a device for an I/O activity (such as the DG/UX system or an application) will look in this directory for instructions. The node name supplies the device driver, which contains routines for performing I/O functions; the device type; and the device's unit drive, controller, or unit number. Device nodes come in two forms:

- Character mode (for character-at-a-time access)
- Block mode (for buffered access)

For more information on node names, see the **mknod**(1M) manual page. You should never need to use the **mknod** command to create nodes because all standard devices automatically are configured and have nodes created for them at boot time.

Disk drive nodes

Each disk drive is accessible in both block and character mode. Block nodes are in **/dev/pdsk**. Character nodes are in **/dev/rpdsk**. The following are example entries created in **/dev**:

/dev/pdsk/cied@18(FFFFEE00,0)	Block node for **cied(0,0)**
/dev/rpdsk/cied@18(FFFFEE00,0)	Character node for **cied(0,0)**
/dev/pdsk/sd(insc@7(FFF8A000,C),2,0)	Block node for **sd(insc(0),2,0)**
/dev/rpdsk/sd(insc@7(FFF8A000,C),2,0)	Character node for **sd(insc(0),2,0)**

Tape drive nodes

Tape drives are accessible only in character mode. The character-access nodes are in **/dev/rmt**; an example is **/dev/rmt/st(cisc@28(FFFFF300)4,0)**. All possible combinations of density and rewind options are created for each unit as follows:

/dev/rmt/*tape***@***dev-code*(*adapter*,*SCSI-ID*,*LUN*) [*density*] [*compress*] [**n**]

where:

density **l**, **m**, or **h** indicate whether you are using a low, medium, or high density storage medium. A node without a denoting letter is also created, which accesses the device's default density. For a drive that supports just one density, the system ignores any of these letters.

compress **c** or **u** mean use compression mode and use uncompressed mode, respectively. The **c** or **u** apply to a drive that supports compression mode (such as a DAT drive) only. For a drive that does not support compression, the system ignores any of these letters.

n indicates that you do not want the device to rewind after it closes.

For example,

/dev/rmt/st(cisc@28(FFFFF300),4,0)lu

This is a SCSI tape with a Ciprico SCSI adapter with default device code and address. The low-density, uncompressed tape is at the default SCSI ID of 4.

Virtual disk nodes

If you create virtual disks on devices, node names will be created for them also in **/dev/dsk** and **/dev/rpdsk**. More information on virtual disks is given in Chapter 4.

You may access virtual disks in both block and character mode. Therefore, the kernel creates both types of nodes. Block nodes are in **/dev/dsk** and character nodes are in **/dev/rpdsk**. When you use **sysadm** to create a virtual disk, a corresponding virtual disk node is created. If you create a virtual disk named **comm**, the nodes in **/dev** will be

/dev/dsk/comm	**/dev/dsk(comm,2D627F27,0C0208CB,0)**
/dev/rdsk/comm	**/dev/rdsk(comm,2D627F27,0C0208CB,0)**

Configuring a device

All devices must be configured in the kernel. Before you configure a device, check the current list of configured devices in the kernel, in case the device is already configured.

As superuser, enter the **sysdef** command. Typical output follows:

```
# sysdef ↵
# Configured devices
#
kbd()
grfx()
lp()
duart(0)
inen()
wdt()
sd(insc(),0)
st(insc(),4)
st(insc(1),*)
...

# Device drivers
#
sd
st
insc
duart
grfx
kbd
lp

...
```

The list includes one SCSI disk drive with a SCSI ID of 0 and two SCSI tape drives. The first tape drive, attached to the first controller (0), has a SCSI ID of 4. The second tape drive in the list includes a SCSI tape drive that is characterized by the asterisk wildcard. The asterisk (*) is used as a wildcard to match all device SCSI IDs (0 through 15) — assuming wide SCSI addressing is supported — on the second (1) controller. Therefore, if you add another tape drive to this controller, the device is already recognized by way of the wildcard; you do not need to configure it explicitly.

CAUTION: *If you have a failover configuration do not use the asterisk wildcard for device names in your system file.*

You must configure explicitly a device under these circumstances:

- You add one or more devices to an existing controller whose device name is not represented with the asterisk wildcard character.

  For example, if you add a second device to the second (1) integrated SCSI controller and you do use wildcards in the system file, you must configure the device.

- You add a nonstandard device to your configuration.

  You must always configure nonstandard devices.

  All device controllers are built into the kernel by default.

You may also scan the list of configured devices in the system file (from which the kernel is extracted) at this location: **/usr/src/uts/aviion/Build**. Scan your system files using the **more**, **cat**, **grep**, or **view** commands.

You may configure a device in either of two ways: dynamically configure the device while the system is operating or rebuild the kernel and take the system down to reboot it. The primary advantage of dynamic configuration is that it does not disrupt system operation. It is also a desirable method for adding a device temporarily. The disadvantage of a dynamic configuration is that if you reboot or if the system crashes, you lose the device configuration. To permanently configure a device, however, eventually you must rebuild the kernel, incorporating the new device, and reboot the system. Also, you must rebuild the kernel when configuring nonstandard devices and device drivers.

The device's device driver must also be configured in the kernel. Whenever you configure the device itself, the device's driver is automatically configured as well. Refer to the section on device drivers in this chapter for details.

Procedures for a dynamic device configuration and a kernel build follow.

Dynamically configuring a device

To dynamically configure a device while the system is operating, use the **sysadm** operation:

```
Device -> Configure
```

The system prompts for the name of the device to configure.

Alternatively, you may use the shell command:

admdevice -o configure *'device-name'*

Configuring a device puts an entry for it in either **/dev/pdsk**, **/dev/rpdsk** or **rmt** (tape nodes), whichever is appropriate.

The configure operation does not update your system file with the added device's name. Eventually, you must rebuild the kernel to include the added device in the system file to ensure that it gets recognized at boot time.

Each device driver is configured automatically in the kernel; you needn't be concerned about its explicit configuration.

Rebuilding the kernel

You may rebuild the kernel with **sysadm** using either an automatic configuration or a custom build. Both methods automatically

configure all standard devices that are physically connected to your configuration.

The auto configure option builds a kernel that includes default variable values and requires no user interaction. It builds the kernel and links it to **/dgux** automatically. You may choose this option for a computer system that:

- Does not share a disk-array storage system with another host
- Does not connect to a nonstandard device

The custom build option lets you can edit the system file, producing a custom kernel.

In both cases, you must reboot the system, activating the new kernel.

Building an auto-configured kernel

To build an auto-configured kernel, follow these steps though **sysadm**.

```
System -> Kernel -> Auto Configure
```

A typical dialog follows:

```
System configuration file name:[sport] ↵
Overwrite existing [system.sport] [yes]: ↵
[system.sport]  Correct? [yes] ↵
Operating system client? [no]  ↵
Okay to perform operation? [yes] ↵
```

After you confirm, **sysadm** displays this message:

```
Warning: Only devices with drivers configured in the
currently-running kernel will have drivers in the new
kernel.
```

This means that the new kernel can support only those devices whose controllers are recognized by the current kernel.

Then **sysadm** builds the new kernel. The process takes several minutes. These messages appear:

```
Configuring system...
Building kernel...
Successfully built dgux.sport
Linked /dgux.  You must reboot in order for this
kernel to take effect.
```

You have successfully built an auto-configured kernel. The kernel file resides in **/dgux.sport**. The system configuration file resides in **/usr/src/uts/aviion/Build/system.sport**.

The new kernel is not effective until you boot it. Follow these steps through **sysadm** to boot:

```
System -> Kernel -> Boot
```

Building a custom kernel

To build a custom kernel, follow these steps though **sysadm**:

```
System -> Kernel -> Build
```

Answer these prompts:

```
System configuration file name:[sport] ↵
[system.sport]  Correct? [yes] ↵
Operating system client? [no] ↵
Editor: [/usr/bin/vi] ↵
```

The system file is several screens full. Figure 2–1 shows an excerpt from a system file.

```
# Automatically Configured Hardware Devices:
#
# These hardware devices were found on the system by probedev(1M).
#
    vme()           ## VME bus (number 0)
    kbd()           ## Workstation keyboard
    grfx()          ## Workstation graphics display
    lp()            ## Integrated parallel line printer controller
    duart(0)        ## Dual-line terminal controller (number 0)
    duart(1)        ## Dual-line terminal controller (number 1)
    syac(vme(0),0)  ## Systech terminal controller 0 on VME channel 0
    dgen(0)         ## Second-Generation Integrated Ethernet controller 0
    wdt()           ## Watchdog Timer
    sd(ncsc(0),0)   ## SCSI disk 0 on Second-Generation SCSI adapter 0
    sd(ncsc(0),1)   ## SCSI disk 1 on Second-Generation SCSI adapter 0
    sd(ncsc(0),3)   ## SCSI disk 3 on Second-Generation SCSI adapter 0
    st(ncsc(0),4)   ## SCSI tape 4 on Second-Generation SCSI adapter 0
    st(ncsc(0),5)   ## SCSI tape 5 on Second-Generation SCSI adapter 0
    st(ncsc(0),6)   ## SCSI tape 6 on Second-Generation SCSI adapter 0
```

Figure 2–1 Hardware devices excerpt from system file

To configure a standard device for an OS server or a stand-alone workstation, see "Automatically Configured Hardware Devices." For

a standard device attached to an OS client, see "Typical AViiON OS client device configuration."

As an example, to configure a nonstandard device on a new Ciprico SCSI adapter, you might hand edit the system file as follows:

```
# Hand-entered Nonstandard Devices

cird@28(FFFFF300,3)     ## Nonstandard Ciprico SCSI adapter at
                       ## controller address FFFFF300
```

Be sure to document your hand-edited entries using comment symbols (#). You may enter the nonstandard device anywhere in the file; however, logically, it is best located near the automatically configured hardware devices.

After you finish editing the system file, save the file and exit from the editor.

The following sample dialog completes the kernel building process.

```
Link the new kernel to /dgux? [yes] ↵
Save the old kernel? [yes] n ↵
Continue with the build?  [yes] ↵
Configuring system...
Building kernel...
Successfully built dgux.sport
```

Assuming that you want to link the new kernel to **/dgux**, **sysadm** also displays

```
Linked /dgux.  You must reboot in order for this
    kernel to take effect.
```

Now you must boot the kernel you just built.

Booting the kernel

To boot the kernel just built, follow these steps though **sysadm**.

```
System -> Kernel -> Boot
```

Sysadm displays the boot path. A typical dialog follows:

```
Boot path: [sd(insc(0),0,0)root:/dgux -3]↵
All currently running processes will be killed.
Are you sure you want to reboot the system? [yes] ↵
```

The kernel you specified boots automatically to a run level of 3. The screen displays a series of messages whose content depends on the run level to which you are booting and the particular packages you have set up. The time it takes to complete the booting process depends on a number of conditions. Eventually, a login prompt appears. A sample display looks like this:

```
Booting sd(ncsc(0),0,0) root:/dgux -3
DG/UX Bootstrap 5.4R3.10
Loading image ..........................

sport
Console login:
```

For more information on kernel building, see *Managing the DG/UX™ System*.

End of Chapter

3 Tape devices

After you have physically installed a tape device and configured it into the DG/UX system, you are ready to use it. How you use a tape device depends on its basic capabilities and the application you use, if any.

This brief chapter introduces the DG/UX commands used for writing data to tape. For additional information on using tapes as backup devices, see *Managing the DG/UX™ System*. Also, refer to the Preface for a list and description of documents about tape and disk drives, software archiving products and other tape software products.

DG/UX tape dump and copy commands

Tape devices are used typically for storage of dumps or backups. The following DG/UX commands are useful for archiving to tape:

tar	Saves and restores files on magnetic tape.
cpio	Copies file archives in and out.
dump	Performs an incremental file system dump; copies to magnetic tape all files changed after a certain date in a particular file system.
dump2	Same as **dump**, but faster. Performs an incremental file system dump; copies to magnetic tape all files changed after a certain date in a particular file system.
dd	Copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block size may be specified to take advantage of raw physical I/O.
restore	Reads files and symbolic links dumped with the **dump** or **dump2** commands.

See the respective manual pages for more information.

Using NetWorker to back up disks to tape

Legato NetWorker® for AViiON Computers, which is bundled with the DG/UX system, is a disk backup application that allows you to

back up NetWorker clients' disks to the NetWorker server's tape drives. You have the option to install NetWorker along with other DG/UX bundled software products when you install the DG/UX system. You must configure NetWorker to behave in the desired way. See the NetWorker documentation and *Installing the DG/UX™ System*.

Operating the tape autoloader

You use the **mt** and **rmt** commands to change media for the Model 61001 and 61002 1/2-cartridge tape drives with autoloader (see the *Operating* manual). See the **rm**(7) and **rmt**(7) manual pages for details on these commands.

You may use the DG/UX tape dump and copy commands and the Legato NetWorker® product to operate on each tape device.

The primary use of the tape drive with autoloader is expected to be data interchange with IBM systems. The DG/UX system does not offer a utility that translates or exchanges this data. If you require this feature, a REELexchange product is available that incorporates data conversion functions and automatic stacker operation. This product is available from Software Clearing House (SCH) at (513) 579–0455 (USA).

IMPORTANT: Do not use the REELexchange product supplied with the DG/UX system to operate this product. The REELexchange product was designed for use with nine-track tape devices and applications.

Removing a tape drive from the configuration

Removing a tape drive from a configuration (or deconfiguring) removes its entries from **/dev/rmt** (tape nodes).

Select the following **sysadm** operation to remove the tape drive from the configuration.

```
Device -> Deconfigure
```

A successful deconfiguration is confirmed.

Alternatively, you may choose to use the `command:`

admdevice -o deconfigure *'device-name'*

The deconfiguration operation does not update your system file; the tape drive you just deconfigured still remains. After you physically

remove the device from the configuration, you must rebuild the kernel to delete the deconfigured device. Otherwise, at system boot, the operating system, consulting its system file, would attempt to configure a nonexistent device because it is still listed in the system file. Using the autoconfigure option, the system automatically configures only attached devices. Since the deconfigured device is no longer attached, it will not be autoconfigured.

IMPORTANT: If you used the asterisk (*) notation in the device name in the system file, you do not have to delete it from the system file and rebuild the kernel. Refer to Chapter 2 for information on building an autconfigured kernel.

End of Chapter

4 Disk resource planning

To use your devices most effectively, you must decide how to organize your data. Proper planning now can prevent time-consuming reorganization later.

You have two primary objectives:

- to design your file system layout, and
- to map your software needs onto devices.

This chapter attempts to help you design your DG/UX system and presents a strategy for creating it.

IMPORTANT: For any physical disks in a disk-array storage system that you intend to use for failover see the 014-series manual supplied with the storage system.

Organizing your DG/UX file system

Ultimately, you want to organize your data and work areas in a hierarchical structure called a file system. The term *file system* has two meanings: (1) the entire tree comprising many hierarchical levels and (2) a component of the tree that contains its own directory structure. Figure 4–1 shows a simple example of a DG/UX file system.

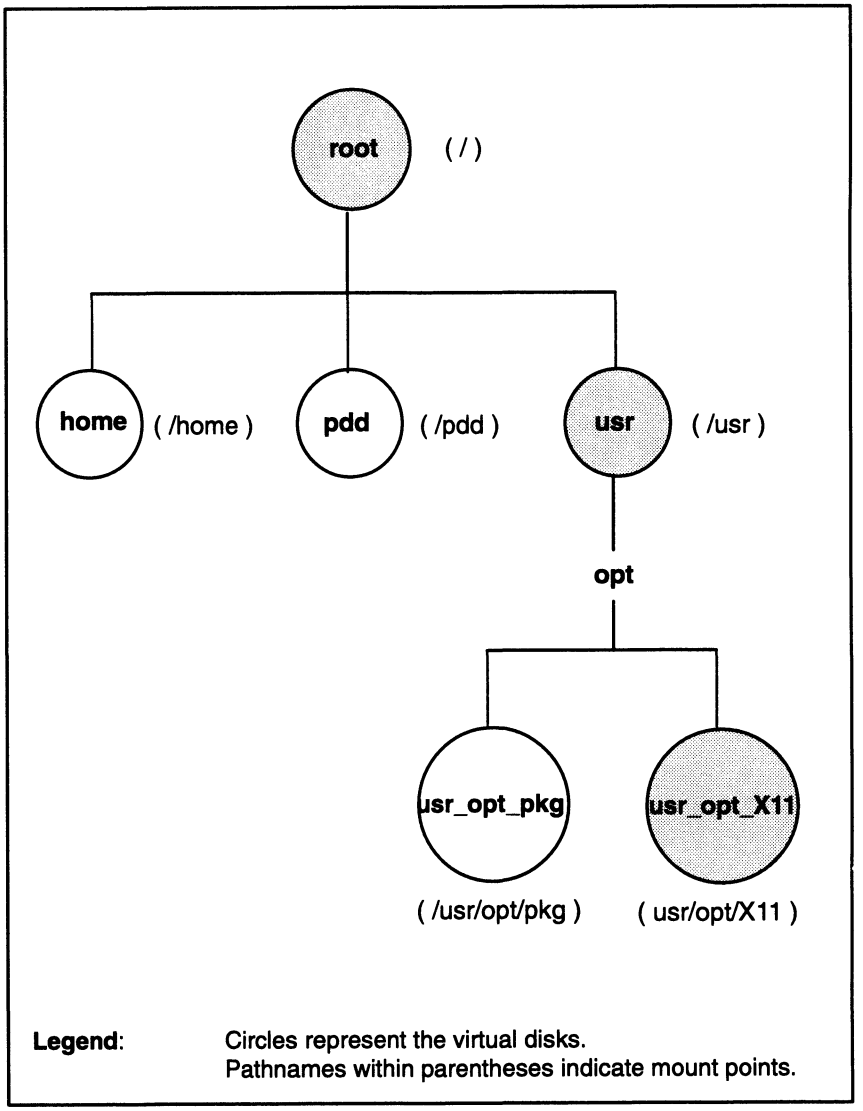


Figure 4–1 Simple DG/UX file system structure

The entire structure is referred to as the DG/UX file system, and each circle component is also a file system. The shaded circles represent file systems that compose the DG/UX operating system: **root**, **usr**, and **usr_opt_X11**. The remaining file systems are examples of those you may wish to include in your DG/UX system. Accordingly, each component of the DG/UX file system will contain its own set of file systems. If your DG/UX system is operational, you can list the files residing in the root file system (represented as /) by typing these commands:

```
% cd / ↵
% ls -CF ↵
admin/      dgux.moe.old*   opt/     tftpboot/
bin@        dgux.installer* pdd/     tmp/
dev/        etc/            proc/    udd/
dglib/      interleaf/      sbin/    usr/
dgux*       lib@            spool/   var/
dgux.moe*   local/          srv/
```

The root file system contains the executable file for the kernel, **dgux**, and other critical directories: **/dev** for device files, **/etc** for important system files and maintenance utilities, and **/tmp** for storing temporary files, to name a few.

A typical file system you likely will create is **home**, which would house your personal files and users' home directories. You might also create a file system for a software application that is critical for your users. Such an application is represented as **usr_opt_pkg**.

As you can see, the DG/UX file system looks like an inverted tree, starting at the directory / and continuing downward to other levels. To specify the location of a particular file system, you must specify its pathname. For example, the **home** file system is located at **/home** and the **usr_opt_pkg** file system is located at **/usr/opt/pkg**.

After you create a file system, you must graft it onto the DG/UX tree by mounting it. A file system's mounted location is referred to as a *mount point*.

File systems that you create on your devices are considered *local file systems*. However, you are not limited to using just the file systems that you create and those related to the operating system such as / and **/usr**. In addition, you may access and use file systems that already exist on other AViiON computer systems attached to a network. File systems on remote systems connected to a network are referred to as *remote file systems*. You must also mount the remote file systems onto your DG/UX file system to enable your access to them.

Planning worksheets

The planning worksheets contain tables that you complete to organize your file system needs. You will identify the names of tape devices, virtual disks, sizes, possible types, and mount points in the DG/UX directory structure. Time you spend here will speed up the resource allocation process. The planning worksheets are shown in Figures 4–5 through 4–7.

Organizing file systems on virtual disks

A file system is based on a higher level abstraction called a virtual disk. A virtual disk may be thought of as a container of disk space over which a file system structure is imposed. In Figure 4–1, the circles are actually virtual disks on which file systems reside. The file system on the virtual disk is what's mounted onto the DG/UX file system. For example, the virtual disk **root** containing a file system is mounted at /. As another example, virtual disk **usr_opt_pkg** is mounted at **/usr/opt/pkg**. DG/UX virtual disk's names, by convention, suggests their mount point. Most file systems you create will require virtual disks.

See Appendix B for information about the on-line management of virtual disks.

Naming virtual disks

Virtual disk names may contain 31 characters, including alphabetic characters, numbers, period (.), hyphen (-), and underscore (_). However, some characters are invalid:

Table 4–1 Invalid characters in virtual disk names

Character	Description
	space
"	double quotation mark
'	single quotation mark
( )	left and right parentheses
,	comma
/	slash
:	colon
@	at sign
\000 through \037, \200 through \237 \177	ASCII control characters

IMPORTANT: Avoid using a . (dot) as the first character in a file name, which may cause confusion with other file names beginning with a dot, which signifies a system startup file or a partition file.

You may want to adopt a naming scheme that identifies the mount point of the file system that will be created on the virtual disk. Or,

select a name that is memorable, one that describes its purpose. Especially for a series of related virtual disks, adopting a common naming scheme can be helpful. For example, a virtual disk named **tax_86** might contain tax records for 1986.

Determining your virtual disk needs

Typical virtual disks that you may need for your data storage and work areas are provided in the following list.

- User home directories
- Third-party and database packages
- Work directories
- Tools directories
- Temporary space
- Front-end cache

The virtual disks you create are not restricted to those listed here. You also may wish to create virtual disks for OS clients and secondary releases of an operating system. Refer to *Managing the DG/UX™ System* for information on disk planning for these purposes.

When computing virtual disk sizes, you must account for file system overhead.

Factoring in file system overhead

To accommodate a file system that is readable and writable, add 10 percent to the virtual disk's size as overhead. For a read-only virtual disk, add nothing for overhead.

For example, the release notice for a software package recommends 100 Mbytes of space. To account for 10 percent overhead, you would calculate vertical disk size in disk blocks as follows:

100 + 10% *overhead = virtual disk (readable and writable) size*

100 + (100 * .1) = 110 Mbytes = 239,616 blocks

Blocks = (Mbytes * 1,048,576) / 512

Some virtual disks (such as for a database application requiring a raw disk) do not require a file system, in which case there is no overhead. Read your software application's release notice for details.

User home directories (home)

A home directory is useful for containing each user's work on the system. It also contains files that customize each user's shell,

electronic mail environment, and X Window environment (for example, through the **.login** or **.profile**, **.mailrc**, and **.Xdefaults** setup files). Usually one file system is needed to accommodate all users' home directories.

A user's home directory requires a variable amount of space depending on the work the user does and the files the user accumulates. As an example, suppose you determine that each user's home directory needs 40,000 blocks in which to save mail, write memos, collect product specifications, and accommodate various temporary files that the system or other programs produce, such as scratch files. For five users, you could calculate home directory space as follows.

*(number-of-users * blocks-per-user) + 10% overhead*

```
(5 * 40,000) + 10% = 200,000 + 10% = 220,000 blocks
```

You must account for any OS clients' home directories as well as directories for local users. *Installing the DG/UX™ System* explains adding OS clients; *Managing the DG/UX™ System* explains adding user accounts.

Third-party and database packages (usr_opt_pkg)

You can create a separate virtual disk for each third-party package, or you can create a single virtual disk for all third-party packages. Database packages are singled out because, in many cases, the third-party package (such as a retail accounting system) may be built upon an industry-standard database package. So, in some instances, both the database and the third-party package have virtual disk requirements.

If you put all third-party packages on one virtual disk, it must be sufficiently large to accommodate the sum of the individual packages. Planning for such a virtual disk may be difficult to forecast. To save disk space, consider making a virtual disk for each package so that you use only the disk space required per package. The release notice that accompanies the third-party package specifies its size requirements. Without release notice instructions, you may calculate size requirements using these guidelines. When calculating the size of a virtual disk for a third-party package whose file system will be readable and writable, add 10 percent of the package size for file system overhead requirements. For a read-only file system, add nothing for overhead.

Normally, you should mount virtual disks for third-party packages below **/usr/opt** in the DG/UX file system. For example, for a third-party package named **retail**, you might create a virtual disk named **usr_opt_retail** and mount it at **/usr/opt/retail**. That way, the name of the virtual disk reminds you of its mount point.

Work directories

Work directories, such as software development build areas or large databases, may serve as common work areas for your system's users. If such work directories are too large for a single disk drive, or if you suspect that disk I/O performance could deteriorate during multiuser access, you can create an aggregation of virtual disks to distribute the work directories onto multiple partitions on different disk drives.

Tools packages

Tools are commonly placed in **/usr/local/bin**. The DG/UX system has a **/usr/local** mount point included in the **/usr** file system. You can place a tools packages on a separate virtual disk. An appropriately named tools directory is easily recognizable and accessible to users on your system. You may choose to limit users to read-only access to a directory containing tools.

Temporary file space (tmp)

User programs need temporary space for startup and execution. Large program compilations, heavy network traffic, and large database I/O activities use temporary file space.

To segregate temporary file space from the / directory, you can create a virtual disk for temporary file space and mount it on the **/tmp** directory. By default, the system allocates 40,000 blocks to the **root** virtual disk for its file system. After you load the / file system, 12 Mbytes remain as free space that you can use for **/var** and **/tmp**. All subdirectories of **/var** use the same space. Also, you can create separate virtual disks for the **mail** and **news** subdirectories of **/var**.

IMPORTANT: If you want to link the **/var/tmp** directory to the **/tmp** directory (or the **/tmp** directory to the **/var/tmp** directory), use relative pathnames. Using absolute pathnames causes problems when you attempt to install an update release.

Front-end caches

Caching configurations use NVRAM or a fast disk as a fast and stable front-end device and a slower disk functioning as the slow back-end device. Even though the RAM board has limited storage capacity, its fast I/O performance greatly improves the overall performance of applications whose network-driven I/O activities are synchronous transmissions that occur in bursts. A disk serving as the back-end device has more storage capacity than does the front end.

In response to an I/O request, the system attempts to write the data to the fast front-end device. If the front end is already full of data, it will flush one or more of its buffers to the back end to make room for the incoming data. You will get optimal performance when the majority of the data being accessed fits on the front-end device.

If you intend to use an entire NVRAM board as a front end, select the largest block size offered, such as 3967 blocks. This number of blocks results from subtracting 33 blocks for system partitioning overhead from a total capacity of 4000 blocks. Alternatively, if a fast disk is used as a front end, as a rule, make the front end one-tenth the size of the back end.

```
front-end =  10% (back-end)

front-end = .10 (200,000 blocks)

front-end = 20,000 blocks
```

Deciding where to mount file systems

The hierarchical tree is a flexible structure, allowing you to organize virtual disks and file systems with respect to root (/). By convention, the **root**, **usr**, and **usr_opt_X11** virtual disks are placed at the locations shown in Figure 4–1. Some guidelines follow.

- Do not rearrange the locations or contents of **root**, **usr**, **usr_opt_X11** and any other operating system software. Doing so will cause trouble when you upgrade your DG/UX operating system. Reliable locations of key system files allow a quick and easy upgrade.
- If mounting third-party packages, follow the advice given in the documentation and release notices. By convention, third-party packages are mounted at **/usr/opt**.
- Establish mount points for all other file systems beneath root (/).

  The **/etc/fstab** file lists all usable file system mounts. At system boot time, the DG/UX system checks the **/etc/fstab** file and mounts all those listed, making them accessible to users. More information on the **fstab** file is given in Chapter 8.

Local file systems

A local file system is one that is located on a device attached to your local AViiON computer.

Locally mounted file systems may be of four types. They are:

- DG/UX
- CD-ROM
- DOS
- memory file system

Generally, you will create DG/UX file systems on virtual disks that you create and establish their mount points. If you have a CD-ROM device or DOS-formatted diskette whose contents exist, however, you need only establish its mount point and identify its file system type, to make the device's contents usable. For a memory file system, not only must you establish its mount point and identify its type as a memory file system, but also you must specify its size. Chapter 6 gives instructions on establishing a file system's type and mount point.

Figure 4–2 expands the view of the typical DG/UX file system shown in Figure 4–1 to include local file systems that do not require virtual disks. File systems **/usr/opt/floppy** and **/usr/opt/docbrowse** are a DOS-formatted diskette and a CD-ROM, respectively.

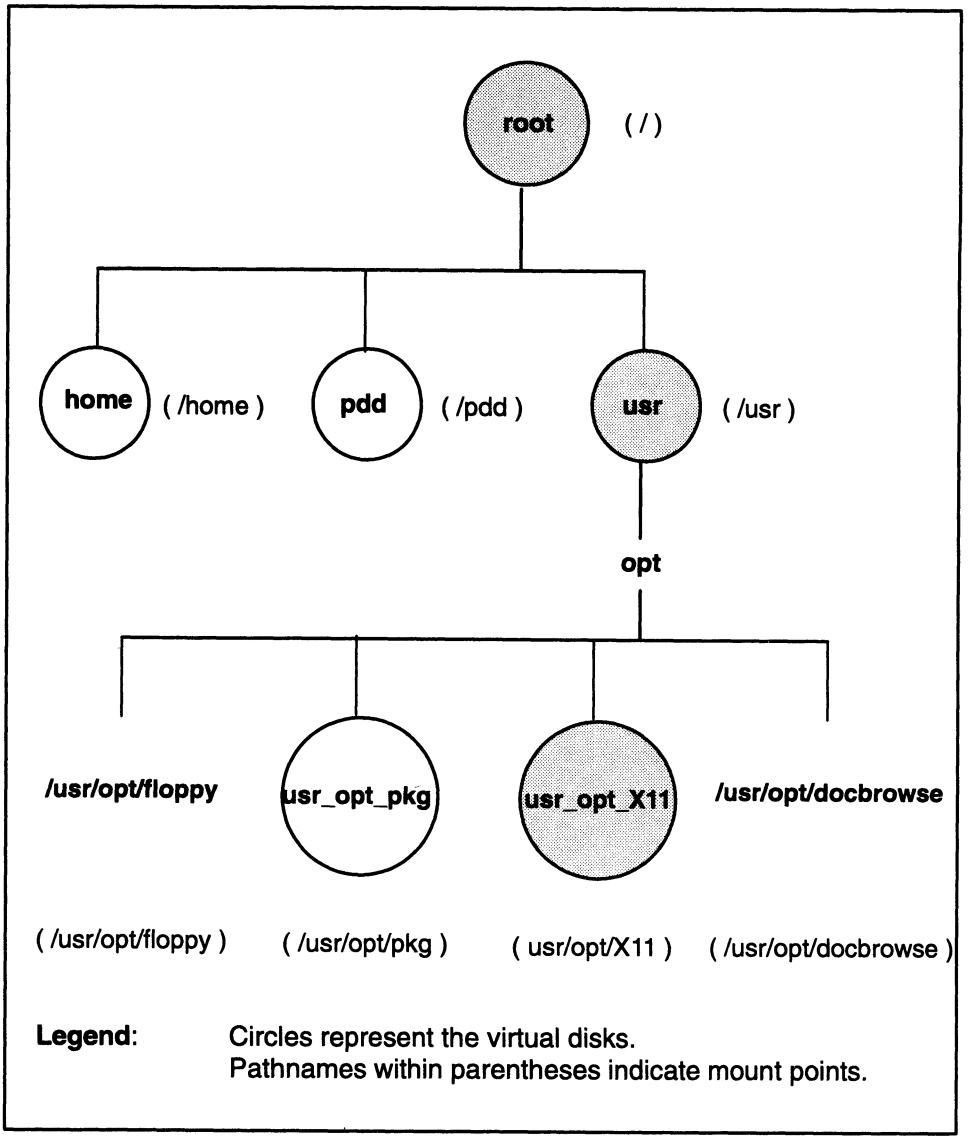


Figure 4–2 Sample mount points on the DG/UX directory structure

Remote file systems

A remote file system is one that is located on a device attached to a remote host that you can access by way of a LAN. You need to know the location of the remotely mounted file system on the remote host and the mount point you desire on your local system. Mounting remote file systems eliminates the need for duplicating file systems throughout configurations that are connected by a LAN.

Figure 4–3 expands the view of the file system shown in Figure 4–2 to include remotely mounted file systems **/pdd/pics/image**, **/pdd/notes**, **/pdd/conf/papers**, and **/pdd/spleen/games**.

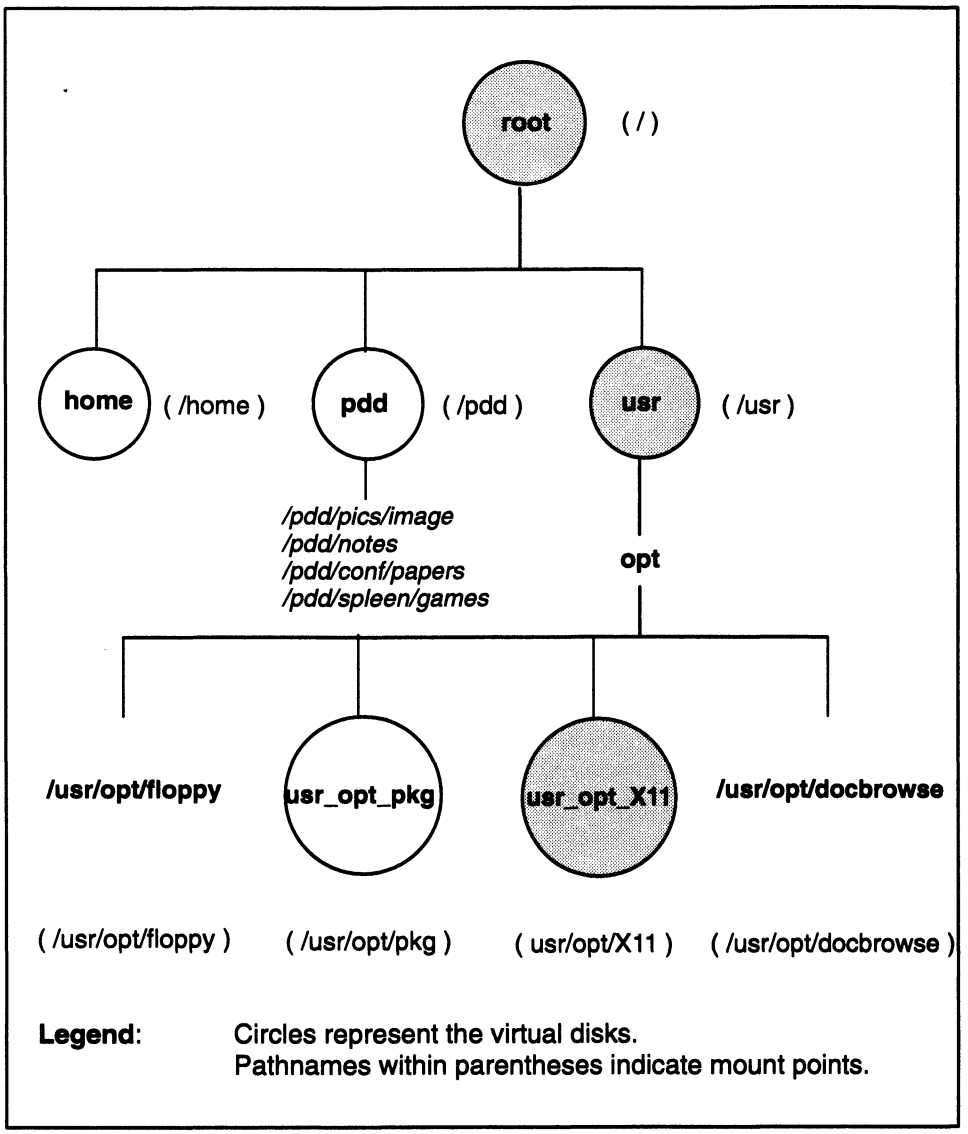


Figure 4–3 Sample mount points on the DG/UX directory structure

Figure 4–3 shows the following remote and local mount points.

Remote Mount Point	Local Mount Point
dot:/pdd/sam/image	/pdd/pics/image
oz:/pdd/notes	/pdd/notes
kansas:/pdd/otis/papers	/pdd/conf/papers
toto:/pdd/spleen/games	/pdd/spleen/games

The remote mount point begins with the name of the remote host followed by a colon (for example, **dot:**).

The examples indicate that remote and local mount points do not have to be identical. Select an appropriate local mount.

Mapping your data and work areas onto disks

Your second objective is to map your data and work areas onto disk drives.

Deciding how to best use the device depends on the capabilities of the device itself, the application's requirements, and your performance goals. You should create virtual disks and DG/UX file systems on disks to maximize disk use and performance. The following sections give recommendations and possibilities for each device type.

Winchester disks and disk arrays

Traditionally, you organize Winchester disks into virtual disks and DG/UX file systems. However, if required by an application, it is possible to use a "raw" disk (a virtual disk but no file system). Some database products require virtual disks but use their own style of file system instead of a DG/UX file system. Consult your database documentation for requirements. After you create a DG/UX file system, you must also create its mount point to allow access.

Rewritable magneto-optical and WORM disks

Neither the magneto-optical nor the WORM disk uses a virtual disk or a DG/UX file system. However, devices configured with virtual disks and DG/UX file systems can still be used. Refer to the OpStar™ documentation listed in the Preface for details.

NVRAM

Traditionally, you would organize NVRAM boards into virtual disks but not DG/UX file systems. NVRAM is used as a front-end device for data caching applications.

Diskette

A limited capacity diskette is best for storing a single file system, which precludes the need for a virtual disk. A diskette file system may be of two types: DG/UX or MS-DOS. Define a DOS file type if you intend to use the diskette in both a personal computer running MS-DOS and the DG/UX system environments. Otherwise, define the file system as a DG/UX type. You must create the file system's mount point to allow access.

Should you choose to organize multiple file systems on a diskette, however, virtual disks would be required. Again, you must establish its mount point to allow access.

CD-ROM

Since you cannot write to a CD-ROM, you create neither a virtual disk nor a DG/UX file system. The CD-ROM, however, contains a file system for which you must create a mount point to allow access.

Memory file system

Like the diskette, the memory file system has limited capacity, so does not use a virtual disk. However, you must specify the amount of memory you want to use and create a mount point to allow access.

Table 4–2 provides a summary of recommendations for using the disk devices and database applications.

Table 4–2 Disk and database applications virtual disk and file system needs

Disk Device Type	Create Virtual Disk?	Create File System?
Winchester disk	✓	✓
Rewritable Magneto Optical and Write Once, Read Many (WORM)	See your hardware documentation and application documentation, if applicable.	
NVRAM	✓	
Diskette		✓
CD–ROM		✓
Memory file system		✓
Databases	✓	See your database documentation

You must create a mount point for each file system on the device.

Assessing disk capacity

Knowing a disk drive's model number can help you determine its capacity. You can find the drive model number (and perhaps the capacity) in the hardware installation and setup manual or the packing list. If those documents do not provide capacity

information, see the following manual pages for the most current information on supported devices: **sd**(7), **cird**(7), **cimd**(7) and **cied**(7).

For disk-array storage systems, a drive's capacity depends on how you bind the storage system's disk modules into physical disks. Currently, the disk-array storage systems offer disk drives of the following capacities: 500 Mbytes, 1.0 Gbyte, 1.2 Gbytes, 2.0 Gbytes and 4.0 Gbytes. For example, two 2-Gbyte modules bound into a RAID-1 mirrored pair yield a physical disk with 2.0 Gbytes of storage. Five 2-Gbyte modules bound into a RAID-5 group yield a physical disk with 8.0 Gbytes of storage. For more information, see the documentation for your disk-array storage system.

In addition to the model number and capacity, you also need to know the DG/UX device name for each disk drive. For more information on DG/UX device names, see Chapter 2 and Appendix A.

Viewing a disk device's layout

To determine remaining disk drive resources as you create virtual disks display the drive's layout periodically. Follow this path through **sysadm** to display a drive's layout:

```
Device → Disk → Physical → List
```

Specify additional options as **sysadm** asks for them. Listing disks is very helpful when you want to create more than one virtual disk on the same disk drive. You can also check the space occupied by a virtual disk or verify the layout of a removable medium in a given drive.

Figure 4–4 shows a sample listing for a system disk obtained by the following **sysadm** sequence.

```
Device -> Disk -> Physical -> List
Physical disk(s): sd(ncsc(0,7),0,0)
Listing style: partitions
List label:[no]
```

```
Disk name                State   Reg? Format Total blocks  Free blocks
sd(ncsc(0,7),0,0)         avail     y  vdisks      1295922        22768

Name                          Role                Address         Size
swap                                                  859       100000
root                                                100859        80000
usr                                                 180859       300000
usr_opt_X11                                         480859       200000
usr_opt_networker                                   680859        50000
udd                                                 730859       300000
usr_opt_gif                                        1030859        60000
usr_local                                          1090859        50000
var_opt_relimon                                    1140859         2500
usr_opt_xdt                                        1143359        50000
<free space>                                       1193359       122563
```

Figure 4–4 System disk layout

The top two rows give the

- physical disk name, here shown as **sd(ncsc(0,7),0,0)**
- state (avail means available: not owned or registered by a different host system)
- registration status (y means registered, n means not registered, c means registered in compatibility mode)
- software format (vdisks means virtual disks, ldisks means logical disks)
- total number of 512-byte disk blocks, and total number of free disk blocks

The following rows give the

- virtual disk name (for a named virtual disk) or name of parent virtual disk (for an unnamed child partition)
- role (for a multiple-piece disk, displayed as piece *n* of *n*)
- address of the starting disk block on the physical disk
- size in disk blocks

Most virtual disks are one partition, although you may create an aggregated virtual disk with more than one partition.

Strategies for mapping virtual disks to disk drives

There are several ways to arrange virtual disks on disk drives:

- Contiguous virtual disks (one or more virtual disks created in sequence on a disk drive to use all available space)

- Multiple-partition virtual disk partitions (aggregations) of any size on different disk drives or the same disk drive
- Mirror: a virtual disk that uses two or more virtual disks of a given size on different disk drives for data redundancy (known as software disk mirroring)
- Software disk cache: a small, fast front-end virtual disk (NVRAM) associated with a large, slower back-end virtual disk (usually located on a disk) to provide a large, fast storage resource.

  Software disk mirroring and software disk caching are sophisticated resource allocation strategies that you may select to improve performance or to ensure data reliability. See Chapter 5 for details.

  Using contiguous or multi-partitioned virtual disks are the more traditional options; they are discussed in the next sections.

Contiguous virtual disks

The simplest way to arrange virtual disks is to take the default starting address when you create each virtual disk until you use all available space. This way, the virtual disks are contiguous on each drive. You also can specify the whole physical disk address space as a virtual disk.

Instructions for creating virtual disks are in Chapter 7.

Multiple-piece virtual disks (aggregations)

If you need enormous amounts of file space or have fragments of disk space scattered across different disk drives, you can aggregate partitions on different drives.

For example, assume you want 2,000,000 disk blocks available in bulk. You could create an aggregated virtual disk from partitions on three disk drives as follows:

Virtual Disk Partition	DG/UX Device Name	Size (in blocks)
1 of 4	sd(cisc(0),0,1)	200,000
2 of 4	sd(cisc(0),0,1)	400,000
3 of 4	sd(cisc(0),0,2)	649,500
4 of 4	sd(cisc(0),1,0)	550,500
TOTAL		2,000,000

You can specify up to 120 partitions for an aggregation.

Aggregations can also be striped. Software disk striping is described in the next section.

Instructions for creating aggregated virtual disks are in Chapter 7.

Software disk striping

Software disk striping improves disk I/O performance by distributing the disk load across multiple disk drives. Software disk striping is useful for applications that perform many random reads and writes or many sequential reads. Such an application might query the entire database requiring a sequential read of the database. If a database occupies multiple disk drives, those drives can share the I/O load, thus improving performance.

Software disk striping does not improve I/O for applications that perform intensive sequential writes. To maintain the integrity of a database, the application might perform a series of writes to a log file before updating the data file.

A disadvantage of software disk striping is greater vulnerability to failure: if one disk fails, the entire virtual disk becomes inaccessible. Of course, you can always software mirror the striped disk.

IMPORTANT: Software disk striping is different from hardware disk-array striping. For information on hardware disk-array striping, see the disk-array storage system documentation.

Requirements and recommendations for striped virtual disk are as follows.

- The virtual disk consists of multiple pieces that must be the same size.
- For best performance, put each piece on a different disk drive, or better yet, disk controller.
- You can set up a virtual disk for striping only when you create it; that is, you cannot modify an existing virtual disk to allow striping.
- The stripe size must be an even divisor of the piece size.

  For example, possible stripe sizes for a 400-block partition are 2, 4, 5, 8, 16, 25, 50, 100, and 200. Put another way, the size of a piece must be a multiple of the stripe size — 400 blocks is a multiple of possible stripe sizes 2, 4, 5, 8, 16, 25, 50, 100, and 200. If you intend to put DG/UX file systems on the virtual disks, for maximum performance, set the stripe size to be a multiple of the data element size. The default data element size is 16 blocks. See the **mkfs**(1M) manual page for more help on selecting a stripe size.

  If you are not creating a DG/UX file system on each component, select a stripe size that would be most beneficial for the application. A suitable stripe size is generally a multiple of the size of the average reads and writes that the application issues. Good choices for stripe size are 16, 32, and 64, for example.

- You cannot change the size of a striped virtual disk after creating it, for example, by shrinking or expanding.
- You cannot stripe virtual disks that contain the operating system — **root**, **usr**, and **swap**.

To implement software data striping, you need to create the virtual disk with this purpose in mind. Once you have created the striped virtual disk and its file system, striping is transparent to your applications. You use and manage the striped file system just like any other file system. The only difference is that you cannot change the size of a striped virtual disk; you cannot expand or shrink it.

The system implements striping by placing consecutive file elements in the file system so that they alternate from one partition of the virtual disk to the next. The system will place the first data element in the first partition, the second data element in the second partition, and the third data element in the third partition, and so on. The performance advantage results not only because you have distributed the I/O load across three disks, but also because you are using the hardware's read-ahead implementation to get the next element on that disk, even before you have explicitly requested it.

If your application does not appear suited to striping, do not attempt to implement striping: striping can have a negative impact on performance for inappropriate applications.

Completing the disk planning worksheets

You are now ready to plan your disk resources by answering these questions.

- What are the DG/UX disk drive device names?
- What is the capacity of each disk drive?
- What name do I want to give each virtual disk?
- On the DG/UX directory structure, where do I mount each local file system associated with each virtual disk I have created?
- Will there be any software striped virtual disks or aggregations (virtual disks with two or more virtual disks) and how large will each virtual disk be? If a virtual disk is striped, what is its stripe size?
- Do I have any pre-existing file systems (e.g., CD-ROM or DOS diskettes) to mount? Do I want to capture memory and use it as a file system?
- Do I want to mount any remote file systems? If so, where are they remotely located? Where will I mount them on the local system?

  If you don't care which physical disk holds a virtual disk, you can specify space alone and have **sysadm** choose the physical disk. You cannot do this for a striped virtual disk. For any virtual disk for which you plan to let **sysadm** choose the physical disk(s), write "n/a" in the physical disk specification box in the worksheet.

  For the size of a virtual disk, you may choose all space remaining on the physical disk. **Sysadm** will display this number when you create the virtual disk and you can enter it then. For each such disk, wait until you create the disk to enter the size of the virtual disk in the worksheet (see Chapter 7 for information on creating a virtual disk).

Virtual disk planning

Figure 4–5 shows a sample virtual disk planning worksheet. The sample entries appear in italic typeface. Blank worksheets are provided in Appendix C.

The sample system has two physical disks: **sd(cisc(0),0,0)** and **sd(dgsc(0),1,0)**

Sample Worksheet
Virtual Disk Layout Worksheet

Virtual Disk Name	Mount Point Directory	Striped ?	Size in blocks	Drive Name *sd(cisc(0),0,0)* *1,200* Mbytes		Drive Name *sd(cisc(0),1,0)* *4,800* Mbytes	
				Piece	Size in Blocks	Piece	Size in Blocks
swap	NA			1	*72,000*		
root	/			1	*40,000*		
usr	/usr			1	*300,000*		
usr_opt_X11	*/usr/opt/X11*			*1*	*140,000*		
usr_opt _networker	*/usr/opt/networker*			*1*	*50,000*		
var_opt _networker	*/var/opt/networker*			*1*	*5,000*		
usr_opt_xdt	*/usr/opt/xdt*			*1*	*60,000*		
pdd	*/pdd*			*1*	*200,000*		
usr_opt_pkg	*/usr/opt/pkg*			*1*	*200,000*		
database_db	*/database/db*					*1*	*5,000,000*
home	*/database /home*					*1*	*160,000*
usr_opt_ fredware	*/usr/opt /fredware*	✓	*16*	*1*	*40,000*	*2*	*40,000*
usr_opt_ trackingsys	*/usr/opt /trackingsys*	✓	*32*	*1*	*20,000*	*2*	*20,000*
Total Used					*1,127,000*		*5,220,000*
Total Capacity					*2,457,600*		*9,830,400*
Free Space					*1,383,600*		*4,525,400*

Figure 4–5 Sample virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

Local file systems

Figure 4–6 is a sample worksheet for the file systems and mount points shown in Figure 4–2. Again, sample entries appear in italic typeface. Blank worksheets are provided in Appendix C.

Sample Worksheet
Local Existing Layout Worksheet

Device Type	Device Name	Local Mount Point
DOS diskette	*sd(insc(0),5)*	*/usr/opt/floppy*
CD-ROM	*sd(insc(0),3)*	*/usr/opt/docbrowser*

Figure 4–6 Sample local file system planning worksheet

Remote file systems

Figure 4–7 is a sample worksheet for the remote hosts and mount points shown in Figure 4–3. Again, sample entries appear in italic typeface. Blank worksheets are provided in Appendix C.

Sample Worksheet
Remote Pre-Existing Layout Worksheet

Remote Hostname	Remote Mount Point	Local Mount Point
dot	*/pdd/sam/image*	*/pdd/pics/image*
oz	*/pdd/notes*	*/pdd/notes*
kansas	*/pdd/otis/papers*	*/pdd/conf/papers*
toto	*/pdd/spleen/games*	*/pdd/spleen/games*

Figure 4–7 Sample remote file system planning worksheet

End of Chapter

5 Advanced planning: organizing virtual disks into mirrors and caches

The previous chapter introduced simple methods for organizing your disk space into virtual disks and file systems. This chapter offers two advanced methods for disk storage: mirroring and caching. Use the Worksheets at the end of the chapter for recording mirroring and caching configurations.

Software disk mirroring

A software disk mirror offers high availability through redundant images. A mirror comprises up to three virtual disks that are identical images of each other: they all contain the same data.

Software disk mirroring is different from hardware disk mirroring as provided by a disk array RAID-1 mirrored pair. For information on hardware disk mirroring, see the 014-series manual supplied with the storage-system hardware.

Mirrored virtual disks include these benefits:

Data availability

If the disk that holds an image fails, the DG/UX system automatically sends all read and write operations to another image until you repair or replace the failing disk. As long as one virtual disk in the mirror remains functional, users experience no interruption in service.

Data integrity

With multiple identical images of your data, you greatly reduce the risk of losing data due to hardware failure on one drive.

Performance

Mirrored virtual disks whose images lie on different physical disks offer increased throughput in environments where multiple concurrently running applications perform intensive reads of the mirror. This benefit arises because the system can use the images of the mirror as individual virtual disks during concurrent read operations, using one image to satisfy one read request while using another image to satisfy a different read request. Thus, the mirror distributes the I/O activity across multiple disk drives.

While a single running application will not exhibit increased performance, the system overall will show an improved

performance. This benefit does not occur in environments where only one running application reads the mirror at a time, nor does it occur in environments where the images do not reside on different physical disks.

A mirror appears as a virtual disk on the system, which you can access the same as a virtual disk that is not mirrored. Figure 5–1 shows a typical mirrored virtual disk configuration.

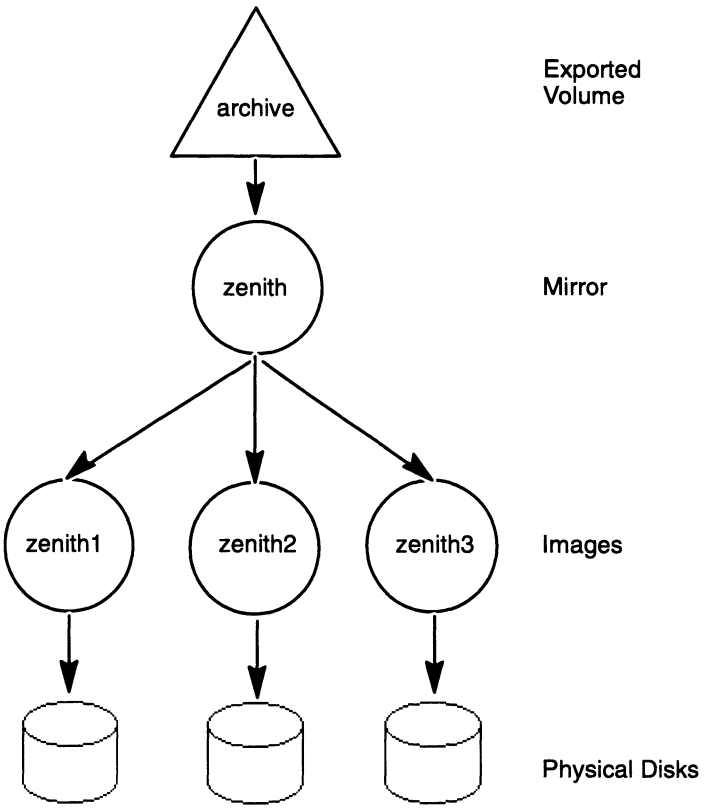


Figure 5–1 Typical mirrored virtual disk configuration

Named images appear in the **/dev/dsk** and **/dev/rdsk** directories, but you may access them for read-only operations while they are part of the mirror.

When a user or an application writes to a file on a mirrored virtual disk, the system duplicates the write operation on each mirror image. When a user or application performs a read operation on a mirrored virtual disk, the system selects one of the images to satisfy the read request.

If a read operation fails on one image, the system satisfies the read request by reading from another image instead. If a bad block caused the original failure, and the physical disk on which the image resides has bad block mapping enabled, the system attempts to repair it by replacing the bad block with a known good block and

updating it with the correct data from a good block on another image. If this repair operation succeeds, the image remains an active member of the mirror. If the repair operation fails, however, the system responds as to a failed write operation, described in the next paragraph.

If a write to any image fails, the system marks the image as corrupt and suspends further use of the image. If there are still functioning images in the mirrored virtual disk, the system continues to serve read and write requests to the mirror.

When an image becomes corrupt, the system issues a warning. An example follows:

```
Oct 22 10:31:45 homer dg/ux: Warning:  Image
'vdm(marge_image2,2CC7EE0E,0C052A9D,0)' on mirror
'vdm(marge_mirror,2CC7EE30,0C052A9D,0)' has failed
(status = 77005171).
```

Probably, a second message will follow when the mirror attempts to update the time stamps of both images, and gets a failure when writing to the failed one.

Once an image becomes corrupt, the data in the image is inconsistent with the data in the other images in the mirror: the images are no longer identical. After you fix the problem and restore the image to service, you then need to synchronize the corrupted image with a known good image. The system then copies the master image onto the out-of-sync image.

The system conducts synchronization without interrupting user access to the mirror. The system handles any concurrent user access to the mirror in a fashion that protects data integrity and keeps all images up to date. Except for a possible effect on disk I/O performance, users will not know that synchronization is occurring.

As part of rebooting activities, the system registers physical disks, synchronizes mirror images, and logs a message using the **syslog** error logging facility. The message is from the **kern** facility and is level **warning**. If synchronization fails, the system logs a **kern.err** message, which by default goes to the system console and to **/usr/adm/messages**.

Subsequently, when the mirror attempts to update the time stamp of both images, it gets a failure when writing to the failed one.

Considerations for mirrored virtual disks

Before you create a mirror, you need to decide several things:

- How many images will the mirror include?
- On which physical drives and controllers will you put the images?
- How will you configure the virtual disks for each image?
- How many images must be available before mounting the mirror?
- Will the system synchronize images (as needed) at every boot? How quickly should the system perform the synchronization?

How many images?

In deciding how many images will make up the mirror, you need to consider the tradeoff between availability and cost. A mirror comprising three images provides higher availability of data: the mirror can withstand more individual image failures before it becomes completely out of service. On the other hand, you can save disk space by having only two images for the mirror. To make a three-image mirror that has the effective size of 300 Mbytes, you need to create three 300-Mbyte virtual disks, making a total resource cost of 900 Mbytes.

Where should you put them?

As you consider which physical drives and controllers will hold your mirror images, you encounter a similar tradeoff between data availability and resource expense. Ideally, you want each image to reside on a different physical disk attached to a different hardware controller. By isolating the images this way, you insulate them from each other's possible hardware failures. Consider, for example, a three-image mirror where all the images are on the same physical disk. If the disk or disk controller fails, the entire mirror becomes inaccessible until you can repair the disk and restore the data from backup. Obviously, placing each image on a different physical disk is the wisest configuration. The same principle applies to the disks' hardware controllers. If the images are all on different disks, but the disks all depend on the same controller, you risk losing the entire mirror if the controller fails.

For example, you could create three identically sized virtual disk images to form a mirror virtual disk that occupies 2,400,000 blocks, provides 800,000 blocks of storage capacity, and spans three different disk controllers. Its arrangement follows:

Virtual Disk Image	Physical Device	Size (in blocks)
1 of 3	sd(ncsc(0),0,0)	800,000
2 of 3	sd(ncsc(0),1,0)	800,000
3 of 3	sd(ncsc(0),2,0)	800,000
TOTAL		2,400,000

What kind of virtual disks should you create?

After you decide on the number and location of the images, you need to make the decisions that are pertinent any time you create a virtual disk. The only requirement for mirroring is that all virtual disks be the same size. The names of the virtual disks and the number and placement of any virtual disk pieces, however, are completely up to you.

What minimum number of images do you require?

You need to decide how many images must be available before you can mount the mirror and make it accessible to users. This question is just another way of asking how many corrupt images you will tolerate.

The data in a mirror image can be in either of three states:

- completely good
- in the process of being synchronized and will eventually be good
- corrupt

If your mirror has three images and you require at least two functional images, the data is either completely good or is in the process of being synchronized. If you are more interested in availability than reliability, with one corrupt image, one being synchronized, or only one good image, you will be using only one image for your transactions. Should you trade off reliability for availability, you may decide to delay your transaction processing until the image has completed synchronization. You will then have two truly good, functional images.

Keep in mind that you may not be on hand to evaluate the state of the system after a system crash. After a power outage, for example, a Data General AViiON computer can reboot itself without operator intervention as soon as power returns. You can configure the system to come up all the way to run level 3, where local users and OS clients may begin work and access file systems. The availability requirement you set determines whether or not users can access a mirror that may have lost an image in the crash. If the data in the

mirror is such that you do not want it in use if there is only one functioning image, you should set the availability requirement to two or three.

Setting the availability limit allows you to enforce either a policy of high data availability, where downtime is an issue, or a policy of high data integrity, where you depend on redundant images to protect from data loss. For high availability, select a lower integrity requirement. For high integrity, select a higher availability requirement.

Note that when counting "in-sync" images, any image that is being automatically synchronized is considered to be in-sync for the purposes of deciding if a sufficient number of images is available to put the mirror into use. For example, consider a mirror with three images — one is consistent and two are inconsistent — for which auto-sync is enabled. In this case, two synchronize operations are started, and the mirror is immediately considered to have three consistent images. You cannot tell the system to wait for the automatic synchronize operations to complete before placing the mirror into service.

How many lost images will be tolerated?

You are balancing the minimum number of images you can operate with and the maximum number of images you can operate without. An image may be lost (because it can't be located) if its host disk is disconnected or if the disk is unavailable because of a crash, for example.

The value you select for this parameter is again a question of reliability versus availability. Suppose you tolerate one lost image, which you know to contain the most up-to-date data. If you tolerate this loss, the remaining images will still be available for use, but the data integrity may be questionable. Alternatively, if you aim for reliability, you must halt service until the damaged disk is back on-line, sacrificing availability. To maximize reliability, you may want to choose 0 as the maximum number of lost images to tolerate; for availability, perhaps, 1. Of course, this selection must be made with consideration of the minimum number of images required.

Should the system automatically synchronize mirror images at boot time?

When the system boots, it examines the images of each mirror that it finds. If it appears that the images may be out-of-sync with each other (for example, because of a system crash), you can configure the system to automatically initiate a synchronize operation to make the images identical. You may elect automatic synchronization at system boot when you create the mirror. After

you create the mirror, you may modify the sync speed through the throttle operation.

Refer to Chapter 7 for instructions for creating a mirror and Chapter 10 for details on throttling sync speed.

How do I back up a mirror image?

Different from previous releases, DG/UX 5.4R3.10 allows you to perform a higher performance mirror image backup through its mirror fracture operation.

The mirror fracture operation is a major improvement over its predecessor. Rather than explicitly unlinking the image to be backed up, you fracture it. In a single operation, fracturing unlinks the image from the mirror and logs the data I/O activity intended for the image to a bitmap file for the duration of the backup. Upon the completion of the backup, you synchronize the image to the mirror. Instead of copying the entire contents of the master image to the fractured image, the fracture operation copies only the changed data stored in the bitmap to the image. This abbreviated synchronization improves performance considerably. Refer to Chapter 10 for details on the fracture operation.

Software disk caching

Disk caching associates two virtual disks, typically one on a small, fast device (front end) with another on a large, slow device (back end). An application uses the fast device for read and write operations while the operating system duplicates these operations on the larger device. The purpose of the configuration is to accelerate file system access for I/O-intensive applications without risking data integrity. Figure 5–2 shows a typical caching configuration.

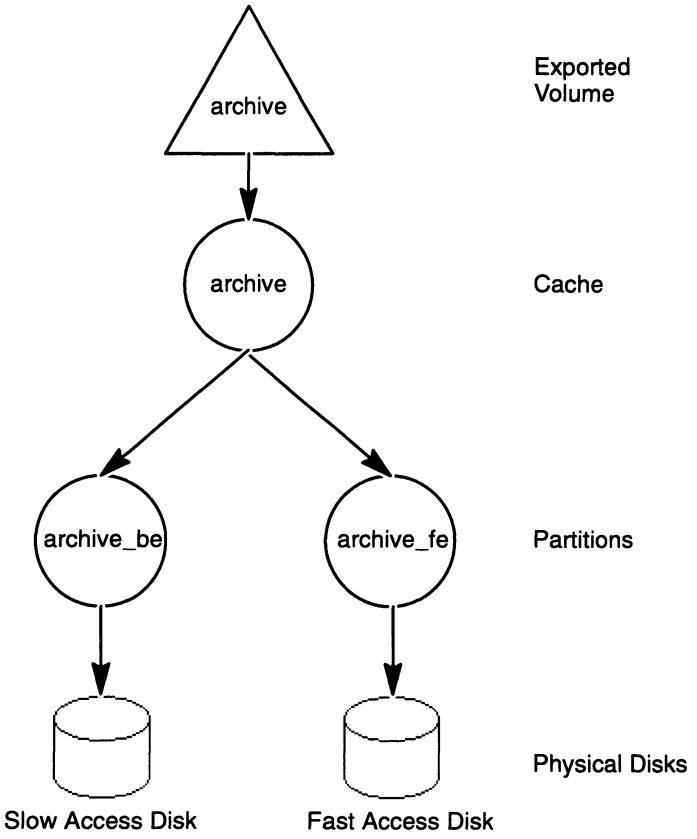


Figure 5–2 Typing caching configuration

In this configuration, the cache named **archive** associates a slow back-end device, **archive_be,** with a fast front-end device, **archive_fe**.

The primary caching configuration uses a nonvolatile random access memory (NVRAM) or a battery backed-up random access memory (BBURAM) board functioning as the front-end device and a physical disk functioning as the back-end device. Although the RAM board has a relatively small storage capacity, its superior I/O performance can boost the performance of I/O-intensive applications such as database management systems. In addition, you may use a disk device, preferably a fast one, as a front–end device.

The system supports these caching configurations:

- Single back-end device and a single front-end device.
- Multiple back-end devices sharing a single front-end device.
- Multiple back-end devices with multiple front-end devices (which may reside on the same NVRAM board).

Figure 5–3 shows sharing one front-end device with two back ends.

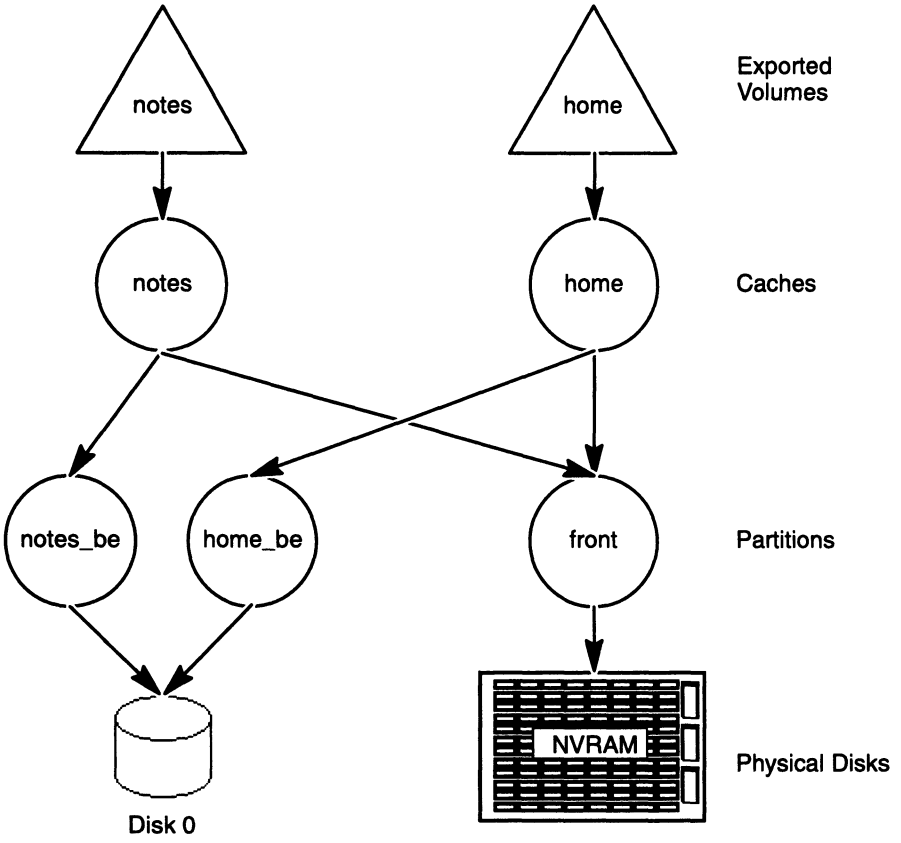


Figure 5–3 Cache sharing of one front end with two back ends

RAM-based caches introduce the risk that a failure could lose the data in the cache before the system has a chance to write it to the more stable back-end device. The ideal front-end device is nonvolatile or battery backed-up RAM or a fast disk, which provides the required speed as well as stability.

The virtual disk functioning as the back end, meanwhile, provides greater storage capacity than a RAM device or disk device and has the added stability normally attributed to disk drives. The cache and its front- and back-end devices must be on local devices; however, other remote systems can access them through ONC/NFS.

The DG/UX system optimizes disk caching for accessing DG/UX file systems rather than for other data structures (such as databases built directly on virtual or physical disks). You may, nevertheless, use cached virtual disks for any purpose that benefits from the accelerated I/O performance. You may consider running your applications both with and without disk caching, then comparing results to see which configuration offers the best performance. The following section tells how to get the most out of a cached disk configuration.

As I/O requests arrive for the cached virtual disk, the system allocates buffers in the front-end device to hold data for the back end device, or disk. These allocated buffers are either clean or dirty. A clean buffer is one whose data matches the corresponding buffer on disk. For example, a buffer copied from disk to cache for a read operation is clean because it contains the same data that is on the disk. A dirty buffer contains data that is inconsistent with the disk. For example, a buffer written by an application but not yet flushed to disk is dirty.

Freeing buffers involves seeking the least frequently accessed buffers and flushing their contents to disk (if dirty) and then flagging them as unallocated. The system is then free to allocate them for more I/O requests.

To determine which buffers are the least frequently accessed, the system maintains a single weight number for each buffer. Each time either a read or write I/O request accesses the buffer, the system increments its weight number. The system uses the weight numbers to find the least frequently used buffers to remain in the cache and the most frequently accessed buffers to be freed from the cache.

When you shut down the system (using the **shutdown** and **halt** commands), access to the cached disk stops just as for any other disk. The cache flushes its dirty buffers to disk and flags all buffers as clean.

If an abnormal failure, such as a crash or panic, causes the system to stop without executing the normal **halt** sequence, the cache may contain dirty buffers that were never flushed to disk. In this case, the cache device maintains the data until the DG/UX system reboots. After the system reboots, the buffers are recovered.

Building a cache involves linking a virtual disk (back end) with a new virtual disk to serve as the front end. You create the front end just as you do a virtual disk: by partitioning an existing virtual disk or by using entirely an existing virtual disk. The cache itself is an abstraction. You refer to the entire configuration as a cache; but, the action really occurs between the front- and back-end devices. You must also configure the front-end device in the kernel. The device driver name for the NVRAM board is **nvrd()**.

The cache inherits its name from the back-end device, which leaves the back end nameless. You may rename the back end, or you may leave it nameless. A back-end device name is important only when you need to refer to it specifically, when modifying or unlinking, for example. You must explicitly name a virtual disk to be used as a front-end device.

If you intend to create a cache in which one front end connects to one back end, by convention, you may name the cache **foo**, the back end **foo–be**, and the front end **foo–fe**. If, however, you intend to create multiple caches to share the same front end, you may adopt a more generic naming convention for the front end. For example, you may name the first cache and back end **foo** and **foo–be**; the second cache and back end, **bar** and **bar–be**; and the front end for both, **front**.

Besides identifying the back and front ends, you must also tune several parameters to optimize cache efficiency. Altogether, these tuned parameters form a cache policy.

Instructions for creating cached virtual disks are in Chapter 7.

Completing the disk planning worksheets

Figure 5–4 is a sample mirror worksheet. Sample entries appear in italic typeface. See Appendix C for blank worksheets.

Sample Worksheet
Mirror Layout Worksheet

Mirrored Virtual Disk Name	Mount Point Directory	Striped Image being Mirrored ?	Size in blocks	Drive Name *sd(cisc(0),0,0)* *1,200* Mbytes		Drive Name *sd(cisc(0),1,0)* *4,800* Mbytes		Drive Name *sd(cisc(0),2,0)* *1,200* Mbytes	
				Piece/ Image	Size in Blocks	Piece/ Image	Size in Blocks	Piece/ Image	Size in Blocks
usr_opt_ samware	*/usr/opt /samware*			*1*	*15,000*	*2*	*15,000*	*3*	*15,000*
usr_opt_ trade–data	*/usr/opt /trade–data*	✓	*16*	*1*	*30,000*	*2*	*30,000*		
Total Used					*1,124,000*		*5,217,000*		*15,000*
Total Capacity					*2,457,600*		*9,830,400*		*2,457,600*
Free Space					*1,386,600*		*4,528,400*		*2,442,600*

Figure 5–4 Sample mirrored virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

Figure 5–5 is a sample cache worksheet. Sample entries appear in italic typeface. See Appendix C for blank worksheets.

Sample Worksheet
Cache Layout Worksheet

Cached Virtual Disk Name	Mount Point Directory	Striped ?	Size in blocks	Drive Name *sd(cisc(0),0,0)* *1,200* Mbytes		Drive Name *sd(cisc(0),1,0)* *4,80* Mbytes		Drive Name *nvrd(0)* *2* Mbytes	
				Piece/ Image	Size in blocks	Piece/ Image	Size in Blocks	Piece/ Image	Size in Blocks
archive_fe	*/database* */frontend*			*1*	*100,000*				
archive_be	*/database* */backend*							*1*	*4,096*
Total Used					*100,000*		*0*		*4,096*
Total Capacity					*2,457,600*		*9,830,400*		*4,096*
Free Space					*2,357,600*		*9,830,400*		*0*

Figure 5–5 Sample cached virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

End of Chapter

6 Soft formatting and registering disk drives

Before you can create virtual disks or DG/UX file systems on disk drives, you must format them. The soft formatting sequence comprises these steps:

- Install a disk label
- Create virtual disk table
- Establish bad block mapping
- Install a bootstrap

Soft formatting also registers the disk. Registering a disk enables access to its virtual disks. You cannot access a virtual disk unless its host disk drive is registered.

Soft formatting

Soft formatting effectively erases any user data on a disk.

Some disk devices do not require soft formatting. Table 6–1 lists each type of disk device and its formatting requirements.

Table 6–1 Disk device formatting requirements

Disk Device Type	Soft Formatting Required?
Winchester disk	Yes
Rewritable Magneto-Optical and Write Once, Read Many (WORM)	No
NVRAM	Yes
Diskette	Yes, only if you intend to put more than one DG/UX file system on it.
CD-ROM	No
Memory file system	No
Application requiring raw device	Yes, only if application requires virtual disks.

A raw disk is one that contains virtual disks but not DG/UX file systems. The application may provide its own file management system.

IMPORTANT: Diskettes do not necessarily require formatting. If you intend to put only one DG/UX file system on a diskette or if you plan to use the diskette as a tape, do not soft format it. A DOS-formatted diskette that you wish to use in a DG/UX environment requires only a mount point.

You can perform all formatting steps in one **sysadm** sequence or shell command, or you can perform them individually. The easiest method is to perform the steps in one sequence.

All soft formatting steps

▶ To soft format a disk, follow these steps:

1. Make sure the drive or medium is write enabled (if this applies) and, for a diskette, properly insert it in a drive.
2. Follow this path through **sysadm**:

   ```
   Device -> Disk -> Physical -> Soft format ->
      All Soft Formatting Steps
   ```

   Sysadm prompts

   ```
   Physical Disk(s):
   ```

3. If **sysadm** does not display a list of devices, enter ? for a list; for example

   ```
   Physical Disk: ? ↵

   ...Choices are

       1  sd(cisc(0),0,0)
       2  sd(ncsc(0),1,0)
   ```

4. Select one of the disks listed, by name, number or the default name; for example, from the list above,

   ```
   Physical Disk(s): 2 ↵
   ```

5. **Sysadm** prompts you for each formatting step. First, it prompts

   ```
   Install Disk Label? [yes]
   ```

   A disk label contains the disk layout (tracks per cylinder, bytes per sector, and so on). SCSI disks (including those in a disk-array storage system) have generic labels and do not need labels. All other disk types require a label so that the system can access it. Another consideration is whether or not the disk will be bootable: bootable devices must have labels. In any case, installing a label does no harm, so take the default:

   ```
   Install Disk Label? [yes] ↵
   ```

6. Create a virtual disk table.

   The operating system needs a virtual disk table to keep track of the virtual disks on a physical disk. This table establishes the virtual disk format.

   ```
   Create Virtual Disk Table? [yes] ↵
   ```

7. Specify whether or not to use bad block remapping.

   Bad block mapping lets the DG/UX system note bad blocks and map them to a good area on the disk. (Bad blocks are disk blocks that are flawed and cannot properly hold data.) Bad block mapping is not required for disks whose controllers have hardware bad block mapping, but enabling it does no harm. The disk must be registered for remapping to occur. Answer **yes** unless you are formatting an NVRAM memory board, in which case answer **no**. For example,

   ```
   Establish Bad Block Mapping Facilities? [yes] ↵
   ```

8. Specify the size of the back block remap area.

   For a hard disk, the default remap area size is usually sufficient. For a diskette, we suggest 0 (since you will probably not want to remap, but instead copy and then discard a diskette if bad block develops on it). Enter a number or press Enter for the default. For example,

   ```
   Size of Bad Block Remap Area (blocks): [n] ↵
   ```

9. Specify whether or not to install a bootstrap.

   ```
   Install Bootstrap? [yes] ↵
   ```

   You need to install a bootstrap only if the disk will contain bootable software as it does if it contains **root** or **usr** virtual disks.

   The bootstrap consumes little space. For a diskette or NVRAM memory board, you should answer **no**. For all other disks, answer **yes**.

10. Specify the type of disk label.

    ```
    Disk label type [Generic SCSI] ↵
    ```

    The X window-based **sysadm** displays a list of label types you can choose from. If you are using the ASCII version of **sysadm**, you must enter a question mark (?) for the list of label types. Take the default, or specify the type of label you want; for example,

```
Disk label type [Generic SCSI] ↵
Caution:  creating a new virtual disk table will
    destroy all the data on physical disk
    sd(ncsc(0),1,0).   Do you wish to continue?
    [yes]
```

This warning emphasizes that the soft format procedure will erase any information on the disk. If you are sure you want to continue, press Enter:

```
sd(ncsc(0),1,0).    Do you wish to continue? [yes] ↵
```

Sysadm now creates the label and virtual disk table; then it registers the disk and creates the bad block remap area and bootstrap if you told it to. It displays

```
Virtual Disk Information Table created on sd(ncsc(0),1,0)
Physical disk sd(ncsc(0),1,0) registered.
```

Sysadm has software formatted and registered the disk. This completes the formatting steps. You may now want to create a virtual disk on the physical disk or NVRAM board (Chapter 7). To format another disk, repeat the steps in this section.

Installing a disk's label

1. To install a physical disk label, select the sysadm operation:

   ```
   Device -> Disk -> Physical -> Soft Format -> Label
   Disk
   ```

   Disk labels contain the disk geometry (such as tracks per cylinder, bytes per sector, and so on), information that the system requires to write to the disk, read from it, and keep track of damaged disk blocks.

2. Specify the device on which to install a label.

   ```
   Physical Disk(s): [sd(cisc(0)1,0)]
   ```

3. Select the type of the physical disk to be labeled.

   ```
   Disk Type:
   ```

   The choices are:

   ```
   1  Model 6442:  full-height ESDI, 327 MB
   2  Model 6541 or 6542:  SMD, 1066 MB
   3  Model 6555:  full-height ESDI, 648 MB
   4  Model 6661:  half-height ESDI, 330 MB
   5  Generic SCSI
   ```

After you label the physical disk, it is registered automatically.

An example of a disk label, including the bootstrap, follows:

```
Disk Label:
cylinders_per_drive     0
visible_cylinders_per_drive     0
tracks_per_cylinder     0
sectors_per_track       0
bytes_per_logical_sector        0
bytes_per_unformatted_sector    0
defect_info_start_sector        0
bytes_in_defect_info    0
number_of_relocation_areas      0
sectors_per_relocation_area     0
next_relocation_sector  0
interleave      0
head_skew       0
cylinder_skew   0
head_group_skew 0
spares_per_track        0
bytes_per_data_preamble 0
bytes_per_id_preamble   0
base_head_for_volume    0
flags   0
bytes_in_gap_1  0
bytes_in_gap_2  0
sanity_flag     305441741
version_number  1
Bootstrap: start = 17, size = 500, version = 1
```

To display a disk's label, select the **sysadm** operation `Device -> Disk -> Physical -> List`, which is described in a later section.

Creating a virtual disk information table (VDIT)

IMPORTANT: This operation replaces the create system areas operation in releases prior to DG/UX 5.4R3.00.

To create a virtual disk information table, select the **sysadm** operation `Device -> Disk -> Physical -> Soft Format -> Create VDIT`.

CAUTION: If a physical disk already contains data, this operation destroys all data on the physical disk.

In some cases, such as reinstalling the operating system, you may want to use this operation to deliberately destroy the data on a physical disk.

A virtual disk information table enables the physical disk to receive virtual disks that you create explicitly. It controls the mapping of virtual disks. You must create a virtual disk information table before you create virtual disks.

After you create its table, the disk drive is registered automatically.

An example of a newly created virtual disk information table, including the bad block remap area and bootstrap, appears as follows:

```
Disk name            State   Reg? Format Total blocks  Free blocks
sd(cisc(0),1,0)      avail     y  vdisks      1295922      1230279

Partition Name              Role              Address         Size
.Label,2CA9A8E1                                     0            1
.Primary_Vdit,2CA9A8DF                              1           16
.Bootstrap,2CA893F1                                17          500
.Primary_Bad_Block_Table,2CA893F2                 517            5
.Remap_Area,2CA893F3                              522          100
.Secondary_Bad_Block_Table,2CA893F4               622            5
<free space>                                    65627      1295279
.Secondary_Vdit,2CA9A8E0                      1295906           16
```

System partitions begin with a period (.). In this example, roughly the first 620 blocks and the last 16 blocks on the physical disk are reserved for system partitions. The numeric strings appended to the system partitions differentiate them from other system partitions located on other physical disks.

If you are familiar with previous releases of the DG/UX system, you will see that the new virtual disk information table uses for overhead not only the beginning of the physical disk but the final 16 blocks of the physical disk as well. To guard against accidental destruction of a virtual disk information table, the system puts two separate virtual disk information tables on a physical disk.

Among the system areas that the operation creates are the label and the primary virtual disk information table, which arranges the virtual disks that are on the physical disk.

When preparing a diskette, it is not worthwhile to create a virtual disk information table on the diskette. Diskettes are so small that the system areas would use up too much space. Typically, you create a single file system on the entire diskette. For more information on preparing diskettes, see Chapter 8.

Establishing bad block mapping

To establish bad block mapping, select the **sysadm** operation `Device -> Disk -> Physical -> Soft Format -> Mapping`.

IMPORTANT: You do not need to map highly reliable devices, such as RAID mirrors or NVRAM boards.

The system uses a bad block remap area to store known good blocks to replace blocks that go bad elsewhere on the disk. When you create the virtual disk information table on a disk, the operation installs a bad block remap area and bad block table. The default size is generally sufficient.

You must register a disk before you can establish bad block mapping on it.

After establishing bad block mapping on a disk, you may list the blocks that were mapped or unmapped using `Device -> Disk -> Physical -> Bad Blocks`. Go to Chapter 10 for information on tracking bad blocks on a physical disk.

Installing a bootstrap

To install a bootstrap, select the **sysadm** operation `Device -> Disk -> Physical -> Soft Format -> Install Bootstrap`.

A physical disk must have a current bootstrap if you are to boot from the disk. As a matter of course, you should install a bootstrap on the physical disk in case one is required at a later date. It uses only 500 blocks.

You should install bootstraps on any disk from which you intend to boot, whether booting the kernel (**/dgux**), stand-alone **sysadm** (**/usr/stand/sysadm**), or any other bootable image.

Registering a disk

Registering a disk enables access to its virtual disks. You cannot access a virtual disk unless its host disk drive is registered.

You can register a physical disk only if it contains a virtual disk information table, which is used by the system to track virtual disks. You create a virtual disk information table with `Device -> Disk -> Physical -> Soft Format -> Create VDIT`. Diskettes and CD-ROM disks not require registration.

To register a disk, select the **sysadm** operation `Device -> Disk -> Physical -> Register`.

Sysadm prompts for the name of the physical disk to register. You can use the help option (**?**) to list the physical disks that are not registered. You can then select a physical disk from this list. If all configured disks are registered, **sysadm** displays an error message.

If you register a physical disk that contains logical disks, the physical disk will be registered in compatibility mode. Should you

prefer to continue using logical disks in a virtual disk environment, you may do so with restrictions. See Chapter 10 for information on using disks in compatibility mode.

If the physical disk does not exist, is not soft formatted, or if either of its node files (in **/dev/pdsk** or **/dev/rpdsk**) is already open, the register operation will fail. A physical disk's node file may be open if a database management system has opened it for direct access, for example.

To list the registered physical disks, follow this path through **sysadm**. An example of a brief (normal) listing follows:

```
Device -> Disk -> Physical -> List
Physical disk(s): all ↵
Listing style: [normal] ↵
List label:[no] ↵
```

```
Disk name                    State  Reg? Format Total blocks  Free blocks
sd(ncsc(0,7),0,0)              avail   y   vdisks   2780030      1524986
sd(ncsc(0,7),1,0)              avail   y   vdisks   3933040      1516986
```

This display gives the physical disk names: **sd(ncsc(0,7),0,0)** and **sd(ncsc(0,7),1,0)**; state (avail means available: not owned or registered by a different host system); software format (vdisks means virtual disks); total disk blocks; and total disk blocks free. For more information, you can list partitions, as follows.

```
Device -> Disk -> Physical -> List
Physical disk(s): (ncsc(0,7),0,0) ↵
Listing style: partitions
List label:[no]
```

A sample partitions display follows.

```
Disk name                State   Reg? Format Total blocks  Free blocks
sd(ncsc(0,7),0,0)         avail     y  vdisks      1295922        22768

Name                          Role                Address         Size
swap                                                  859       100000
root                                                100859        80000
usr                                                 180859       300000
usr_opt_X11                                         480859       200000
usr_opt_networker                                   680859        50000
udd                                                 730859       300000
usr_opt_gif                                        1030859        60000
usr_local                                          1090859        50000
var_opt_relimon                                    1140859         2500
usr_opt_xdt                                        1143359        50000
<free space>                                       1193359       122563
tmp                                                1315922        25000
var_mail                                           1340922        20000
```

For information on removing a physical disk from a configuration (deregistering), see Chapter 10 for details.

End of Chapter

7 Creating virtual disks

You have planned your file system's layout, and formatted and registered physical disks in preparation for creating virtual disks and DG/UX file systems. A virtual disk organizes a physical disk into discrete units to contain your software needs. A file system is a directory structure you impose on a virtual disk to organize files and directories that you create and use.

This chapter covers creating three types of virtual disks: ordinary, software mirrored and cached.

Creating a virtual disk

You typically create a virtual disk on a physical disk. However, you generally do not create virtual disks on read-only devices (magneto-optical disk drives, CD-ROMs, and memory file systems) and diskettes. (Refer to Table 4–2 for a review of device requirements.)

To create a virtual disk, use the procedures presented in this section (or you can use a shortcut method when you create a file system, explained later in this chapter). Refer to your Virtual Disk Planning Worksheets for the following information:

- Virtual disk name
- Whether or not you want software disk striping, and stripe size, if you decide to stripe
- Names of disks to hold the virtual disk, unless you plan to let **sysadm** choose the physical disk
- Size of virtual disk partitions in blocks
- Whether the virtual disk will be a mirror of multiple images (for software mirroring), a striped virtual disk, and/or an aggregation of multiple pieces

You can create a virtual disk by any of the following methods:

- Creating a virtual disk by size alone. Use this when you do not care which physical disks the space comes from.
- Creating a virtual disk by physical disks to partition. Use this when you do care which physical disks the space comes from; for example, when you create virtual disk images to mirror or you create a striped virtual disk.

- Creating a virtual disk by name of existing virtual disks (an aggregation). Use this when you have already created one or more virtual disks (generally partitions) and want to combine them into a larger aggregation virtual disk.
- Creating a mirror virtual disk. Use this when you want to software mirror existing virtual disks.
- Creating a cache. Use this when you want to create a software cache for an existing virtual disk.

Proceed to the section for the kind of virtual disk you want.

Creating a virtual disk by size alone

Use this method when you don't care which physical disks the space comes from. Do not use it to create a virtual disk that will serve as part of a software mirror, striped disk, or cache. For those, go to the section "Creating a virtual disk by physical disk(s) to partition."

▶ To create a virtual disk by size alone, follow this path through **sysadm**:

```
Device -> Disk -> Virtual -> Create
```

Type ? for help at any **sysadm** prompt.

1. Supply the name of the virtual disk to create.

   This name will identify the virtual disk whose mount point is listed in directory **/dev/dsk**. After you create and mount a file system on the virtual disk, users will access the virtual disk's file system by the name of the mount point directory. The name should be descriptive and memorable; for example,

   ```
   New Virtual Disk Name: test ↵
   ```

2. Since you will not stripe this disk, press Enter for the default, no. (If you want to create a striped disk, skip to the section about creating a virtual disk by physical disks to partition.)

   ```
   Striped? [no] ↵
   ```

3. You can have **sysadm** create a file system now or later. Generally, it's easier to have the file system created here. Choose the course you want and specify it; for example,

   ```
   Create File System? [yes] ↵
   Select Space by: [Size alone]
   ```

4. Press Enter for the default, size alone.

   ```
   Select Space by: [Size alone] ↵
   ```

5. Enter the size for the new virtual disk, as a decimal number of 512-byte disk blocks.

   ```
   Size in Blocks: (1-n) [1]
   ```

   Sysadm will take space from available physical disks. You can enlarge or shrink the virtual disk later. For example, to create a virtual disk of 20,000 blocks (about 10 Mbytes):

   ```
   Size in Blocks: (1-n) 20000 ↵
   Virtual disk "test" created.
   Virtual disk "test" made a volume.
   ```

   You have created a virtual disk by size alone. You may create another virtual disk or proceed to establish a mount point for the disk just created. This procedure is referred to as "adding a file system," explained in the next chapter.

Creating a virtual disk on specified physical disks

Use this method to specify which physical disks the space comes from; for example, to create the virtual disk images of a software mirror or to create a striped virtual disk. You should also use this method to create the virtual disks for a cache: both for the back-end device on a disk and the front-end disk, either on a NVRAM board or a fast disk. Recall that cache devices must also contain virtual disks.

▶ To create a virtual disk by on specified physical disks, follow this path through **sysadm**:

```
Device -> Disk -> Virtual -> Create
```

1. Supply the name of the virtual disk to create.

   This name will identify the virtual disk whose mount point is listed in directory **/dev/dsk**. After you create and mount a file system on the virtual disk, users will access the virtual disk's file system by the name of the mount point directory. The name should be descriptive and memorable.

   If you are creating an image of a mirror, you might consider appending an image-identifying suffix to the name (for example, **.image***n*) to the name you choose. For example, you might use **pdd.image1** to identify the first virtual disk image and **pdd.image2** to identify the second virtual disk image.

   An example follows:

   ```
   New Virtual Disk Name: pdd ↵
   ```

2. To software stripe the virtual disk, enter **yes** and continue with this step. If you don't want the new disk to be striped, enter **no** and skip to step 4. For example, if you don't want software striping,

   ```
   Striped? [no] ↵
   ```

3. **Sysadm** prompts the stripe size: the number of blocks that are used on one piece of a striped virtual disk, before going on to the next piece.

   The stripe size must be an integral divisor of the number of blocks in each piece. Generally, use the default. If you specify the size, remember to make the virtual disk size an integer multiple of that size. For example,

   ```
   Stripe Size (in blocks): [16] ↵
   ```

4. Decide whether or not to create a file system.

   With the exception of virtual disks used by database management applications, each virtual disk needs a file system before you can use it to store files. You can have **sysadm** create a file system now or later (explained in Chapter 8). Generally, it's easier to create the file system now.

   If you are creating an additional image for a virtual disk that has a file system and user data on it, you can, but need not, create a file system. Later synchronization will copy the existing virtual disk's file system and data to the image you are creating.

   For a virtual disk to be used with a database management application, you very likely do not need to create a file system on it. Consult your application's documentation first.

   Choose the course you want and specify it; for example,

   ```
   Create File System? [yes] ↵
   ```

5. Your answer to this prompt must be "Disk to partition and partition size." You can specify this by a unique abbreviation (such as **Disk**) or if the default is "Disk to partition and partition size" by taking the default. For example,

   ```
   Select Space by: [Disk to Partition and partition
      size] Disk ↵
   ```

6. Specify the physical disk from which to create the new virtual disk or piece.

   IMPORTANT: If you are creating the second or third image of a mirror, or the second or subsequent stripe of a striped disk, then the physical disk you specify should differ from any disk you specified earlier for this mirrored or striped disk.

   If you are creating a virtual disk that will be the front end of a software cache, you must supply its name. Typing ? will list configured devices from which to choose. A configured NVRAM board will show up as **nvrd()** in the list.

Enter ? to get a listing of disks. For example,

```
Disk to Partition From: ? ↵
 1  sd(ncsc(0,7),0,0)
 2  sd(ncsc(0,7),1,0)
 3  sd(ncsc(0,7),2,0)
 4  nvrd()
```

You can specify the physical disk by its number or disk drive name. For example, to specify disk **sd(ncsc(0,7),1,0)** from the list above enter 2 and press Enter:

```
Disk to Partition From: 2 ↵
```

7. You must specify the size in 512-byte disk blocks. The number *n* represents all available space left on the physical disk. You can specify *n* if you want the new virtual disk to occupy all space remaining on the physical disk.

   If the new virtual disk will be just one partition, your answer will set the total space allotment for the virtual disk. The same is true if you're creating one image of a software mirror: even though you will create another image later, the size you specify will set the usable storage size of the mirror virtual disk. For any virtual disk except a striped disk, you may enlarge or shrink the size, as desired.

   If you are creating a striped virtual disk, the size you enter here will be multiplied by the number of pieces you specify to yield the total storage for the disk. For a striped disk, make sure the number you enter is an even multiple of the stripe size.

   If you are creating the second or third image of a mirror, then the length you specify must match the length you specified earlier for the previous image(s).

   For example, to create a virtual disk of 204,800 blocks (about 100 Mbytes), enter the number of blocks and press Enter:

   ```
   Length of Piece in Blocks (1-n) 204800 ↵
   ```

8. Specify the starting address of the virtual disk on the physical disk. Unless you have computed your own disk layout for some special purpose, use the default:

   ```
   Starting Block (optional): ↵
   ```

9. To specify another piece (for example, to create a striped disk and specify a stripe on a different physical disk), answer **yes**. **Sysadm** prompts for the disk to use; return to step 6.

If this virtual disk has all the space you want for it (the virtual disk or an image is complete) enter **no**. **Sysadm** creates the new virtual disk and displays confirmation messages. For example,

```
Do you want to specify more pieces for this virtual
   disk? [yes] no ↵
Virtual disk "pdd" created.
Virtual disk "pdd" made a volume.
File system created on virtual disk /dev/dsk/pdd
```

You have created a virtual disk by partitioning specific physical disks. If you need to create an additional image of a mirror, return to step 1. If you want to create another type of virtual disk, go to the appropriate section. You will need to create its mount point, or add it. See Chapter 8.

Example — Creating a virtual disk on specified physical disk(s)

The following example shows creation of two identically sized virtual disks that could be used for mirroring. It does not create the mirror; to create the mirror, see a later section.

```
Device -> Disk -> Virtual -> Create
New Virtual Disk Name: dailies1 ↵
Striped? [no] ↵
Create File System? [yes] ↵
Select Space by: [Disk to partition and partition size] ↵
Disk to Partition From: ? ↵
 1  sd(ncsc(0,7),0,0)
 2  sd(ncsc(0,7),1,0)
 3  sd(ncsc(0,7),2,0)
Disk to Partition From: 2↵
Length of Piece in Blocks (1-n) 1516986 ↵
Starting Block (optional): ↵
Do you want to specify more pieces for this virtual
   disk? [yes] no ↵
Virtual disk "dailies1" created.
Virtual disk "dailies1" made a volume.
File system created on /dev/dsk/dailies1
```

```
Device -> Disk -> Virtual -> Create ->
New Virtual Disk Name: dailies2 ↵
Striped? [no] ↵
Create File System? [yes] ↵
Select Space by: [Disk to partition and partition size] ↵
Disk to Partition From: ? ↵
 1  sd(ncsc(0,7),0,0)
 2  sd(ncsc(0,7),1,0)
 3  sd(ncsc(0,7),2,0)
Disk to Partition From: 3↵
Length of Piece in Blocks (1-n) 1516986 ↵
Starting Block (optional): ↵
Do you want to specify more pieces for this virtual
   disk? [yes] no ↵
Virtual disk "dailies2" created.
Virtual disk "dailies2" made a volume.
File system created on /dev/dsk/dailies2
```

Creating a virtual disk from existing virtual disks (an aggregation)

Use this method when you want to create an aggregation of existing virtual disks. The virtual disks can be of any type: stripe, mirror, cache, or none of these. To create an aggregation, follow these steps.

IMPORTANT: Creating an aggregation renders useless any data currently residing on the virtual disks you specify.

▶ To create a virtual disk from existing virtual disks, follow this path through **sysadm**:

```
Device -> Disk -> Virtual -> Create
```

1. Supply the name of the aggregation virtual disk to create.

   This name will identify the virtual disk whose mount point is listed in directory **/dev/dsk**. After you create and mount a file system on the virtual disk, users will access the virtual disk's file system by the name of the mount point directory. The name should be descriptive and memorable.

   An example follows:

   ```
   New Virtual Disk Name: temp_storage ↵
   ```

2. To software stripe the aggregation virtual disk, enter **yes** and continue with this step. If you don't want the new disk to be striped, enter **no** and skip to step 4. For example, if you don't want software striping,

   ```
   Striped? [no] ↵
   ```

3. **Sysadm** prompts the stripe size: the number of blocks that are used on one piece of a striped virtual disk, before going on to the next piece.

   The stripe size must be an integral divisor of the number of blocks in each piece. Generally, use the default. If you specify the size, remember to make the virtual disk size an integer multiple of that size. For example,

   ```
   Stripe Size (in blocks): [16] ↵
   ```

4. Decide whether or not to create a file system.

   With the exception of virtual disks used by database management applications, each virtual disk needs a file system before you can use it to store files. You can have **sysadm** create a file system now or later (explained in Chapter 8). Generally, it's easier to create the file system now.

   Choose the course you want and specify it; for example,

   ```
   Create File System? [yes] ↵ HERE
   ```

5. Specify the method for selecting space. To create an aggregation, choose "Name of an existing virtual disk" by entering "Name" and pressing Enter.

   ```
   Select Space by: [Name of an existing virtual disk]
     Name ↵
   ```

6. **Sysadm** asks for the name of a virtual disk to build into the aggregation. If the names of valid choices are not on display (as with the ASCII **sysadm**) enter **?** for a list; for example,

   ```
   Child Virtual Disk:  ? ↵
   ...
       11  local
       12  mail
       13  temp
       14  temp1
   ```

   Select a virtual disk to become part of the new aggregation by entering its number. For example, to pick **temp** from the preceding list,

   ```
   Child Virtual Disk: 13 ↵
   ```

7. You can specify another virtual disk or create a partition to be part of the aggregation. If you enter **yes**, **sysadm** prompts for the child virtual disk; return to step 6.

   After you have specified all the virtual disks you want and answered **no** to this prompt, **sysadm** creates the new virtual disk and displays confirmation messages. For example,

```
Do you want to specify more pieces for this virtual
   disk? [yes] no ↵
Virtual disk "temp_storage" created.
Virtual disk "temp_storage" made a volume.
File system created on virtual disk
    /dev/dsk/temp_storage
```

   You have created an aggregation virtual disk by specifying two or more virtual disks and/or partitions. If you want to create another virtual disk, go to the appropriate section. You still must create its mount point (or add it). See Chapter 8.

Example — Creating an aggregation from existing partitions

The following example shows creation of an aggregation of two existing virtual disks.

```
Device -> Disk -> Virtual -> Create
New Virtual Disk Name: temp_storage ↵
Striped? [no] ↵
Create File System? [yes] ↵
Select Space by: [Name of existing virtual disk]↵
Child Virtual Disk:  ? ↵
...
    11  local
    12  mail
    13  temp
    14  temp1
Child Virtual Disk:  13 ↵
Do you want to specify more pieces for this virtual
   disk? [yes] ↵
Child Virtual Disk:  ? ↵
...
    11  local
    12  mail
    13  temp
    14  temp1
Child Virtual Disk:  14 ↵
Do you want to specify more pieces for this virtual
   disk? [yes] no ↵
Virtual disk "temp_storage" created.
Virtual disk "temp_storage" made a volume.
File system created on virtual disk
    /dev/dsk/temp_storage
```

Example — Creating an aggregation from dynamically created partitions

The following example shows the creation of an aggregation from two virtual disks that you create dynamically. Creating an aggregation in this way produces two unnamed child partitions.

```
Device -> Disk -> Virtual
New Virtual Disk Name: zenith↵
Striped? [no] yes↵
Stripe Size (in blocks): [16]↵
Create file system? [yes]↵
Select Space by: [Disk to partition and partition size]↵
Disk to Partition From: sd(insc(0),1,0)↵
Length of Piece in Blocks: (1-1290279) 30000↵
Starting Block (optional):↵
Do you want to specify more pieces for this virtual disk? [no] yes ↵
Select Space by: [Disk to partition and partition size]↵
Disk to Partition From: sd(insc(0),5,0)↵
Length of Piece in Blocks: (1-1290279) 30000↵
Starting Block (optional):↵
Do you want to specify more pieces for this virtual disk? [no]↵
30000-block unnamed child partition created at 5627 on
"sd(insc(0),1,0)"
30000-block unnamed child partition created at 35627 on
"sd(insc(0),5,0)"
Virtual disk "zenith" created.
Virtual disk "zenith" made a volume.
File system created on /dev/dsk/zenith
```

You created a striped aggregation named **zenith** that contains two unnamed partitions on fast disks **sd(insc(0)1,0)** and **sd(insc(0),5,0)**. Each partition is 30,000 blocks.

Note that the second set of partitions created was unnamed. In general, having unnamed partitions does not pose a problem as long as you access the parent aggregated virtual disk. However, to access directly a child virtual disk, you must use its long name, represented as **vdm**(*x*,*y*,*z*), where *x* is a generation number, *y* is the system identifier, and *z* is a duplicate number. You may list unnamed child partitions using the list operation. You must name a virtual disk before you can manipulate it. You can assign a name to an unnamed virtual disk in the **sysadm** operation `Device -> Disk -> Virtual -> Rename`. (See Chapter 10.)

Creating a mirrored virtual disk

This section explains how to create a software mirror from two or three existing virtual disks. Software mirroring helps to protect against data inaccessibility (due to a failed disk drive) by duplicating the contents of a virtual disk on one or more mirror images.

A software mirror consists of two or three virtual disks, each the same size, and each on a different physical disk drive. A mirror has

a name (for example, **pdd**), that may differ from its virtual disk image names (for example, **pdd.image1** and **pdd.image2**). A virtual disk must already exist for each image you want to specify.

Follow this outline with **sysadm** to build and synchronize a software disk mirror. You can execute these steps at any time to add an image to an existing mirror.

1. Create (or have available) two or three virtual disks to form images of the mirror (explained in a previous section). If you have a virtual disk with a file system and data on it, you need only one (identically sized) virtual disk to serve as the other image.
2. Build the mirrored virtual as described in the following section.

   If no image has a file system on it, on the primary image tell **sysadm** to create a file system; then later tell **sysadm** to synchronize the images. (Creating a file system is not necessary if the mirror will hold a database management system that will create its own file system.)

   If you are creating a second image for a virtual disk that has a file system and user data on it, do *not* tell **sysadm** to create a file system, but *do* tell **sysadm** to synchronize the images. After **sysadm** creates the second image, use **sysadm** to synchronize using the primary image as a source. This will copy the disk with user data to the new image.
3. Create a mount point for the local file system as explained in Chapter 8.

Building the mirrored virtual disk

▶ To create a mirror from two or three existing virtual disks, follow this path through **sysadm**:

```
Device -> Disk -> Virtual -> Mirrors -> Mirror
```

1. **Sysadm** prompts for the name of the existing virtual disk you want to mirror. As always, you can enter **?** for a list of valid answers. For example, if you earlier created a virtual disk named **pdd.image1** as a primary image for the mirror,

   ```
   Virtual Disk: pdd.image1 ↵
   ```
2. Enter the number of images of the mirror virtual disk that must be available and synchronized before you can mount the disk.

   If the value is 1, then the mirror virtual disk will be mountable and available to users as soon as the first image is present. If the value is more than one, the mirror will not be mountable and will not be available to users until the specified number of images are available and synchronized. Generally, one image is enough. For example,

   ```
   Minimum Images Required: (1-3) [1] 1 ↵
   ```

3. Specify how many lost images the system will allow to mount the mirror. (By Lost, **sysadm** means broken; that is, an unsynchronized image does not count as lost.)

   A value of 0 means that if an image is lost, DG/UX will not mount any of the mirror images. Generally, you should specify one less than the number of images, giving users access to the mirror's information if one image fails. You can then fix the problem and start synchronization of the new image. If your application requires that a backup image must always be available, you might want to specify 0 (for a two image mirror) or 1 (for a three-image mirror). For example,

   ```
   Maximum Number of Lost Images to Tolerate: (0-3)[0] 1↵
   ```

4. Decide whether or not to enable automatic synchronization at system startup.

   If you select automatic synchronization, at startup the DG/UX system will start synchronizing images of this mirror that are unsynchronized. Having the images synchronized is desirable; if you do not select automatic synchronization, DG/UX will not synchronize the images until you direct it to do so using **sysadm**. A disadvantage of synchronization can be slower response time throughout the system; also, if you answer no, you (not the system) can control which image is the master.

   ```
   Automatically Synchronize on System Boot? [yes]↵
   Throttle Value in Milliseconds: (0-300) [0]
   ```

5. Specify how many milliseconds the system will wait between succeeding I/O operations during synchronization. The default value lets I/O proceed at full speed, which might slow users' access to the good disk. A non-zero value will let time elapse between I/Os and might reduce contention for the good disk. For example,

   ```
   Throttle Value in Milliseconds: (0-300) [0] ↵
   ```

6. Specify the name of the mirror.

   As mentioned earlier, the mirror name should differ from the name of its constituent virtual disk images. The default name is the name of the virtual disk you specified in step 1. Depending on the name of that virtual disk, you might want to specify a different name for the mirror; if not, take the default. For example,

   ```
   New Name for Mirror: [pdd.image1] pdd ↵
   ```

7. Next, specify a different name for the virtual disk *image* you specified in step 1. You might want to enter a name that distinguishes the image from the mirror and from other images. For example,

   ```
   New Name for Image: pdd.image1 ↵
   ```

8. **Sysadm** prompts for another image. If you have specified all the virtual disk images you want (at least two), enter **no**; skip to step 10.

   If you want to specify another virtual disk image, enter **yes** or press Enter and continue with the next step. For example,

   ```
   Another image? [yes] ↵
   ```

9. Specify the name of the second (or third) virtual disk image. You can enter **?** for a list of valid answers. For example, if you earlier created a virtual disk named **pdd.image2** as a secondary image for the mirror,

   ```
   Child Virtual Disk: pdd.image2 ↵
   ```

10. Decide whether or not to synchronize the mirror images.

    Having the images synchronized will let you mount and use the mirror immediately. Answer **yes** unless you have another image to specify.

    IMPORTANT: The synchronization will copy the image you specified in step 1 to the child virtual disk you specified in step 9. Any information on the child virtual disk will be overwritten. If you think there is any user data on either image, make sure that the image you specified in step 1 is the source you want and the image you specified in step 9 is the destination you want.

    Make your choice and respond. For example,

    ```
    Begin Sync Immediately? [yes] ↵
    ```

    Sysadm displays confirmation messages like the following:

    ```
    Virtual disk "pdd.image1,2CB42817,0D19AB77,0"
         renamed to "pdd"
    Virtual disk inserted at pdd.
    Child virtual disk made a volume.
    Virtual disk "pdd.image2" linked as a child to
       virtual disk "pdd".
    Synchronization started on virtual disk
       disk mirror "pdd".
    ```

    You have created a mirror from two (or three) virtual disk images. If you want to create another virtual disk, go to the appropriate section. If no image has a file system on it, you will need to create a file system on the mirror, as explained in Chapter 8, and then use **sysadm** to synchronize the images. If you answered **no** to the synchronize prompt, you will need to use **sysadm** to synchronize the images.

After the images are synchronized, the system will maintain them as copies. While synchronized, the images will not be accessible as individual virtual disks. However, you can fracture the mirror with **sysadm** and then access one of the disks (as for backup), and then relink and synchronize. See Chapter 10 for information on synchronizing, unsynchronizing, linking and fracturing.

To add an image to or delete an image from an existing mirror, you may do so with the link and unlink operations described in Chapter 10.

Booting from a mirrored virtual disk

If you mirrored a virtual disk that will be bootable, be sure that you specify the name of the image containing the bootstrap rather than the name of the mirror itself. For example:

```
SCM> b sd(cisc(),0,0)usr_opt_fredware.image1:/fred ↵
```

Example — Creating a mirrored virtual disk

The following example shows the mirroring of the two mirror images created in an earlier section.

```
Device -> Disk -> Virtual -> Mirrors -> Mirror
Virtual Disk: pdd.image1 ↵
Minimum Images Required: (1-3) [1] ↵
Maximum Number of Lost Images to Tolerate:(0-3)[0] 1 ↵
Automatically Synchronize on System Boot? [yes] ↵
Throttle Value in Milliseconds: (0-300) [0] ↵
New Name for Mirror: [pdd.image1] pdd ↵
New Name for Image: pdd.image1 ↵
Another image? [yes] ↵
Child Virtual Disk: pdd.image2 ↵
Begin Sync Immediately? [yes] ↵
Virtual disk "pdd.image1,2CAC4C26,373A2C53,0"
   renamed to "pdd"
Virtual disk inserted at pdd.
Child virtual disk made a volume.
Virtual disk "pdd.image2" linked as a child to
    virtual disk "pdd".
Synchronization started on virtual disk
   disk mirror "pdd"
```

Creating a software disk cache

Software disk caching offers the performance enhancements inherent in a traditional cache for file systems. You create a software disk cache by creating a cached virtual disk.

A software cache can include one or more front ends (multiple NVRAM boards or fast disks on which you have created virtual disks and one or more slower back ends. Each front end and back end is a virtual disk. To create a cache with multiple front ends, you specify all the front-end virtual disks when you create the cache; to create caches with a shared front end, you specify the same front-end virtual disk as you create multiple caches. Creating a cache does not affect any information stored on the back-end virtual disk; therefore you can cache an existing virtual disk without harming information on it.

In the following example, you create a cache by associating a single back-end device with a single front-end device.

Make sure the back-end virtual disk exists. (You create the front-end virtual disk when you create the cache.) If not, create it as explained in "Creating a virtual disk on specified physical disk," earlier.

▶ To create a cache, follow this path through **sysadm**:

```
Device -> Disk -> Virtual -> Caches -> Cache
```

1. Supply the name of the existing virtual disk that you want to use as the back end. For example, if you earlier created a back-end virtual disk named **payroll**,

   ```
   Virtual Disk: payroll ↵
   ```

2. Specify whether or not to enable reading from the cache. Answering yes causes data to be shipped from the back-end device to the cache for reading. Otherwise, data is read directly from the slow back-end device, which is referred to as pass-through mode for reading. Accepting yes accelerates read operations.

   ```
   Cache Reads? [yes] ↵
   ```

3. Specify whether or not to enable writing to the cache when possible. Answering yes causes data to be written directly to the cache, which subsequently sends the data to the back-end. Otherwise, data is written directly to the slow back-end device in pass-through mode. Writing to the cache accelerates write operations.

   ```
   Cache Writes? [yes] ↵
   ```

4. Specify whether or not to cache only file system metadata. Answering **no** selects both the metadata and the data for caching. Answering **yes** selects only the metadata for caching.

   Metadata captures important file system statistics such as file system inodes, size, the date stamp, and owner. If you are caching on a raw device, there is no file system structure in place, and consequently, there is no file system metadata.

   Answer **yes** to caching only metadata under these circumstances:

   - A database manager application is responsible for regulating its own data transmissions. Allowing the DG/UX system to perform the caching function would be superfluous and probably ineffective.
   - NFS servers using the cache over a LAN will also regulate their own buffering needs. Data received over a LAN would likely flood a cache's buffers. Selection of caching only the metadata would be preferred in these instances. File system data would be read and written in pass-through mode, bypassing the front-end entirely.

   Answer **no** to caching only metadata under these circumstances:

   - Caching both the file system data and its metadata would be desired for a local file system.
   - If the cache is to contain a non-DG/UX file system, such as a database, do not elect to cache file system metadata only. Such a selection would prevent any data caching.

   ```
   Cache Only File System Metadata? [no] ↵
   ```

5. The asynchronous write policy is meaningful only when caching writes are being cached. You may request direct writes from the front-end to the back-end device immediately after caching the data under certain conditions:

`Never`	The system will never write data directly to the back-end. The advantage is that no redundant disk activity takes place, but the disadvantage is that cache performance may degrade if the front-end gets full.
`All writes`	Any time the system writes a buffer to the cache, an asynchronous I/O flushes the buffer to the back-end device. This setting may improve performance in caches doing more reads than writes. A disadvantage of this operation is that substantial redundant disk activity may take place.

`First write`	The first time the system writes a new buffer to the cache, an asynchronous I/O will flush it to the back-end device. This setting helps to keep the cache clean when there are many buffers that are written only once such as for large sequential writes. However, it minimizes disk activity for those blocks that are written repeatedly. In general, "first write" is the best policy.

```
Asynchronous Write Policy: [First write] ↵
```

6. Releases prior to DG/UX 5.4R3.00 referred to read weight as read retention rate. The read weight is meaningful only if you cache read operations.

   The read weight parameter assigns a relative priority to read operations that occur in a front-end device. A larger weight gives higher priority to the specific operation. When the cache is being searched for available space, buffers with lower weights are reused first. Thus, buffers with a high weight remain in the cache longer.

   If multiple cache virtual disks share a front-end device, you assign a higher priority to one cache by increasing its read weight value. Conversely, you may assign a lower priority to one or more remaining caches by decreasing their read weight values. The maximum read weight is 1000.

   ```
   Read Weight: [1] ↵
   ```

7. Releases prior to DG/UX 5.4R3.00 referred to write weight as write retention rate. The write weight is meaningful only if you cache write operations.

   The write weight parameter assigns relative priority to write operations occurring in a front-end device. A larger weight gives higher priority to the specific operation. When the cache is being searched for available space, buffers with lower weights are reused first. Thus, buffers with high weights remain in the cache longer.

   If multiple cache virtual disks share a front-end device, you assign a higher priority to one cache by increasing its write weight value. Conversely, you may assign a lower priority to one or more remaining caches by decreasing their write weight values. The maximum write weight is 1000.

   ```
   Write Weight: [1] ↵
   Search percentage: (0-100) [10] ?
   ```

8. Specify the percentage of the cache's front-end for the system to search when looking for a clean buffer or data block with the lowest weight. If no clean buffer is found, a dirty buffer with the lowest weight is flushed and reused. On the next search for a clean buffer,

the search will begin where the last search ended, so eventually the entire cache will be searched. A value of zero causes the next buffer to be used. By default, when creating a cache, the search percentage is 10 percent.

Ideally, a cached disk improves performance through much faster memory access. There are two obstacles, however, that prevent disk caching from reaching this ideal level of performance:

- The cache device is not large enough to contain all the data that applications require. Therefore, some I/O requests must access the back-end device for data not currently in cache. The ratio of I/O requests satisfied by the front end cache (cache hits) to the total number of I/O requests is called the cache hit rate. You want the cache hit rate to be as high as possible.

- If the buffer that your application needs to read or write is not in the cache, or those available are dirty, you must wait for the data to be flushed. This waiting period is referred to as a *stall*, which unnecessarily degrades system performance. Increasing the search percentage will give the flusher a greater amount of buffer to search through when seeking dirty buffers to flush. Locating dirty buffers quicker will cause more frequent flushes, which reduces stall times. You want stalls to occur as seldom as possible. Increasing the search percentage will bring down the occurrence of stalls.

By experimenting with the parameters, you can find ways to maximize the cache hit rate and minimize the frequency of stalls for your cached disk. Once you have created the cached virtual disk, use `Device -> Disk -> Virtual -> Cache -> List` to view performance statistics, and use `Device -> Disk -> Virtual -> Cache -> Modify` to adjust the operating parameters. (These operations are documented in Chapter 10.) Also you may use the **nsar** command and AV/SysScope™ performance monitor to check performance statistics. Refer to the **nsar** (1M) manual page and the *Using the AV/SysScope™ Performance Monitor* for more details.

```
Search percentage: (0-100) [10] ? ↵
```

9. Decide whether or not to implement cyclic flushing.

   If you cache reads only, do not use the flusher. If you cache writes only, use the flusher. In a cache used for writing as well as reading, the cache must at some point write, or flush, the data in its buffers to disk. If your application accesses the cache during a flush, or if your application causes a flush, it will have to wait, or stall, until the flush completes. You want to minimize stalls. A cyclic flush occurs on a regular basis.

   ```
   Flusher Type: [Cyclic] ? ↵
   ```

10. Specify the name for the cache. Normally, the cache assumes the name of the virtual disk that is being cached (for example, **payroll**), and the virtual disk being cached is given a new name (next prompt). However, you may give the cache and the virtual disk on the back-end device any names you please, as long as they are unique.

    ```
    Name for Cache: [payroll] ↵
    ```

11. Specify the new name for the cache back end. Press Enter to accept the default, which is its former name appended with **.be**. You are renaming the virtual disk that is being cached. Typically, the name of the virtual disk being cached travels with the cache, and you must select a new name to associate with the back-end device. Alternatively, you may select a new name for the cache itself. You may assign the cache and the back end any names you please, as long as they are unique.

    ```
    New Name for Back End: [payroll.be] ↵
    ```

12. Specify the name of the virtual disk to serve as a fast, front-end device.

    ```
    Front End Virtual Disk: payroll.fe ↵
    ```

    Assuming you have already created a virtual disk on the desired front-end device, such as a NVRAM board, supply its name. Skip the next step, and go to step 14.

    If, however, you do not have a virtual disk for use as a front-end, you may create it now.

13. Create a virtual disk to use as a front end by partitioning an existing virtual disk or a physical disk or by using an existing virtual disk. Type ? for a list of choices. For a recap of ways to create a virtual disk, refer to the appropriate sections in this chapter.

    ```
    Select space by:  [Disk to partition and partition
    size]
    ```

    Create the virtual disk using the desired method. Refer to Chapter 2 for information on sizing the front-end device.

14. You can specify as many front-end virtual disks as you want. For example, if you have multiple virtual disks consisting of NVRAM boards, you can specify each of these. At runtime, DG/UX will treat all the front ends as pool of cache space. When you have no more front ends to specify for this cache, enter **no**. For example,

    ```
    Specify Another Front End? [yes] no ↵
    Virtual disk inserted at payroll.
    Child virtual disk made a volume.
    ```

    Sysadm has created the cache (in this example, **payroll**).

NVRAM caching and continuous data access: a typical scenario

Suppose you have a file system on a local AViiON server which you later decide to export over the network. The drawback of exported file system I/O is its slow speed. One of the principal causes of slowness of remotely mounted file systems is delayed data write confirmations. To maintain data integrity, ONC/NFS does not acknowledge a successful write operation until the data is actually written to the remote physical device.

Caching is one answer to reducing write confirmation delays. With normal operating system circumstances (e.g., sufficient CPU), the ability to create a cache dynamically encourages you to experiment with your configuration without the inconvenience of downtime. Through experimentation, you decide whether or not to cache the virtual disk.

Assume that you have a partition named **notes** that resides on a slow, back-end device for which you want to create a fast front-end device. The following table identifies the goals for this cache.

Cache Name	Back End	Size in blocks	Front End	Size in blocks
notes	notes–be	100,000	notes–fe	3,967

A typical **sysadm** session to cache virtual disk **notes** follows.

The cache, by convention, assumes the name of the virtual disk being cached. When you create the cache for the virtual disk, the cache will inherit the name of the back-end virtual disk, and the back-end virtual disk assumes a new name. For example, when caching the **notes** virtual disk, the cache will inherit the virtual disk's name, **notes**, and the back-end will assume a new name, such as **notes–be**. Naturally, the front-end of the **notes** cache would be **notes–fe**.

To create virtual disks to serve as a back end and front end, perform twice the **sysadm** `Device -> Disk -> Virtual -> Create` operation.

To build a cache from the existing virtual disks, select the **sysadm** operation `Device -> Disk -> Virtual -> Caches -> Cache`.

End of Chapter

8 Creating and mounting file systems

This chapter provides instructions for:

- Creating a DG/UX file system
- Creating a mount point for the file system
- Exporting a file system (making a file system accessible to users on remote hosts connected to a network)

File system table (/etc/fstab)

When you create a mount point for a file system, the system records it automatically in the **/etc/fstab** file, making the file system accessible for use. At boot time, the DG/UX file system mounts all files listed in **/etc/fstab**. If the DG/UX system cannot mount a remote file system (because of network problems, for example), you cannot use that file system.

Figure 8–1 shows an excerpt of a typical **fstab** file:

```
# Files systems mounted from local disk
/dev/dsk/root  /  dg/ux  rw  x  0
/dev/dsk/swap  swap_area  swap  sw  x 0
/dev/dsk/usr /usr dg/ux rw x 0
/dev/dsk/usr_opt_X11 /usr/opt/X11 dg/ux rw x 1
/dev/dsk/usr_opt_pkg /usr/opt/pkg dg/ux rw d 1
/dev/pdsk/3 /usr/opt/floppy dos rw 0 0
/dev/pdsk/4 /usr/opt/docbrowse cdrom ro 0 0
#
# File systems mounted from remote disk
dot:/pdd/sam/image /pdd/pics/image nfs rw,hard 0 0
oz:/pdd/notes     /pdd/notes      nfs rw,hard 0 0
kansas:/pdd/otis/papers /udd/conf/papers nfs rw,hard 0 0
toto:/pdd/spleen/games /pdd/spleen/notes nfs rw,hard 0 0
```

Figure 8–1 Excerpt of /etc/fstab file

Comments (signified by the leading # sign) in the preceding **/etc/fstab** file designate the local and remote file mounts. Each entry contains fields that provide important file system attributes, which the DG/UX system uses to manage them. A typical entry for a local file system follows with an explanation of each field. For a complete explanation of file types and associated legal field values, see the file list operation in Chapter 11.

```
dot:/pdd/sam/image /pdd/pics/image nfs rw,hard x 0 0
```

An explanation of each field follows:

`dot:/dev/sam/image`	File system source
`/pdd/pics/image`	Local mount point
`nfs`	File system type
`rw` (readable and writable)	Access
`hard` (applies to remote mounts only)	NFS mount
`x` (backup frequency)	Do not back up
0 (range from 0-9)	File system checker pass

The **fstab** file contains information for commands that mount, unmount, backup, restore, and check file systems.

To create a mount point, you may use a **sysadm** operation or a shell command, or you may edit the **/etc/fstab** file.

Does my device need a DG/UX file system?

The operation you perform to create a file system and a mount point depends on how you use the particular device or application. Table 8–1 lists each type of device and how it can be used.

Table 8–1 DG/UX file system requirements for disk device types

Disk Device Type	Create DG/UX File System?	Go To
Winchester disk	Yes	"Creating a DG/UX file system and mount point"
Rewritable Magneto Optical and Write Once, Read Many (WORM)	Refer to your hardware documentation or application documentation, if applicable.	
NVRAM	No	"Creating a file system mount point"
Diskette	Contains virtual disks?	"Creating a DG/UX file system and mount point"
	Already DOS formatted?	"Creating a file system mount point"
	Want to create a single file system	"Creating a single file system on a diskette"
CD–ROM	No	"Creating a file system mount point"
Memory file system	No	"Creating a file system mount point"
Databases	Depends	Your database documentation

Use your planning worksheets while performing the file system operations in this chapter.

Creating a DG/UX file system and mount point

This section describes how to create a DG/UX file system and its mount point. This operation implicitly creates a virtual disk if you have not already done so.

Use these procedures only under these circumstances:

- You have a Winchester disk and you do not care on what physical disk the file system is located. A virtual disk is implicitly created.
- You have a Winchester disk on which you have already created a virtual disk but no DG/UX file system.

- You wish to put multiple DG/UX file systems on a diskette to use with the DG/UX system.
- Your database management application requires both virtual disks and DG/UX file systems.

▶ To create a file system, follow this path through **sysadm**:

```
File System -> Local Filesys -> Create
```

1. **Sysadm** prompts for the name of the virtual disk on which you want to create the file system. With ASCII sysadm, enter ? to display a list of virtual disks.

   If the virtual disk you specify does not exist, **sysadm** will create it, prompting you for the desired method to create a virtual disk.

   ```
   Virtual Disk: []: pdd ↵
   ```

2. **Sysadm** asks for custom options you want to add to the **mkfs** command that will create the file system. You can access the shell (for example with **!csh**) and look at the **mkfs** man page for available options. Generally, you do not need any options, in which case press Enter:

   ```
   Mkfs options: ↵
   ```

3. Specify the mount directory where the DG/UX system mounts the file system at system startup. The mount directory you specify here need not exist; **sysadm** will create it for you later on. This forms the pathname through which users will access the file system, therefore it is very important.

   Specify the mount directory's full pathname from the root directory. Provide the virtual disk name as the last entry in the mount point. For example,

   ```
   Mount Directory: /pdd ↵
   ```

4. Specify whether or not the file system is exportable.

   An exportable file system allows other hosts connected to the network to mount it on their own directories. Making a file system exportable creates an entry in the **/etc/exports** file. This is a value you can change later on using Modify, without harm to the file system.

   If you do not want the file system to be exportable, answer **n** (or press Enter) and skip to step 6. If you want to export the file system, answer yes (**y**) and specify export options.

   Sysadm prompts

   ```
   Export options?
   ```

   Export options restrict the access to an exported file system. The export options are as follows in Table 8–2.

Table 8–2 Export options

Option	Meaning
secure	Requires OS clients to use a secure protocol when accessing the directory.
ro	Exports the directory as read-only. Read-write is the default.
rw=hostname[,*hostname*]...	Host systems specified in this option have read-write access to the directory. Does not override normal file and directory permissions.
anon=*uid*	Sets an effective user ID for anonymous users. The default anonymous user ID is **-2**. The value **-1** disables anonymous access.
root=hostname[,*hostname*]...	Gives **root** (superuser) access only to superusers from the specified computer systems. By default, superusers from other computer systems do not have superuser access to the directory. A superuser is any user whose user ID is **0** (**root** and **sysadm**, for example).
access=client[,hostname]...	Gives mount access to each host listed. A host can be represented as *hostname* or *netgroup*; see the **netgroup**(5) manual page. Each host in the list is first sought in the **/etc/netgroup** database and then in the **/etc/hosts** database. The default value allows any computer system to mount the given directory.

5. Specify the export options. Precede the first option with a hyphen (–) and no intervening space, and list of the export options separated by commas as shown in the example. For example,

   ```
   Export options? -rw=accounts,purchasing ↵
   ```

6. **Sysadm** warns you that

   ```
   The Create operation will destroy any data currently
      stored on virtual-disk-name.  Are you sure? [yes]
   ```

   If any file system already exists on this virtual disk and you think there may be useful information in that file system, answer **n** and add (mount) and examine the file system. If you're sure there is no useful information on this virtual disk (or if there has never been a file system on this disk), press Enter to confirm.

   Sysadm creates the file system on the virtual disk, adds the file system to the **fstab** file and mounts the file system. It displays messages like these:

   ```
   File system created on virtual disk /dev/dsk/pdd.
   File system added:  /usr/opt/pdd
   File system mounted: /usr/opt/pdd
   ```

   Repeat these steps or create a mount point for a DG/UX file system using the procedures in the next section for each virtual disk listed in the worksheets.

Creating a mount point for a local file system

Creating a mount point comprises the following steps:

- adding the file system name to the **/etc/fstab** file, so that the file system will be mounted automatically at system startup
- mounting the file system

Sysadm's create operation includes these two steps.

Do not use this operation when you want to add a file system that you have manually unmounted; instead, use the mount sequence that is covered in Chapter 11.

This section explains creating a mount point for:

- DG/UX file system
- CD-ROM
- Diskette
- Memory file system

Creating a mount point for a DG/UX file system

Generally, a DG/UX file system is the type you choose for virtual disks on standard Winchester disk drives, software disk caches and DG/UX file-formatted diskettes. Before you can add a file system, the file system must already exist. If your disk does not have a file system, use create as in the previous section.

▶ To add a file system, follow this path through **sysadm**:

```
File System -> Local Filesys -> Add
```

1. **Sysadm** prompts for a system type. The default is a DG/UX file system. Press Enter.

   ```
   File System Type: [dg/ux] ⤵
   ```

2. Specify the name of the virtual disk that holds the file system. Enter **?** to display a list of virtual disks.

IMPORTANT: If you are adding a file system to a mirror virtual disk, enter the name of the mirror virtual disk, not the name of a virtual disk image.

For example, to add the virtual disk **pdd**:

```
Virtual Disk: pdd ⤵
```

3. Specify the mount directory's full pathname from the root directory. This forms the pathname through which users will access the file system, therefore it is very important. Use the virtual disk name as the last filename in the mount point pathname. For example,

   ```
   Mount Directory: /pdd ↵
   Write Permission: [Read/Write]
   ```

   The mount directory you specify here need not exist. If it does not exist, **sysadm** will offer to create it for you later on.

4. Specify whether or not users can write to the mounted file system. To make the file system read-only, enter **ro**. Take the default by pressing Enter.

   ```
   Write Permission: [Read/Write] ↵
   ```

5. Specify the dump frequency (how often you archive your file system). Type ? to display a list of frequencies. Press Enter to accept the default.

   ```
   Dump Frequency: [Daily] ↵
   ```

   Setting this value does not fully enable archiving. See the **sysadm** `File System -> Backup` operation in *Managing the DG/UX™ System* for more information.

6. Specify the **Fsck** pass number indicating when the DG/UX system checks for corrupted file systems when the system boots.

   Values range from 0 to 9; 0 means the file system is never checked; 1 means it is checked first; and 9 means that it is checked last. See the **fsck**(1M) manual page for more information. Multiple file systems can be checked in one pass. See a Chapter 12 for details on fast-recovery file system checking.

   Generally, you will want the default **fsck** pass number; if so, press Enter. For example,

   ```
   Fsck Pass Number: [1] ↵
   ```

7. Enable or disable file system logging. The log collects file system modifications for a quick recovery of the file system when the system crashes. Logging, however, slows runtime write performance. You are advised to use logging primarily when rapid recovery and high availability are crucial.

   Decide on your answer, **no** (default) or **yes**. If you answer **yes**, you are later asked for the **fsck** log size.

   ```
   Fsck Logging? [no]
   ```

8. Specify whether or not the file system is exportable. If you do not want the file system to be exportable, answer **no** (or press Enter).

   If you want to be able to export the file system, answer **yes** and continue with this step; you will need to specify export options.

```
Exportable? [no] yes ↵
Export options?
```

For the export options, see Table 8–2 .

If you answered **no** to **fsck** logging in step 7, skip to step 10.

9. If you answered **yes** to **fsck** logging, **sysadm** prompts:

   ```
   Fsck Log Size: [64]
   ```

   Specify the log size in 512-byte blocks. Performance varies with log files of different sizes. A large log file improves run time performance but prolongs recovery time. A small log file reduces recovery time but degrades run time performance. Recovery time depends on the size of the log, not the size of the file system.

10. **Sysadm** prompts you to confirm before mounting the file system. The prompt includes the words "and export" if you specified that the file system was exportable. Confirm by pressing Enter at both prompts.

    ```
    Mount [and export] the file system? [yes] ↵
    Ok to perform the operation? [yes] ↵
    ```

11. If the mount directory you specified in step 3 does not exist, **sysadm** prompts:

    ```
    Mount point directory /pdd does not exist.
    Do you wish to create it? [yes]
    ```

    If you enter **no**, **sysadm** aborts the operation and returns to the initial prompt. If you want to create the directory, answer **yes** or press Enter.

    Sysadm displays

    ```
    File system added: /pdd
    File system mounted: /pdd
    File system exported: /pdd     (if it is exportable)
    ```

    You have added and mounted the file system. Repeat this procedure for each virtual disk whose file system you want to add and mount. Consult the planning worksheets.

Creating a mount point for a CD-ROM device

A CD-ROM device is a read-only memory compact disk drive that is formatted in either the High Sierra or ISO 9660 standard. A file system for a CD-ROM is a read-only file system. When adding a file system for a CD-ROM device, you don't need a virtual disk. However, you may want to obtain mount directory information from the planning worksheets.

▶ To add a file system that's on a CD-ROM disk drive, follow these steps:

1. Insert a CD correctly in the CD drive.
2. Follow this path through **sysadm**:

   ```
   File System -> Local Filesys -> Add
   ```

3. Specify the file system type as **cdrom**:

   ```
   File System Type: [dg/ux] cdrom ↵
   ```

4. Enter the complete pathname of the device file that belongs to the CD-ROM disk drive. For example, enter the device file named **pdsk1** in the **/dev** directory:

   ```
   Device file: /dev/pdsk1 ↵
   ```

5. Specify the mount directory's full pathname from the root directory. This will form the pathname through which users will access the file system, therefore it is very important. Use the virtual disk name as the last filename in the mount point pathname. For example,

   ```
   Mount Directory: /usr/resources ↵
   ```

   The mount directory you specify here need not exist; **sysadm** will create it for you later on.

6. Specify whether or not the file system is exportable.

   An exportable file system allows other hosts connected to the network to mount it in their own file systems. CD-ROMs are often used in this way because a CD-ROM offers a large read-only resource.

   If you want to be able to export the file system, answer **yes** and continue with this step; you will need to specify export options.

   For export options, see Table 8–2.

   ```
   Exportable? [no] yes ↵
   Export Options:
   ```

7. **Sysadm** prompts you to confirm before mounting. The prompt includes the words "and export" if you specified that the file system was exportable.

   ```
   Mount [and export] the file system? [yes] ↵
   ```

8. Confirm by pressing Enter at both prompts:

   ```
   Mount [and export] the file system? [yes] ↵
   Ok to perform the operation? [yes] ↵
   ```

9. If the mount directory you specified in step 5 does not exist, **sysadm** prompts

   ```
   Mount point directory /usr/resources does not exist.
   Do you wish to create it? [yes]
   ```

10. If you enter **no**, **sysadm** aborts the operation and returns to the initial prompt. If you want to create the directory, answer **yes** or press Enter.

    Sysadm displays

    ```
    File system added: /usr/resources
    File system mounted: /usr/resources
    File system exported: /usr/resources  (if it is exportable)
    ```

    You have added and mounted a file system for a CD-ROM. Repeat this procedure for each CD-ROM you want to add.

Creating a single file system on a diskette

To maximize a diskette's limited capacity, you may choose not to create virtual disks on it. A virtual disk information table (VDIT) serves as a map for a physical disk's contents and uses 640 blocks. A VDIT is required if you intend to put multiple virtual disks on a physical disk. On a diskette it is more efficient to forgo the VDIT and to create a single file system directly on the physical disk. To create a file system on a diskette, use the **mkfs**(1M) command.

You may create two types of file systems on a diskette: DOS or DG/UX.

DOS file system

IMPORTANT: You do not have to format pre-formatted diskettes. However, you must still create a mount point for it and mount it.

Make sure that the diskette is not write-protected.

Use the following **mkfs** shell command format:

```
% mkfs density "format" device-node-name
```

where:

density is the diskette's capacity, either:

`360kb` or `1220kb` 5.25-inch

`720kb` or `1440kb` 3.5-inch

format is the format required by the **dfm** (diskette file manager) driver, which should be configured in the kernel. Either value is allowed:

`dos`

`pc`

device-node-name is the character node name of the diskette device, which is listed in **/dev/pdsk**. You also may check the **/etc/devlinktab** file, which lists the correlation of DG/UX long names to device node names.

Example:

```
mkfs 360kb "dos" /dev/pdsk/3
```

To make the diskette accessible, you also must create a mount point and then mount the DOS-formatted diskette. Refer to the section on creating a mount point for a DOS-formatted diskette for details.

DG/UX file system

IMPORTANT: You do not have to format pre-formatted diskettes. However, you must still create a mount point and mount it.

Make sure that the diskette is not write-protected.

Use the following **mkfs** shell command format:

```
% mkfs density device-node-name
```

where:

density is the diskette's capacity, either:

`360kb` or `1220kb` 5.25-inch

`720kb` or `1440kb` 3.5-inch

device-node-name is the character node name of the diskette device, which is listed in **/dev/pdsk**. You also may check the **/etc/devlinktab** file, which lists the correlation of DG/UX long names to device node names.

Example:

```
mkfs 360kb /dev/pdsk/3
```

To make the diskette accessible, you also must create a mount point and then mount the DG/UX-formatted diskette. Refer to the section on creating a mount point for a DG/UX file system in this chapter for details.

Creating a mount point for a DOS-formatted diskette

For a DOS file system is on a diskette formatted for MS-DOS, you do not need a virtual disk. You need only to add a DOS file system. To add a file system for a DOS format diskette, follow these steps:

1. Insert the diskette correctly in the diskette drive.
2. Follow this path through **sysadm**:

   `File System -> Local Filesys -> Add`

3. Specify the file system type as **dos**.

   ```
   File System Type: [dg/ux] dos ↵
   ```

4. Enter the complete pathname of the device file that belongs to the diskette drive that holds the DOS diskette. For example, enter the device file named **pdsk3** in the **/dev** directory.

   ```
   Device file: /dev/pdsk3 ↵
   ```

5. Specify the mount directory's full pathname from the root directory. This will form the pathname through which users will access the file system, therefore it is very important. Use the virtual disk name as the last filename in the mount point pathname. For example,

   ```
   Mount Directory: /usr/resources ↵
   ```

   The mount directory you specify here need not exist; **sysadm** will create it for you later on.

6. Specify whether or not, when the file system is mounted, users can write to it. To make the file system read-only (not writable by any user, including root), enter **read-only**. For the default, press Enter:

   ```
   Write Permission: [Read/Write] ↵
   ```

7. Specify whether or not the file system is exportable.

   An exportable file system allows other hosts connected to the network to mount it in their own file systems. If you do not want the file system to be exportable, answer **no** (or press Enter) and go to the next step. Generally, you will not want to make a diskette file system exportable; if you do, answer **yes**. You will need to specify export options.

   ```
   Exportable? [no] yes ↵
   Export Options:
   ```

   For export options, see Table 8–2.

8. **Sysadm** prompts you to confirm before mounting. The prompt includes the words "and export" if you specified that the file system was exportable.

   ```
   Mount [and export] the file system? [yes] ↵
   ```

9. Confirm by pressing Enter at both prompts:

   ```
   Mount [and export] the file system? [yes] ↵
   Ok to perform the operation? [yes] ↵
   ```

10. If the mount directory you specified in step 5 does not exist, **sysadm** prompts

    ```
    Mount point directory /spreadsheets does not exist.
    Do you wish to create it? [yes]
    ```

11. If you enter **no**, **sysadm** aborts the operation and returns to the initial prompt. If you want to create the directory, answer **yes** or press Enter.

    Sysadm displays

    ```
    File system added: /spreadsheets
    File system mounted: /spreadsheets
    File system exported: /spreadsheets    (if it is exportable)
    ```

    You have added and mounted a file system for a DOS diskette. Repeat this procedure for each DOS diskette you want to add.

Creating a mount point for a memory file system

IMPORTANT: The contents of a memory file system are temporary; they are lost at system shutdown.

A memory file system is a diskless file system that, when added, reserves a portion of main memory for file storage. You can specify that the contents of the memory file system cannot be swapped out of main memory. A memory file system is not same as a disk cache, which may include a virtual disk created on a NVRAM board or a fast disk.

▶ To add a memory file system, follow this path through **sysadm**:

```
File System -> Local Filesys -> Add
```

1. Specify the file system type as **ramdisk**:

   `File System Type: [dg/ux]` **ramdisk** ↵

2. Enter the name of the device file in the **/dev** directory that you want to associate with memory file system. You will use this name in future to mount the memory file system. For example, if you want the memory file system to have the name **/dev/memdisk**:

   `Device file:` **/dev/memdisk** ↵

3. Specify the mount directory's full pathname from the root directory. This will form the pathname through which users will access the file system, therefore it is very important. Use the virtual disk name as the last filename in the mount point pathname. For example,

   `Mount Directory:` **/memdisk** ↵

   The mount directory you specify here need not exist; **sysadm** will create it for you later on.

4. Specify whether or not to use wired memory.

   By default, a memory file system is not wired, which means that the operating system may swap parts of the file system to disk to free up needed physical memory. If you specify wired memory, then when the memory file system is mounted the system retains the entire file system in your computer's physical memory. Make sure your computer has enough physical memory to hold the memory file system.

   ```
   Use Wired Memory? [no]  yes ↵
   ```

5. Enter the maximum number of 512-byte blocks that the memory file system can use. The system does not allocate the memory until the file system requires it; therefore, entering a high value will not necessarily use much memory. The maximum amount of memory that the system will allocate for the file system is either the size limit that you specify in this query or the amount of memory available, whichever is less. For example,

   ```
   Maximum File Space: [2048] ↵
   ```

6. Enter the maximum number of disk files that can be present in the memory file system. For example,

   ```
   Maximum File Count: [16384] 128 ↵
   ```

7. Specify whether or not the file system is exportable.

   An exportable file system allows other hosts connected to the network to mount it in their own file systems. If you do not want the file system to be exportable, answer **no** (or press Enter) and go to the next step.

   ```
   Exportable? [no] yes ↵
   Export Options:
   ```

   For export options, see Table 8–2.

8. **Sysadm** prompts you to confirm before mounting. The prompt includes the words "and export" if you specified that the file system was exportable.

   ```
   Mount [and export] the file system? [yes] ↵
   ```

9. Confirm by pressing Enter at both prompts:

   ```
   Mount [and export] the file system? [yes] ↵
   Ok to perform the operation? [yes] ↵
   ```

10. If the mount directory you specified in step 5 does not exist, **sysadm** prompts

    ```
    Mount point directory /memdisk does not exist.
    Do you wish to create it? [yes]
    ```

11. If you enter **no**, **sysadm** aborts the operation and returns to the initial prompt. If you want to create the directory, answer **yes** or press Enter.

    Sysadm displays

    ```
    File system added: /memdisk
    File system mounted: /memdisk
    File system exported: /memdisk (if it is exportable)
    ```

    You have added and mounted a file system for a memory file system

Creating a mount point for a remote file system

In addition to mounting file systems on virtual disks that you create locally, you may mount remote file systems. Remote mounting provides access to a single file system from multiple hosts on a LAN. Refer to the planning worksheet for a list of remote file systems to be mounted.

Note that before you can mount a remote file system, the remote host must export that file system. At the remote host, export a file system by following this path through **sysadm**:

```
File System -> Local Filesys -> Export
```

As an example, you want to access a file system containing account information about a client of your company; it is mounted at **/accounts/midwest/minn_brands** on remote host **igor**. To mount it at **/midwest** on your local DG/UX file system, follow these steps.

1. Execute these **sysadm** steps.

   ```
   File System -> Remote Filesys -> Add
   ```

   Sysadm prompts

   ```
   Mount Directory:
   ```

2. Supply the path the mount directory, which is the location on your DG/UX file system at which you want the remote file system mounted. This will be the pathname by which users access it. To continue with the example above, you would enter

   ```
   Mount Directory: /midwest ↵
   ```

3. Enter the remote hostname. For example,

   ```
   Remote Host Name: igor ↵
   ```

4. Enter the remote mount directory name. The file system must be mounted at that mount point on that host. For example,

   ```
   Remote Mount Directory:  /accounts/midwest/minn_brands ↵
   ```

5. Specify the write permissions; the default is read/write. For example,

   ```
   Write Permission: [Read/Write] ↵
   ```

6. Select the type of NFS (remote) host. Your selection of a hard versus a soft mount will determine the effect of a remote host crash on your current operation. A hard mount suspends your operation until the problem at the remote host has been fixed. A soft mount allows your operation to abort (time out) so that resources are not tied up while the remote host is hung.

   A hard mount is more suitable for write-intensive operations performed with remotely mounted file systems. A soft mount is more suitable for read-only operations in which you risk no loss of data. A soft mount is preferable for operations containing frequent reads and infrequent writes. The advantage of a soft mount in this case is that you guard against unnecessary down time for read operations while accepting a minimal risk of occasional data loss.

   You should never use a hard mount on a remote file system in your host's / (root) file system. Checking files located in the / file system is often a helpful aid in detecting local problems. However, if you use a hard mount on a remote file system in **/tmp** and the remote machine fails, executing the **ls** command in **/tmp** would hang.

   ```
   NFS Mount Type: [Hard]? soft ↵
   ```

7. Specify whether or not you can interrupt an operation initiated with a remote file system. Choosing this option is useful for cancelling an operation involving a hard-mounted file system on a remote host that is hung. Usually, pressing Ctrl-C (or other interrupt function key) cancels the operation. For example,

   ```
   Interruptible? no ↵
   ```

8. Specify whether or not the network will retry in background mode if the OS server does not respond.

   ```
   Retry in background? [yes] ↵
   ```

9. Specify whether or not to mount the file system.

   If you answer **no**, **sysadm** only adds the file system name to **fstab** for later automatic mounting.

10. To mount the file system, press Enter at both prompts:

    ```
    Mount the file system? [yes] ↵
    Ok to perform the operation? [yes]
    ```

    If the mount directory you specified in step 2 does not exist, **sysadm** prompts

    ```
    Mount point directory /midwest does not exist.
    Do you wish to create it? [yes]
    ```

11. If you enter **no** to have **sysadm** abort and returns to the initial prompt. To create the directory, press Enter.

    Sysadm displays

    ```
    File system added: /midwest
    File system mounted: /midwest
    ```

    The file system is mounted at the desired mount point. Repeat this procedure to continue mounting the remainder of your remote file systems.

End of Chapter

9 Using DG/UX file systems

Your application determines how you use many file systems. For example, you use a CD-ROM that contains a documentation browser by following the application's instructions. The application creates your working environment, making it unnecessary for you to perform operations that deal directly with a physical device or the mounted file system.

This chapter discusses primarily how to use DG/UX file systems on a Winchester disk. In addition, this chapter reviews typical uses for the other devices.

Using Winchester drives or diskettes containing DG/UX file systems

Typically, you divide virtual disks containing DG/UX file systems into subdirectories to organize regular files. Regular files hold data, which may be an application, text, source code, work area, or anything else that you need to store. Let's use the **home** virtual disk, designed to house users' home directories, as an example to illustrate how you might parcel out storage to users.

A virtual disk named **home** should be sized at 220,000 blocks for five users, with each user having 40,000 blocks each. Accordingly, you might visualize this **home** virtual disk like this:

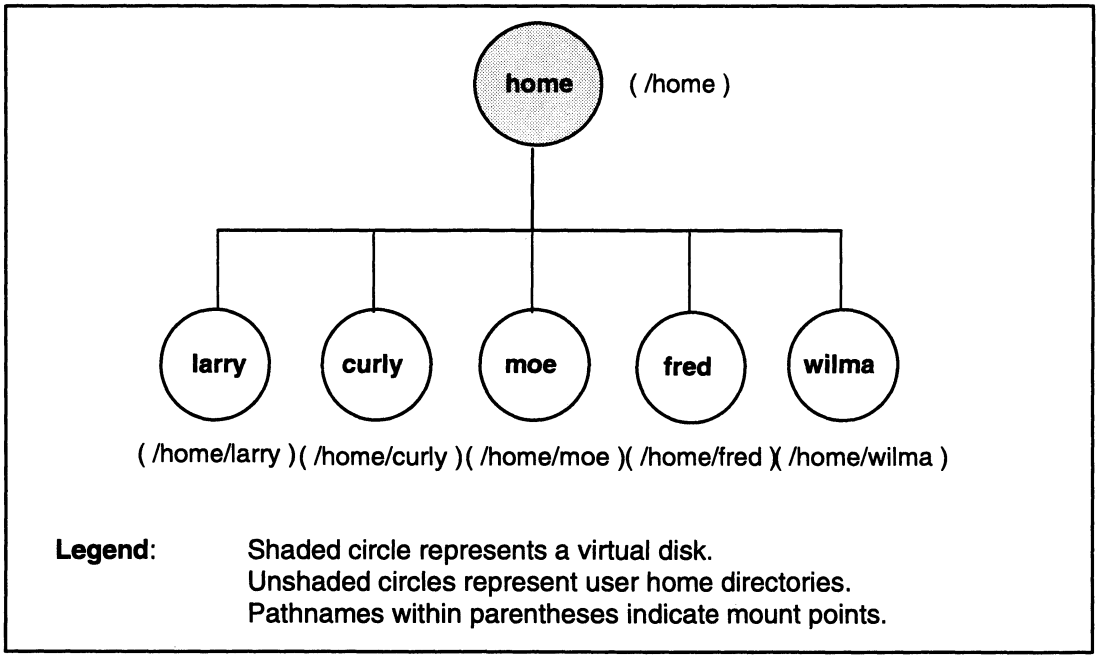


Figure 9–1 Example of home directory

Even though each user in a DG/UX system configuration will manage his workstation and attached devices, typically, the **home** virtual disk is centrally located and is accessed over a network. In this example, for each user, you make individual directories using the shell **mkdir** command:

```
mkdir -c directory-name ...
```

The **mkdir** command makes the specified directories in mode 777 (read, write, and execute permissions for all), which you can alter with the **umask** command. The **–c** option declares each home directory as a control point directory, which sets an upper limit to the size of each directory and its descendents. You define the limit with the **cpd** command:

```
cpd -b size-per-directory (in blocks)
```

The following commands create five control point user home directories, sized at 40,000 blocks each:

```
# mkdir -c larry curly moe fred wilma ↵
# cpd -b 40000 larry curly moe fred wilma ↵
```

It's unwise to allocate all space in the virtual disk for home directories; keeping a reserve allows you to enlarge the control point directory limit later.

You can verify the users' subdirectories that you created by listing the contents of the directory.

Enter the command **ls** with the **-l** option.

```
# ls -l )
total 700
%rwxrwxrwx  1 curly    accounting        512 Apr  9 15:18 curly/
%rwxrwxrwx  1 fred     accounting        512 Apr  9 15:18 fred/
%rwxrwxrwx  1 larry    accounting        512 Apr  9 15:18 larry/
%rwxrwxrwx  1 moe      accounting        512 Apr  9 15:18 moe/
%rwxrwxrwx  1 wilma    accounting        512 Apr  9 15:18 wilma/
```

Each field in this long listing identifies several aspects of file and directory access that you control with a complement of shell file management commands.

Also, you may check the size of the control point directory as follows:

```
# cd home )
# cpd larry )
larry           0 blocks     (    0  bytes)      0 file nodes
maximum:        40000 blocks ( 19.0 Mbytes)      No limit
```

With a subdirectory structure in place with the appropriate permissions, users can create and use files, as desired. Refer to the **mkdir**(1), **umask**(1), **cpd**(1), **chmod**(1), **chgrp**(1) and **chown**(1) manual pages for more details. Also, refer to *Using the DG/UX™ System* for information on managing your DG/UX files and directories and *Managing the DG/UX™ System* for details on setting up and managing user accounts.

Removing a diskette from its drive

Before removing a diskette from the drive, you should unmount its file system with the **sysadm** operation:

```
File System -> Local Filesys -> Unmount
```

Refer to Chapter 11 for information on the unmount operation.

If you remove the diskette without unmounting, you can unmount the file system after you have physically removed the diskette from the drive.

If you put another diskette in the drive without unmounting the previous one, the system prompts for the previous diskette. You cannot access the drive until you insert the previous diskette and unmount it.

Creating text files

A vast array of tools is available for creating and storing text. A simple way to create a text file is to use the **cat** command, which accepts your typed input and sends it directly to a file.

There are many text editors, text formatters, and desktop publishing systems. Examples of text editors are **vi**, **ed** and **Emacs**; third-party editors, WordStar® and WordPerfect®; and desktop publishing systems Interleaf® and FrameMaker™.

See the **cat**(1), **vi**(1), **ed**(1), and **emacs**(1) man page or the particular tool's documentation for more information. **Vi** and **ed** are documented in *Using the DG/UX™ Editors*.

Creating executable files

You may create an application's source file using a variety of programming languages. If, for example, you wrote a C shell script in a text file, you must distinguish it as such by entering the following command as the first line in the file.

```
#! /bin/csh
```

In addition, you must assign an execute permission to the script file in this format:

```
# chmod u + x script-file
```

Refer to *Using the DG/UX™ System* for information on writing Bourne shell and C shell scripts

Loading and setting up third-party packages

Before you can access and use a third-party package, you must load it and set it up on a virtual disk.

To add a package (obtained from either Data General or a third-party vendor), you must:

- Load the package from the release medium
- Set up the package to run with the DG/UX system

Most vendors' packages are loaded at **/usr/opt/***package–name*, where *package-name* is the name of the package.

Before you load and set up a package, read its release notice. You must create the required virtual disks and their file systems, and

add and mount them on the DG/UX file system before you can load the package. For many packages, the installation script leads you through the creation of the virtual disk and file system, and the file system mounting phases.

For more details on loading and setting up third-party packages, see *Managing the DG/UX™ System*.

Using a NVRAM board

You typically use a NVRAM board for caching. The application determines the behavior of the NVRAM board. Check your application's documentation for details.

Using a Magneto-Optical or WORM platter

Magneto-optical and WORM platters typically are configured into a jukebox cabinet whose OpStar software controls its use. Refer to the *Using OpStar™ 2.00 Optical Storage Software* for details.

Using a CD-ROM

You use a CD-ROM primarily for storing large amount of text or data for your retrieval. The application determines the behavior of the CD-ROM. Check your application's documentation for details.

Using a memory file system

For applications that would benefit from very fast access to relatively small databases, you can create memory ("ramdisk") file systems. A memory file system is a portion of your computer's physical memory, formatted as a DG/UX file system and mounted. You can access it the same as any file system.

Using a DOS-formatted diskette

You use a DOS-formatted diskette with an MS-DOS application. Check your application's documentation for details.

Since the DG/UX system accepts DOS diskettes (you can mount a DOS diskette), you can use the DOS diskette both on a personal computer running MS-DOS and the DG/UX system.

Before removing a DOS-formatted diskette from the drive, you should unmount the diskette's file system. If you remove the

diskette without unmounting, you still may unmount the file system after you have physically removed the diskette from the drive.

If you put another diskette in the drive without having first unmounted the previous one, the system prompts for the previous diskette. You cannot access the drive until you re-insert the diskette and unmount it.

End of Chapter

10 Maintaining disk devices

It is assumed that you are now doing useful work with your disk devices — you have prepared the devices and have put data on them in the form of text, an application or storage. Over the lifetime of your disk devices, however, you may need to perform maintenance procedures on either the physical disk or the virtual disk (including mirrors and caches).

Physical and virtual disk operations are different from file system operations (DG/UX, DOS, CD-ROM, and memory file system), which affect only the file systems residing on the virtual or physical disks. Refer to Chapter 11 for file maintenance operations.

This chapter covers four categories of disk maintenance: physical, virtual, mirrors, and caches.

Maintaining physical disks

This chapter covers these physical disk maintenance operations:

- Copying a physical disk
- Deregistering a physical disk
- Removing a physical disk from the configuration
- Converting a physical disk between logical and virtual disk formats
- Building a virtual disk by hand from a failed conversion
- Erasing the entire physical disk
- Repairing a damaged virtual disk information table (VDIT)
- Tracking bad blocks on a physical disk
- Enabling write verification
- Listing physical disks

See Chapter 2 for information on configuring a device and Chapter 6 for details on registering a device.

Copying a physical disk

The physical disk copy operation copies the contents of a source disk to a destination disk. The contents will be identical, but the location of the content on the destination disk may be different. You can use this operation to copy between disks of different types and

sizes (such as from a Winchester disk to a magneto optical disk). However, be sure the destination disk has sufficient space.

The copy operation performs an over-writing copy, obliterating the contents, if any, of the destination disk before writing the source content to it.

Why and when to copy

There are several reasons for copying a physical disk:

- When you replace an old disk with a new one, you can copy the contents of the old one to the new one.
- When installing a number of systems that need the entire operating system loaded from tape, you can save time by loading only the first disk from tape and then copying the other disks from it.

  To use the copy facility for this purpose, you may need to know how to jumper the destination disk for a different address or SCSI ID. This is particularly true if you intend to use the copy facility to load one workstation's SCSI disk from another workstation's SCSI disk. Typically, both SCSI disks come from the factory jumpered as **sd(insc(0),0)**. Before connecting them both to the same system, you need to change the jumpers on one of them; for example, to reset one to a SCSI ID of 1.

  If you use the copy facility to make copies of the DG/UX system virtual disks (**root** and **usr**), perform the copy before setting up software packages (such as TCP/IP) with the DG/UX system. If you perform the copy after setting up packages, you will duplicate system-specific data (Internet addresses, system names, and so on) from the source disk.

- When you need to remove a disk from your system, use the copy operation to copy the contents of the disk onto another disk having the required available space.

  There may also be other scenarios where you find the physical disk copy operation useful.

Performing a disk copy

The source physical disk (the disk being copied) may remain accessible to users for the duration of the copy operation. However, the data copied to the destination will be inconsistent with the data on the source, which continues to undergo I/O activity.

To prevent inconsistent data between source and destination disks, take the source physical disk off-line before performing the disk copy. Perform the following tasks:

1. All file systems located on the source disk must be unmounted.

   To help you determine the file systems to be unmounted, use the **sysadm** operation list to view the contents of the source disk.

   ```
   Device -> Disk -> Physical -> List
   ```

   See a section in this chapter for details on listing physical disks.

2. Unmount the affected file systems.

   ```
   File System -> Local Filesys -> Unmount
   ```

   Refer to Chapter 11 for information on how to unmount a file system.

3. Make sure that the source disk is registered.

   ```
   Device -> Disk -> Physical -> Register
   ```

   Refer to Chapter 6 for information on how to register a disk.

4. Deregister the destination disk.

   ```
   Device -> Disk -> Physical -> Deregister
   ```

   Refer to a later section in this chapter for information on how to deregister a disk.

5. Copy the content of the source disk to the destination disk.

   ```
   Device -> Disk -> Physical -> Copy
   ```

6. Supply the name of the source disk device to be copied.

   ```
   Source physical disk:
   ```

7. Supply the name of the destination disk device.

   ```
   Destination physical disk:
   ```

8. Following a successful copy, register the destination disk only.

   The source and destination disks will contain identical virtual disks with identical content.

   IMPORTANT: Bad blocks will not be remapped to the destination. To recover bad blocks for the destination device, use the unmap operation, which is covered in a later section on bad block tracking.

9. Deregister the source disk.

10. Remount the file systems on the destination disk.

    ```
    File System -> Local Filesys -> Mount
    ```

    Refer to Chapter 11 for information on how to mount a file system.

Deregistering a physical disk

Deregister a physical disk to make its virtual disks (or logical disks, if used in compatibility mode) inaccessible. You should deregister a disk before powering it down or removing it from the configuration.

Besides deregistering a device, you must first unmount its file systems. Unmounting a file system makes it inaccessible. Of course, you needn't deregister unregistered devices such as those that do not contain a virtual disk information table — CD-ROM, magneto optical disk and diskette containing one DG/UX file system. Their file systems, however, must be unmounted.

You cannot deregister a physical disk that is considered busy or open, such as one that contains any part of the operating system — **root**, **usr**, **swap**, **usr_opt_X11** — or that is being accessed. To deregister the system disk, you must take the system down and use stand-alone **sysadm** instead. Refer to Appendix E for a review of the **sysadm** interfaces.

Deregistration fails if any virtual disk on the physical disk is in use; for example:

- A virtual disk's file system is mounted.
- The **swap** virtual disk is located on it.
- An application is using a virtual disk.
- The parent virtual disk for a child virtual disk being used elsewhere is located on the physical disk.

If you deregister a disk to disconnect it from the system, make sure that you turn off its power before removing it. You do not need to turn off power, however, for disks in disk-array subsystems.

1. Select the **sysadm** operation.

   ```
   File System -> Local Filesys -> Unmount
   ```

   Refer to Chapter 11 for information on unmounting a file system.

2. Select the **sysadm** operation.

   ```
   Device -> Disk -> Physical -> Deregister
   ```

3. Specify the name of the physical disk to be deregistered:

   ```
   Physical disk(s):
   ```

   The system displays a message confirming a successful deregistration.

Removing a physical disk from the configuration

Removing a physical device from a configuration (or deconfiguring) removes its entries from either **/dev/pdsk** (block disk nodes) or **/dev/rpdsk** (character disk nodes), whichever is appropriate.

In addition, if the system file, which underlies the kernel, contains the deconfigured device, you eventually must remove it from the system file and rebuild the kernel. At system boot time, the system configures automatically all devices listed in the system file. If, however, you used the asterisk (*) notation in the device name in the system file, you do not have to delete it from the system file and rebuild the kernel.

1. If you have a removable-media drive, keep reading this step. Otherwise, go to the next step.

   If you wish to remove the medium from a removable-media drive (diskette drive, CD-ROM drive or magneto-optical drive), make sure you unmount the file system and deregister the file system first. Removing a mounted or registered disk or diskette results in an error messages such as:

   ```
   From System:
   The file system on device device-name sealed,
               Status nnnnnnn   (nnnnnn is a status number.)
               Run fsck to restore
   ```

   or:

   ```
   File system is no longer fault tolerant
   ```

   To recover, you must re-insert the diskette and run the file system checker (see Chapter 11).

2. If you have a diskette drive, keep reading this step. Otherwise, go to the next step.

   To deconfigure a diskette drive, first you must unmount the file system before you deregister it.

   ```
   File System -> Local Filesys -> Unmount
   ```

   Refer to Chapter 11 for information on unmounting a file system.

3. Before you remove a device from the configuration, while the system is operating, you must first deregister it.

   ```
   Device -> Disk -> Physical -> Deregister
   ```

   Refer to the section in this chapter on deregistering a physical disk for details.

4. After deregistering and unmounting, if necessary, select the following **sysadm** operation.

```
Device -> Deconfigure
```

A successful deconfiguration is confirmed.

Alternatively, you may choose to use the **admdevice -o deconfigure** *'device-name'* command instead.

Converting a physical disk between logical and virtual disk formats

IMPORTANT: This operation is relevant only if your DG/UX system has formerly run a DG/UX software release before DG/UX 5.4 Release 3.00. Given that, if any physical disk is still in logical-disk format, you may consider using this operation. Otherwise, this operation is irrelevant. To find out if any of your physical disks is logical-disk formatted, use the list operation (covered in a later section in this chapter) and check the physical disk's format.

There was an opportunity to convert physical disks from logical-disk format to virtual-disk format when your DG/UX system was upgraded to Release 3.00. However, if there were compelling reasons for remaining in logical-disk format (such as to maintain compatibility with an application that relies on logical disks, for example), the DG/UX system permitted it. A logical disk to be used in a virtual-disk environment is registered in compatibility mode.

Why should you convert to virtual disk format? Compatibility mode puts restrictions on how you operate on disks. A disk that is registered in compatibility mode can be read and written, but it cannot be altered with operations that enlarge, shrink, move, or mirror — operations that re-create the disk's metadata. Metadata describes the layout of a physical disk. It is desirable to have your physical disks in virtual-disk format because its metadata offers a flexibility that will likely accommodate future disk formats. You are strongly encouraged to convert physical disks to virtual-disk format.

The conversion process converts only the disk metadata format whether it be logical or virtual; it does not touch file system metadata or user data. The conversion utility converts the metadata only after it ensures that it is safe to do so, without risk of data loss or corruption.

The process of converting a physical disk's metadata from logical- to virtual-disk or virtual-to-logical format takes only a moment per disk. When converting from logical- to virtual-disk format, there is no risk of format conversion failure because of limited disk space because the virtual-disk format requires no additional space.

Some conversion pointers follow.

- The physical disk(s) whose logical disks you want to convert must be deregistered.
- The disk you want to convert must be writable. Read-only disks like CD-ROMs and WORMs (write once, read many) cannot be converted. You can continue to use these in compatibility mode.
- If a logical disk has pieces on more than one physical disk, convert each physical disk containing a piece. Otherwise, you will have to join the missing pieces later on.
- If a mirrored logical disk has images on more than one physical disk, convert each physical disk containing an image. Success or failure depends how many lost images you will tolerate (see Chapter 7 for information on mirroring a virtual disk). To recover, you may assemble errant images by hand. Refer to the next section for a discussion of rebuilding virtual disks by hand.
- Converting back to logical disks is possible only if you have used none of the new virtual disk features on any logical disk in the physical disk.

  Some virtual disk hierarchical constructs may not successfully revert to logical disk format because of hierarchical complexity. For example, conversion of an aggregated virtual disk composed of other aggregations to the logical disk equivalent does not work. If a backward conversion fails, your virtual disks remain intact. You may continue to use simple partitions, aggregations, and mirrors; but you cannot use higher order configurations such as aggregated aggregations or aggregated mirrors. The next section provides help on reconstructing virtual disks from failed components.

- Dismantle (delete) caches before you convert their constituent physical disks.

1. To convert any logical-disk formatted physical disk to virtual-disk format, use the following **sysadm** sequence:

   ```
   Device -> Disk -> Physical -> Convert
   ```

2. Specify the format to which you want to convert the physical disk: logical or virtual disk format.

   ```
   Conversion target format: [Virtual disk format] ↵
   ```

3. Specify how to handle incomplete conversions:

   ```
   Forcefulness: [Careful] ↵
   ```

   Options are:

`Careful`	If there is a problem, abort the conversion.
`Forceful`	Despite any problems, convert anyway. The conversion produces incomplete logical and virtual disk components. You may be able to salvage the components and reconstruct a logical and virtual disk by hand (see the next section). Use this option only if the disks with the missing pieces are not available.
`No-write`	The conversion program performs a trial conversion. It reports any problems, but will not convert the physical disks regardless of success or failure. If you choose this, you will later need to use one of the other choices.

4. To convert the logical disks on all physical disks, take the default. Or specify the physical disks one by one.

   ```
   Physical disk(s): all ↵
   ```

   Sysadm will prompt for disk(s) to convert. As always, if a list of valid responses is not on display, enter **?** to display one.

Building virtual disks from a failed conversion by hand

If a conversion to virtual disk format fails, the unconverted (or "orphan") components will remain intact and unusable. To rebuild the desired virtual disk "by hand," you must be able to identify the unconverted components. You may do so by listing the contents of physical disks and recognizing unnamed children and their sizes. Such an endeavor will depend largely on your knowledge of locations and sizes of logical disks on physical disks in your configuration. This can be a difficult task. Still, it may be possible to correctly link several unnamed, same-sized mirror images to a new virtual disk mirror or to create an aggregation from several unnamed partitions.

Erasing the entire physical disk

To clear the surface of a physical disk, you must perform two simple operations: deregister the physical disk and re-create a virtual disk information table (VDIT). Deregistration makes the virtual disks

on the physical inaccessible temporarily and re-creation of a VDIT erases all previous content from the physical disk before it lays down a new VDIT.

1. Deregister the physical disk following this **sysadm** path:

   ```
   Device -> Disk -> Physical -> Deregister
   ```

   Refer to the section in this chapter on deregistering a physical disk for more information.

2. Re-create a virtual disk information table following this **sysadm** path:

   ```
   Device -> Disk -> Physical -> Soft Format -> Create VDIT
   ```

   Refer to Chapter 6 for details on creating a VDIT.

Repairing a damaged virtual disk information table

Although rare, a virtual disk information table (VDIT) can get damaged. Usually, this results from a software bug; for example, when you attempt to boot a pre-DG/UX 5.4R3.00 kernel on a DG/UX 5.4R3.00 system. Such an illegal operation can destroy the data at the beginning of a physical disk, which contains the VDIT.

The system alerts you to this problem with the following message on your system console:

```
Error: The VDIT for disk 'sd(insc(),1)' has
     degraded and is using only one VDIT copy.
```

1. To recover, copy the duplicate VDIT, which was written at the end of the physical disk during soft formatting, to the damaged location. Select the **sysadm** operation:

   ```
   Device -> Disk -> Physical -> Repair VDIT
   ```

2. You may list the physical disk for a display of both VDIT locations using this **sysadm** operation.

   ```
   Device -> Disk -> Physical -> List
   ```

   The physical disk must be deregistered and re-registered before the VDIT repair can take effect. The repair operation will attempt to deregister and re-register the physical device; but, will fail if it can't deregister the device. (See the section in this chapter on deregistering a device for explanations of why deregistration fails.) If this happens, you may continue to use the disk in degraded mode until you are able to successfully deregister and register the

physical disk. A degraded mode of operation allows read and write operations to continue, but prohibits operations that alter the disk's metadata such as enlarging, shrinking, moving, and mirroring. Although allowed, do not operate in a degraded mode for an extended period.

3. Supply the name of the physical disk whose VDIT requires repair. An example follows:

```
Physical disk(s): sd(insc(),1) ↵
```

This operation takes only a few seconds. The system copies the VDIT located toward the end of the physical disk to the beginning as well.

Tracking bad blocks on a physical disk

Disk units occasionally develop flaws in the disk surface. Most disk hardware keeps track of these bad parts without depending on the operating system to do it for them; nevertheless, the DG/UX system offers its own bad block tracking mechanism in case the disk hardware fails to detect or remap a flawed disk block.

When the DG/UX system detects a flaw on a disk, it flags the block (a 512-byte portion of disk space) as bad. If a write operation was performed, the system finds a good block to replace the bad block. The operating system takes care of redirecting reads and writes intended for the bad block so that they go to the replacement block instead. A part of the disk called the bad block remap area contains good blocks reserved specifically for this purpose: to replace blocks that go bad elsewhere on the disk.

When the system creates the bad block remap area, it offers a default size for the remap area. You may specify another size if you like, but the default size is generally sufficient.

Mapping bad blocks

Bad blocks are parts of the physical disk that may be unreliable for storage and retrieval of information. The operating system keeps track of bad blocks by automatically listing them in its bad block table. However, it may overlook some bad blocks. If you suspect that a particular block is unreliable or if diagnostics have shown a block to be unreliable, you can add that block to the bad block table using the mapping operation. The physical disk containing the bad blocks to be remapped must be registered, and you must know the physical address of the bad disk block. Use the physical disk list operation (covered in a separate section in this chapter) to find out the bad block's address.

1. Make sure that the physical disk containing the bad blocks to be remapped is registered. To check a disk's registration status, use the **sysadm** list operation:

   ```
   Device -> Disk -> Physical -> List
   ```

2. List and take note of the bad blocks using the following **sysadm** operation:

   ```
   Device -> Disk -> Physical -> Bad Blocks -> List
   ```

   Refer to the section in this chapter on listing back blocks for details.

3. To map bad blocks on a physical disk, select the **sysadm** operation:

   ```
   Device -> Disk -> Physical -> Bad Blocks -> Map
   ```

4. After you select this operation, **sysadm** prompts for the name of the physical disk for which you wish to map bad blocks.

   ```
   Physical Disk:
   ```

5. Enter the number of one or more blocks to map. Enter a block number, a list of block numbers (separated by commas), or a range of block numbers (two numbers separated by a hyphen).

   ```
   Block Numbers:
   ```

   After you confirm your desire to perform the operation, the system performs the remap. The system displays messages that confirm the remapping.

   If you map a block that contains data, the system will not attempt to copy the data from the bad block to the replacement block. The contents of the replacement block are unknown until a write operation writes data to the block.

   If you tell DG/UX to force remap a block, it does so without verification of the original block's status and without confirmation. A forced block ends up as mapped after the system writes it.

What to do if bad block remap area runs out of good blocks

If the bad block remap area does not have any more good blocks (which is highly unlikely), the operation returns an error message telling you that it cannot add the bad blocks to the table. If this happens to a SCSI disk, you should question seriously the reliability of the physical disk. You may wish to copy of the contents of the questionable physical disk to a known good disk (using procedures in this chapter) and then remove the faulty disk

from your configuration. Refer to the section in this chapter for details to remove a physical disk from the configuration.

1. Back up the contents of the physical disk using the preferred method. You may perform a physical disk copy (using the procedures in this chapter), use a favorite backup utility (see the Preface of this manual for a list of software packages and *Managing the DG/UX™ System* for details on performing backups), or dump to tape (see Chapter 3 in this manual for options).
2. Soft format the physical disk, increasing the size of the bad block remap area using the following **sysadm** operation.

   ```
   Device -> Disk -> Physical -> Soft format -> All
     Soft Formatting Steps
   ```

   Refer to Chapter 6 for details on performing all soft formatting steps.
3. Force bad block remapping through the following **sysadm** operation:

   ```
   Device -> Disk -> Physical -> Bad Blocks -> Map
   ```

   Refer to the preceding section for details on mapping bad blocks.
4. Copy the backed up data to the physical disk for which the bad block remapping table has been enlarged. Consult your backup procedure for details on restoring the data.

Unmapping bad blocks

The operation recovers the remapped blocks that no longer need to be remapped. To remap a block means to designate another block to use in place of the bad or unreliable block. For example, block 100000 might be remapped to another block. After you repair the disk, however, you may want to remove block 100000 and all other formerly bad blocks from the bad block table.

1. To unmap (recover) bad blocks on a physical disk, select the **sysadm** operation:

   ```
   Device -> Disk -> Physical -> Bad Blocks -> Unmap
   ```
2. Specify the name of the physical disk for which you wish to unmap bad blocks.

   ```
   Physical Disk:
   ```
3. Enter the number of one or more blocks to unmap. Enter a block number, a list of block numbers (separated by commas), or a range of block numbers (two numbers separated by a hyphen).

```
Block Numbers:
```

After you confirm your desire to perform the operation, the system recovers the blocks. The system displays messages that confirm the recovery of blocks.

Listing Bad Blocks

Typically, you check for bad blocks after soft formatting (see Chapter 6). The operation lists addresses (in decimal form) of bad blocks.

1. To list bad blocks on a physical disk, select the **sysadm** operation

   ```
   Device -> Disk -> Physical -> Bad Blocks -> List
   ```

2. Supply the name of the physical disk whose bad block table you wish to view.

   ```
   Physical Disk:
   ```

   An example of a bad block table follows:

```
Physical Disk(s): sd(ncsc(0,7),0,0)
       Block  Index  Status
          20      0  force
          21      1  force
          22      2  force
          23      3  force
          24      4  force
          25      5  force
          26      6  force
```

The first column lists the bad blocks that you specified for remapping. The second column identifies the block number in the remap area to which the bad block is remapped. The third column specifies the status of the bad block remap area. Possible status conditions are:

`mapped`	The system writes the bad block to a new block and redirects all I/O to the new block.
`unmapped`	The kernel detects an error upon attempting to read this block. Although the original block has a remap block associated with it, the contents of the remap block are undetermined. Until the system performs a write to the new block, reads directed to the original block will fail.
`force`	A user performs a map block operation on this block. Although the original block has a remap

block associated with it, the contents of the remap block are undetermined. Until the system performs a write to the new block, reads directed to the original block will fail.

`pseudo` — An entry with this status corresponds to a block that does not necessarily need remapping but whose contents are undetermined. Reads to pseudo bad blocks will fail. The first write to a pseudo bad block sets its contents and deletes the corresponding remap table entry.

`bad` — An entry with this status represents a block in the bad-block map area itself that is unusable.

Enabling write verification

In applications where data integrity is vital, you can benefit from write verification. By turning write verification on for a physical disk, you can be sure that data written to the disk is readable.

You can enable write verification only for SCSI disks that support the feature. See your disk hardware documentation.

Normally, disk hardware does not verify that data just written to disk is readable. This behavior makes your data vulnerable to flaws in the storage medium because the disk has no way of detecting when it has just written to a flawed block on the disk, for example. With write verification turned on for a disk, however, the disk hardware compares the data read with the data received. If the data does not match, the hardware returns an error to the system. Thus, write verification ensures the integrity of your data.

The tradeoff with write verification is in performance. The additional verification overhead in the hardware can have a significant impact on the performance of write-intensive applications. You should experiment with your applications to see how write verification affects performance. Write verification has no effect on read operations.

To turn write verification on for a disk, use the **dkctl** command with the **wchk** option.

For example, the following command line turns on write verification for disk **/dev/pdsk/0**:

```
# dkctl /dev/pdsk/0 wchk ↵
```

This command line enables write verification for the disk every time you boot the system. To enable write verification only until the next system boot, include the `-t` (temporary) option:

```
# dkctl -t /dev/pdsk/0 wchk ↵
```

To turn write verification off for the disk, issue this command:

```
# dkctl /dev/pdsk/0 -wchk ↵
```

See the **dkctl**(1M) manual page for more information.

Listing physical disks

1. To list statistics about a physical disk, select the **sysadm** operation:

   ```
   Device -> Disk -> Physical -> List
   ```

2. Select the physical disks whose statistics you want to list. You may select all (registered and unregistered devices), only registered devices, or explicit devices by name.

   ```
   Physical Disk(s)  [all]:
   ```

3. Specify the type of list you want: normal, partition, and verbose.

   ```
   Listing Style: [normal] :
   ```

   `normal` — Displays the device name, regardless of its availability (not in use by another host such as with dual-ported hosts) or its registration status. The listing also identifies the disk as formatted for logical disks or virtual disks, the disk's total size in 512-byte blocks, and if registered, the number of free blocks remaining on the disk.

   `partitions` — Enumerates the user-created virtual disk partitions and system partitions on the physical disk. The listing includes the partition's name, its starting address, and size in 512-byte blocks. In cases where there are multiple partitions forming an aggregation, each partition is identified by its role—a piece of an aggregation and its number in a series. A partition listing represents the information about the virtual disk information table, bad block map area, and bootstrap as `<Various System Partitions>`.

   `verbose` — Produces a detailed listing of the system partitions and the user-created partitions.

4. List the contents of the physical disk label of each physical disk. The label describes the geometry of the disk that the operating system uses to manage the space on the physical disk.

   ```
   List Label? [no]
   ```

   An example of a normal listing follows:

```
Disk name          State   Reg? Format Total blocks  Free blocks
sd(insc(0),0,0)    avail      y vdisks      1295922      1230279
```

An example of a normal listing and its label follows:

```
Disk name           State   Reg? Format Total blocks  Free blocks
sd(cisc(0,6),0,6)   avail     y  vdisks      2311043      2298977
Disk Label:
cylinders_per_drive     1224
visible_cylinders_per_drive     1219
tracks_per_cylinder     15
sectors_per_track       47
bytes_per_logical_sector        512
bytes_per_unformatted_sector    0
defect_info_start_sector        0
bytes_in_defect_info    0
number_of_relocation_areas      0
sectors_per_relocation_area     0
next_relocation_sector  0
interleave      1
head_skew       3
cylinder_skew   13
head_group_skew 0
spares_per_track        1
bytes_per_data_preamble 0
bytes_per_id_preamble   0
base_head_for_volume    0
flags   0
bytes_in_gap_1  0
bytes_in_gap_2  0
sanity_flag     305441741
version_number  1

Bootstrap: start = 8, size = 500, version = 1
```

An example of a partitions listing follows:

```
Disk name                  State   Reg? Format Total blocks  Free blocks
sd(insc(0),0,0)            avail     y  vdisks       1295922      1230279

   Partition Name                 Role              Address         Size
   <Various System Partitions>                           0          627
   payroll                                             627         5000
   Part of zenith1                Piece  1 of  2      5627        30000
   Part of zenith1                Piece  2 of  2     35627        30000
   <free space>                                        65627      1230279
   <Various System Partitions>                       1295906           16
```

An example of a verbose listing follows:

```
Disk name                  State   Reg? Format Total blocks  Free blocks
sd(insc(0),0,0)            avail     y vdisks       1295922      1230279

   Partition Name                 Role              Address         Size
   .Label,2CA9A8E1                                       0            1
   .Primary_Vdit,2CA9A8DF                                1           16
   .Bootstrap,2CA893F1                                  17          500
   .Primary_Bad_Block_Table,2CA893F2                   517            5
   .Remap_Area,2CA893F3                                522          100
   .Secondary_Bad_Block_Table,2CA893F4                 622            5
   payroll                                             627         5000
   Part of zenith1                piece  1 of  2      5627        30000
   Part of zenith1                Piece  2 of  2     35627        30000
   <free space>                                        65627      1230279
   .Secondary_Vdit,2CA9A8E0                          1295906           16
```

Maintaining virtual disks

This chapter covers the following virtual disk maintenance operations:

- Copying a virtual disk
- Moving a virtual disk
- Removing a virtual disk
- Renaming a virtual disk
- Expanding a virtual disk
- Effectively expanding a striped virtual disk
- Shrinking a virtual disk
- Effectively shrinking a striped virtual disk
- Listing a virtual disk

See Chapter 7 for information on creating virtual disks (including mirrored and cached virtual disks). A later section in this chapter

contains information on maintaining mirrored and cached virtual disks.

Copying a virtual disk

How you copy a virtual disk depends on its read/write status. Choose the appropriate operation from the following sections.

Copying a readable and writable virtual disk

IMPORTANT: To copy a read-only virtual disk (such as one on a CD-ROM device), do not use the **sysadm** virtual disk copy operation. Should you attempt to copy such a virtual disk, you will receive an error. Instead, go to the next section for instructions.

Recommendations and requirements for the operation follow:

- The source and destination virtual disks should be the same size, although the destination may be larger, but not smaller than the source.
- The physical disks on which the source and destination reside must both be registered.
- The source and destination virtual disks must have different names.
- The source virtual disk's file system may be mounted and in use (for example, users accessing a database on the virtual disk) when being copied.

  Following a copy operation, two copies exist: the source and the destination. If you copy a mounted virtual disk, its data will be consistent (its data is up-to-date despite its continued access during the copy operation) and the destination virtual disk will be suitable for mounting as a DG/UX file system.

CAUTION: This operation destroys any data on the destination virtual disk.

1. To copy a virtual disk that is readable and writable, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Copy
   ```

2. Select the virtual disk that you want to copy.

   ```
   Source Virtual Disk:
   ```

3. Select the name of the virtual disk you want to copy to. The destination virtual disk must already exist, it must have a name that is different from the source disk, and it cannot be in use.

```
Destination Virtual Disk:
```

4. Select a number of milliseconds for the system to wait between successive I/O operations required to complete the copy.

   Choosing a non-zero value slows down the movement of data and can reduce contention for a device. Larger numbers slow down the copy operation. Choosing zero causes the copy operation to proceed at full speed; thus, slowing down all other users' access to the device.

   ```
   Throttle Value in Milliseconds: (0-300) [0]
   ```

Copying a read–only virtual disk

Use this procedure to copy a read-only source virtual disk to a destination virtual disk.

IMPORTANT: Do not use this procedure as a general-purpose operation for copying virtual disks on read-write physical disks. Use the **sysadm** copy virtual disk operation instead.

Requirements and recommendations for the operation follow:

- You may perform the copy while the source virtual disk is mounted and in use.
- The destination virtual disk, however, cannot be mounted.

1. To create explicitly a destination virtual disk, use the **sysadm** operation:

   ```
   Disk -> Virtual -> Create
   ```

   Specify the same attributes for the destination as the source, particularly its size.

2. Copy the source virtual disk to the destination virtual disk using the following command syntax:

   ```
   # cp /dev/dsk/source /dev/dsk/destination
   ```

   cp is the shell command to perform the copy operation. See the **cp**(1) manual page for more information.

   /dev/dsk is the standard location for mounted virtual disks. Each time you create a virtual disk, the system automatically logs an entry to this file.

3. After you complete the copy operation, unmount the file system on the source virtual disk and mount the file system on the destination virtual disk (see Chapter 11).

```
File System -> Local Filesys -> Unmount

File System -> Local Filesys -> Mount
```

Moving a virtual disk

The difference between a copy and a move operation is that a copy results in two copies having different names. A move results in one copy. The content of the source is copied to the destination, which assumes the name of the source. The source is deleted.

The move operation also allows you to keep the source and rename it, while the source's content is copied to the destination which bears the name of the source.

Figure 10–1 shows a comparison of the two operations. The virtual disk's file system may be mounted and in use (for example, users accessing a database on the virtual disk) when you move it.

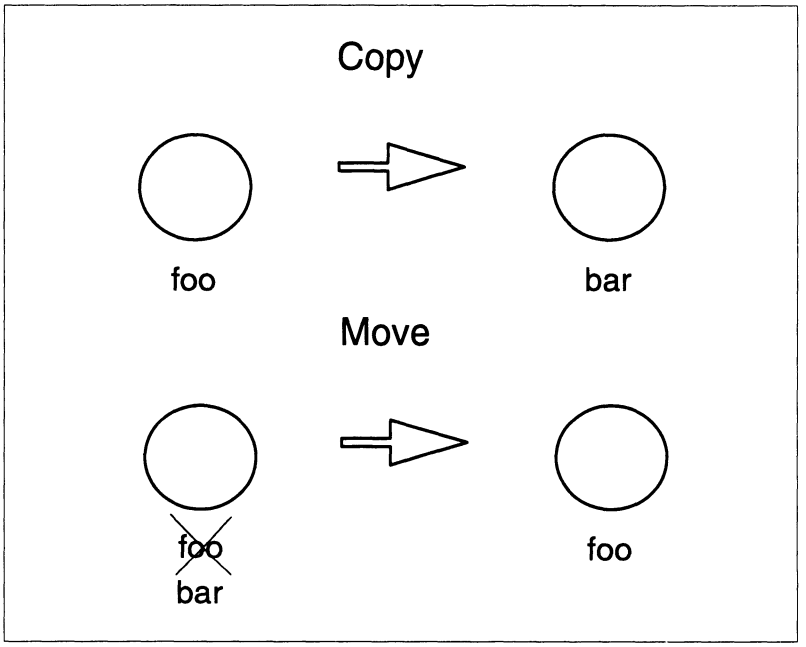


Figure 10–1 Comparison of copy and move operations

CAUTION: This operation destroys any data on the destination virtual disk.

The move operation is useful for salvaging data from a virtual disk on a failing physical disk. The system reports a message about the failing disk at the system console. While the data on the failing device is still in use, you can move its content to another virtual disk without users experiencing interruption in service. The move operation is useful also for reorganizing virtual disks on a physical disk whereby you specify their exact locations on the physical disk.

Recommendations and requirements for the move operation follow:

- The source and destination virtual disks should be the same size, although the destination may be larger, but not smaller than the source.
- The physical disks on which they reside must both be registered.
- The virtual disks must have different names.

1. To move a virtual disk, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Move
   ```

2. Select the virtual disk that you want to move.

   ```
   Source Virtual Disk:
   ```

3. Select the name of the virtual disk you want to move to. The destination virtual disk must already exist and have a name that is different from the source disk.

   ```
   Destination Virtual Disk:
   ```

4. If you wish to retain the source virtual disk with a new name, supply a new name. Otherwise, press Enter to delete the source.

   ```
   New Source Name:
   ```

5. Select a number of milliseconds for the system to wait between successive I/O operations required to complete the move.

   ```
   Throttle Value in Milliseconds: (0-300) [0]
   ```

   Choosing a non-zero value slows down the movement of data and can reduce contention for a device. The larger the number, the slower the data moves. Choosing zero causes the move operation to proceed at full speed; thus, slowing down all other users' access to the device.

Removing a virtual disk

If you no longer need a virtual disk, you can remove it to free up space.

A virtual disk to be removed must be the highest-level (or "parent") virtual disk. Removal of the top-level virtual disk deletes its children virtual disks automatically. If you attempt to remove a "child," the operation will fail. Also, you must unmount the virtual disk's file system before you remove the virtual disk.

IMPORTANT: You cannot remove a virtual disk that is currently in use as a mirror image. There are two ways to remove a mirror image: 1) unlinking an image from the mirror before removing the virtual disk, and 2) dismantling all images from a mirror before removing the virtual disk. Refer to the section in this chapter on maintaining mirrored virtual disks for more information.

1. Before you remove a virtual disk, unmount its file system.

   If you do not know the name of the file system, use this **sysadm** operation:

   ```
   File System -> Local Filesys -> List
   ```

   The listing shows virtual disks and corresponding mounted file systems.

   Refer to Chapter 11 for information on file system operations.

2. Unmount the file system using this **sysadm** operation:

   ```
   File System -> Local Filesys -> Unmount
   ```

   Refer to Chapter 11 for information on file system operations.

3. Remove the virtual disk using this **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Remove
   ```

4. Specify one or more names of virtual disks to remove:

   ```
   Virtual Disk(s):
   ```

   The system warns:

   ```
   Caution:  this operation will destroy virtual
   disk(s) <virtual-disk-name>.  Continue? [yes]
   ```

   After you confirm your intention to remove the virtual disk, it is removed.

Renaming a virtual disk

The virtual disk's file system may be mounted and in use when the virtual disk is being renamed.

1. To rename a virtual disk (including system partitions and virtual disks represented with a long name form), select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Rename
   ```

2. Supply the name of the virtual disk to rename.

   ```
   Virtual Disk to Rename:
   ```

3. Supply the name you want to change it to.

   ```
   New Virtual Disk Name:
   ```

   After you respond to the system's confirmation, the action is performed automatically.

IMPORTANT: The new virtual disk name is not transmitted automatically to the **/etc/fstab** file. You must specify a different mount point in the **fstab** file using the **sysadm** operation `File System -> Local Filesys -> Modify`. Refer to Chapter 11 for information on file system operations.

Expanding a virtual disk

IMPORTANT: This operation is useful only for virtual disks that do not contain a DG/UX file system. If you need to expand a virtual disk containing a DG/UX file system, refer to Chapter 11 for information on expanding a file system.

You cannot expand a striped virtual disk, but you can copy its contents to a larger striped virtual disk (see the next section for details).

To expand a mirror image, refer to the section on effectively expanding a mirrored virtual disk.

Use this operation to expand only partitions and aggregations of partitions. If expanding a partition and sufficient adjacent space exists for the expansion, the partition will subsume the adjacent space. Otherwise, a second partition is created elsewhere to form an aggregation. If expanding an aggregation, you will be creating another partition to include in the aggregation.

1. To increase the size of a virtual disk that does not contain a DG/UX file system, select the **sysadm** operation.

   ```
   Device -> Disk -> Virtual -> Expand
   ```

2. Select the virtual disk that you want to expand. Use ? for a list of virtual disks to choose from.

   ```
   Virtual Disk:
   ```

3. Choose the method by which to expand the virtual disk: size alone, physical disk to partition, or existing virtual disk. You expand a virtual disk by the same methods that you create a virtual disk. Type ? for choices.

   ```
   Select Space by:  [Size alone]
   ```

   `Size alone` Choose this method when you don't care which physical disk(s) the space comes from.

   `Physical disk to partition` Choose this method when you do care where the space comes from.

   `Existing virtual disk` Choose this method to combine its storage with that of the virtual disk being expanded.

   Refer to Chapter 7 for details on creating virtual disks by any of these methods.

   After you confirm your desire to expand, the operation is performed automatically.

Effectively expanding a striped virtual disk

The only way to expand the size of the striped virtual disk is to copy it to a new, larger virtual disk. The data on the virtual disk to be copied cannot be in use.

1. Unmount the virtual disk's file system using this **sysadm** operation:

   ```
   File System -> Local Filesys -> Unmount
   ```

   Refer to Chapter 11 for information on unmounting file systems.

2. Dump the contents of the virtual disk to tape using a preferred method (refer to Chapter 3 for a discussion of options).

3. Create a new striped virtual disk using this operation:

   ```
   Device -> Disk -> Virtual -> Create
   ```

   Refer to Chapter 7 for details on creating virtual disks.

4. Restore the file system from backup tape to the new virtual disk.

   Refer to Chapter 3 for information on restoring the file system from backup to the new destination.

Shrinking a virtual disk

IMPORTANT: This operation is useful only for virtual disks that do not contain a DG/UX file system. If you need to shrink a virtual disk containing

a DG/UX file system, refer to Chapter 11. If you use this procedure to shrink a virtual disk that does contain a DG/UX file system, the virtual disk is trimmed by the number of blocks specified. The data on this virtual disk will be unusable.

You cannot a shrink a virtual disk that is in use. Its file system must be unmounted or disabled.

You cannot shrink striped virtual disks.

To shrink a mirror image, refer to the section in this chapter on mirrored virtual disk operations.

Use this operation to shrink only partitions and aggregations of partitions. If shrinking a partition, you will be trimming blocks from it. If shrinking an aggregation, you may be trimming blocks from the last partition of the aggregation or you may delete entirely one of its child partitions. If shrinking a virtual disk results in only one unnamed child, the aggregation is removed, leaving only one partition which assumes the aggregation's name.

The number of blocks to be trimmed must be less than the total size of the virtual disk.

1. To shrink a virtual disk that does not contain a DG/UX file system, select the **sysadm** operation.

   ```
   Device -> Disk -> Virtual -> Shrink
   ```

2. Select the virtual disk that you want to shrink. Use ? for a list of virtual disks to choose from.

   ```
   Virtual Disk:
   ```

3. Enter the number of blocks you want to remove from the virtual disk.

   ```
   Blocks to Trim:
   ```

   The system informs you of a successful operation.

Effectively shrinking a striped virtual disk

The only way to shrink the size of the striped virtual disk is to copy it to a new, smaller virtual disk.

1. The data on the virtual disk to be backed up cannot be in use. Unmount the virtual disk's file system using this **sysadm** operation:

```
File System -> Local Filesys -> Unmount
```

Refer to Chapter 11 for information on unmounting file systems.

2. Dump the contents of the virtual disk to tape (refer to Chapter 3 for a discussion of options).
3. Create a new, smaller striped virtual disk using this operation:

```
Device -> Disk -> Virtual -> Create
```

Refer to Chapter 7 for details on creating virtual disks.

4. Restore the file system from backup tape to the new virtual disk.

Refer to Chapter 3 for information on restoring the file system from backup to the new destination.

Listing information about a virtual disk

1. To list statistics about a virtual disk, select the **sysadm** operation:

```
Device -> Disk -> Virtual -> List
```

2. List all named virtual disks, excluding system partitions, which begin with a period (.). Typing ? shows your choices.

```
Virtual Disk(s): [All_visible]
```

Choices are:

`All visible`	Presents a report that includes named virtual disks. System partitions (those beginning with a period) are excluded.
`All`	Gives a report on all visible disks and unnamed virtual disks, which result from creating an aggregation or cache with new, unnamed child partitions. Unnamed virtual disks are signified by a hexadecimal number that begins with a comma. Also, the report includes system partitions, whose names begin with a period (.).
`Virtual disks`	Presents a report for the specific virtual disks named.

3. Specify the form in which you want the report:

```
Report type: [one-line]
```

The examples in this section illustrate an aggregation named "simpson" with partitions "marge" and "homer."

The report types are:

`one-line`	Presents abbreviated list containing one line per virtual disk.
`standard`	Presents paragraph-style report on each virtual disk listed.

`recursive`	Presents information about each selected virtual disk, followed by information about its descendents.
`partitions`	Reports information about the partitions and free space on each listed virtual disk. Nameless aggregated partitions are designated by the description beneath "Role" in the form: "Piece 1 of n."
`long`	Gives an exhaustive, paragraph-style report on each listed virtual disk. Use this report style for striping information.
`parents`	Lists the name of the parent virtual disk for the specified child virtual disk. For example, if you specify child partition "marge," the parent listed will be "simpson."

An example of a one-line report follows:

```
Report Type: [one-line]
Name               Volume Temp  Size    Type
homer               V           20000   partition on sd(insc(0),0,0)
marge               V           20000   partition on sd(insc(0),0,0)
simpson             V           40000   aggregation of 2 pieces
sd(insc(0),0,0)                 1295922 physical for sd(insc(0),0,0)
```

Under the "Volume" heading, "V" represents a virtual disk that is a volume, "L" means that the virtual disk has only a long-name entry in **/dev/dsk**, which results if multiple volumes have the same name. "T" means that the virtual disk is temporary and will be deleted when the system shuts down. "F" under the "Temp" column means the virtual disk is floating (changes to virtual disk definition will be deleted when the system shuts down.)

An example of a standard report follows:

```
Virtual disk name:  homer
It is usable, in use, a volume, persistent, 20000 blocks.
Its type is:  partition on sd(insc(0),0,0).
It starts at block 621 and is 20000 blocks long
Partitioned virtual disk:
   Virtual disk name:  <no name>
   It is usable, in use, not a volume, persistent, 1295922 blocks.
   Its type is:  physical for sd(insc(0),0,0).

Virtual disk name:  marge
It is usable, in use, a volume, persistent, 20000 blocks.
Its type is:  partition on sd(insc(0),0,0).
It starts at block 20621 and is 20000 blocks long
```

```
Partitioned virtual disk:
   Virtual disk name:  <no name>
   It is usable, in use, not a volume, persistent, 1295922 blocks.
   Its type is:  physical for sd(insc(0),0,0).

Virtual disk name:  simpson
It is usable, not in use, a volume, persistent, 40000 blocks.
Its type is:  aggregation of 2 pieces.
Piece 1 of the aggregation:
   Virtual disk name:  homer
   It is usable, in use, a volume, persistent, 20000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 621 and is 20000 blocks long
Piece 2 of the aggregation:
   Virtual disk name:  marge
   It is usable, in use, a volume, persistent, 20000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 20621 and is 20000 blocks long

Virtual disk name:  sd(insc(0),0,0)
It is usable, in use, not a volume, persistent, 1295922 blocks.
Its type is:  physical for sd(insc(0),0,0).
```

An example of a recursive report follows:

```
Name                    Volume Temp  Size Type
simpson                      V      40000 aggregation of 2 pieces
   homer                     V      20000 partition on sd(insc(0),0,0)
      <no name>                    1295922 physical for sd(insc(0),0,0)
   marge                     V      20000 partition on sd(insc(0),0,0)
      <no name>                    1295922 physical for sd(insc(0),0,0)
```

An example of a partitions report follows:

```
Name                          Size        Free Comments
simpson                      40000       40000 not partitioned

   Partition Name              Role              Address         Size
   <virgin free space>                                0        40000

Name                          Size        Free Comments
sd(insc(0),0,0)           1295922     1255285

   Partition Name              Role              Address         Size
   <Various System Partitions>                        0          621
   homer                                            621        20000
   marge                                          20621        20000
   <free space>                                   40621      1255285
   <Various System Partitions>                  1295906           16
```

An example of a long report follows:

```
Virtual disk name:  homer
It is usable, in use, a volume, persistent, 20000 blocks.
Its type is:  partition on sd(insc(0),0,0).
It starts at block 621 and is 20000 blocks long
Bad block mapping is enabled.
Partitioned virtual disk:
   Virtual disk name:  <no name>
   It is usable, in use, not a volume, persistent, 1295922 blocks.
   Its type is:  physical for sd(insc(0),0,0).
   It is write-enabled.
   It has software bad block mapping facilities.

Virtual disk name:  marge
It is usable, in use, a volume, persistent, 20000 blocks.
Its type is:  partition on sd(insc(0),0,0).
It starts at block 20621 and is 20000 blocks long
Bad block mapping is enabled.
Partitioned virtual disk:
   Virtual disk name:  <no name>
   It is usable, in use, not a volume, persistent, 1295922 blocks.
   Its type is:  physical for sd(insc(0),0,0).
   It is write-enabled.
   It has software bad block mapping facilities.

Virtual disk name:  simpson
It is usable, not in use, a volume, persistent, 40000 blocks.
Its type is:  aggregation of 2 pieces.
It is striped, stripe size is 16 blocks.
Piece 1 of the aggregation:
   Virtual disk name:  homer
   It is usable, in use, a volume, persistent, 20000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 621 and is 20000 blocks long
   Bad block mapping is enabled.
Piece 2 of the aggregation:
   Virtual disk name:  marge
   It is usable, in use, a volume, persistent, 20000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 20621 and is 20000 blocks long
   Bad block mapping is enabled.

Virtual disk name:  sd(insc(0),0,0)
It is usable, in use, not a volume, persistent, 1295922 blocks.
Its type is:  physical for sd(insc(0),0,0).
It is write-enabled.
It has software bad block mapping facilities
```

Maintaining mirrored virtual disks

This chapter covers the following mirrored virtual disk maintenance operations:

- Linking one or more images to an existing mirrored virtual disk
- Unlinking one or more images from a mirrored virtual disk
- Fracturing an image from a mirrored virtual disk
- Unmirroring (dismantling) a mirrored virtual disk
- Synchronizing a virtual disk's images
- Adjusting synchronization speed (throttling)
- Effectively expanding a mirrored virtual disk
- Effectively shrinking a mirrored virtual disk
- Halting a synchronization in progress
- Modifying a mirror's attributes
- Listing a mirrored virtual disk statistics

See Chapter 7 for information on creating mirrored virtual disks.

Linking one or two images to an existing mirror

You may wish to link one or two images to an existing mirrored virtual disk under these circumstances:

- To add a third image to a two-image mirrored virtual disk for increased availability and reliability.
- To link to a mirror an image which previously had been unlinked for backup or for disk repair.
- To link an image that had been removed temporarily for backup.
- To link an image that had been removed temporarily for physical disk repair.

Recall that a mirror is restricted to three images. All images being linked must be the same size.

1. To link one or more images to an existing mirror, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Link
   ```

2. Specify the name of the mirror to link another image to.

   ```
   Existing Mirror:
   ```

3. Specify the name of an existing virtual disk to serve as another image to link to the mirror.

   ```
   Child Virtual Disk:
   ```

4. Specify whether or not you want to synchronize immediately the newly linked child image with the primary image.

   If you intend to link yet another image to the mirror, rather than synchronizing the newly linked image now, you may defer synchronization until after you have linked the next image. Then you may synchronize both images at once, which will result in better performance. Answer "no" to postpone synchronization. Go to the final prompt in this section.

   If you postpone synchronization, later you may select explicitly an operation to synchronize the newly linked secondary image with the mirror. A later section in this chapter discusses the synchronize operation.

   If you have only one image to synchronize, do so now.

   ```
   Begin Sync Immediately? [yes]
   ```

5. Select a number of milliseconds for the system to wait between successive I/O operations required to complete the synchronization. Choosing a non-zero value slows down the movement of data and can reduce contention for a device. Larger numbers slow down the synchronize operation. Choosing zero causes the synchronize operation to proceed at full speed; thus, slowing down all other users' access to the device.

   ```
   Throttle Value in Milliseconds: (0-300) [0]
   ```

6. To create another secondary image, answer "yes." **Sysadm** repeats all the prompts again to collect information about the next image. Recall that a mirror is restricted to three images. Otherwise, if you have finished linking secondary images, accept the default "no."

```
Do you want to specify another image for this mirror? [no]
```

The desired image(s) are linked to the mirror and synchronized, as appropriate.

Unlinking an image from a mirror

You may wish to unlink one or two images from an existing mirrored virtual disk under these circumstances:

- To free up disk space should availability and reliability no longer be a priority.
- To temporarily remove a mirror image from a physical disk that requires repair.

IMPORTANT: To temporarily remove a mirror image for on-line backup, use the fracture operation (covered in a later section).

Remember that when you unlink an image, you must retain the minimum images required. For example, if you originally set up a three-image mirror to contain a minimum of one image, unlinking one image will leave two functional images. Unlinking two images, however, would exceed the minimum number of images that are required.

If the image to be unlinked contains a file system, the operating system will flush all modified buffers to all mirror images before performing the unlink operation. Consequently, you can mount separately and use an unlinked mirror with assurance that the data is up-to-date.

1. To unlink (remove) an image from an existing mirror, select the **sysadm** operation

   ```
   Device -> Disk -> Virtual -> Mirrors -> Unlink
   ```

2. Specify the name of the mirror from which to unlink the image.

   ```
   Existing Mirror:
   ```

3. Select an existing virtual disk image to unlink from the mirror. Remember that at least one synchronized image must remain in the mirror.

   ```
   Image Virtual Disk(s) to Unlink:
   ```

   Following your confirmation, the operation unlinks (removes) the desired image from the mirror. The virtual disk still exists, but is not associated with the mirror.

Fracturing a mirror image for an on-line backup

Fracturing is similar to unlinking in that an image is removed temporarily from the mirror. However, while the image is removed, suspended I/O intended for the image is logged until the image is resynchronized with the mirror. You should fracture, rather than unlink, an image from an existing mirrored virtual disk to perform a backup.

Fracturing improves synchronization performance when reunited with the mirror. For example, a full-mirror synchronization of 1-Gbyte images on a striped RAID–5 CLARiiON takes about 18.5 minutes to complete. With mirror fracturing, instead of having to perform a full mirror sync, only those changed blocks (those logged) are copied to the image being synchronized. If only 30% of the

source image is modified during the time the mirror is fractured, then synchronization should take only about 6 minutes. The formula 100/*n*, where *n* is the percent of change made to an image while the mirror is fractured, represents how much faster fracturing is than a full copy resynchronization. For a 30% change during the fracturing period, resync time is three times faster than without fracturing.

If the image to be fractured contains a file system, the operating system will flush all modified buffers to all mirror images before performing the fracture. Consequently, you can mount separately and use a fractured mirror with assurance that the data is up-to-date.

1. To fracture an image from an existing mirror, select the **sysadm** operation

   ```
   Device -> Disk -> Virtual -> Mirrors -> Fracture
   ```

2. Specify the name of the mirror from which to fracture the image.

   ```
   Existing Mirror:
   ```

3. Select an existing virtual disk image to fracture from the mirror. Remember that at least one synchronized image must remain in the mirror.

   ```
   Image Virtual Disk(s) to Fracture:
   ```

   Following your confirmation, the desired image is fractured from the mirror.

4. When you fracture a mirror image, the operation creates a bitmap to log I/O intended for the fractured image. The bitmap's granularity indicates the number of blocks per bit in the bitmap. The default granularity of 128 means that one bit represents 128 blocks of data. When I/O modifies any block in the bit's range, the fracture operation turns on the bit in the bitmap.

   When you synchronize a fractured image mirror, the fracture operation copies to the fractured image only the modified (logged) blocks. Regardless of the number of blocks within the bit's range that have been modified, a default granularity of 128 causes all 128 blocks of the source image to be copied to the fractured image.

   If you anticipate high I/O activity to the mirror (over 50 percent of the mirror will be modified), choose a high granularity such as the default 128. If you anticipate low I/O activity to the mirror (less than 50 percent) during the fracture, choose a low granularity (less than the 128 default granularity). When striving for high

performance (low granularity), however, you need more memory. A 2-Gbyte mirror image with a granularity of 16 requires 128 Kbytes of memory, whereas the same mirror image with a granularity of 128 requires only 16 Kbytes.

Choose granularity according to the amount of anticipated I/O during the period in which the mirror image is fractured, the size of the mirror image that is fractured, and the amount of system resources available for creating the bitmap. Legal granularity sizes are 1–the size of the mirror image. A granularity size that is greater than the 128 default will not improve performance — the time it takes to resynchronize the fractured image with the mirror.

```
Granularity [128]:
```

Unmirroring (dismantling) a mirror

Unmirroring leaves behind one child virtual disk image, which assumes the name of the mirror. You may dismantle the mirror while it is in use, without interruption to access of the data, but you cannot unmirror while synchronization is in progress.

This operation deletes the mirror and passes on the mirror name to a mirror image that you specify. A child image's inheritance of its parent mirror's name prevents a need to remount the child image's file system. If the child did not inherit its parent's name, removal of the mirror does require that you unmount the mirror's file system and mount the child image's file system in the mirror's place.

1. To dismantle all images from a mirror, bar one, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Unmirror
   ```

2. Specify the name of the mirrored virtual disk to unmirror.

   ```
   Existing Mirror:
   ```

3. Specify the name of the child image that will inherit the parent mirror's name. Applications that are currently accessing the data on this image will use this renamed image.

   ```
   Image Virtual Disk to Get Name [mirror-name]:
   ```

   This operation removes the mirror, unlinks the images from the mirror, and assigns the parent's name to a child image. After you have dismantled a mirror, you can use its images just as you would any virtual disk.

Synchronizing a mirror's images

Under normal circumstances, the system maintains the exact same data in all images in a mirror. Under some conditions, however, one

image in a mirror may become inconsistent with the rest. An inconsistent image is considered "out of sync" or "corrupt."

IMPORTANT: A fractured image is considered out-of-sync.

When this happens, use the synchronize operation to bring the inconsistent image up-to-date. When an image is out of sync, the operating system will not use that mirror's image. If there are insufficient synchronized images, the operating system will make the entire mirror inaccessible. In this case, you must synchronize the mirror's images. You do not need to unmount a mirror to synchronize an image.

You may synchronize multiple mirror images at once but from only one primary image. It is more efficient to synchronize multiple images at once rather than separately.

You may continue to use the mirror while synchronization is in progress, however, you cannot dismantle a mirror that is being synchronized.

The image to be synchronized must be the same size as the existing mirror images.

1. To synchronize one or more images, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Synchronize
   ```

2. Specify the name of the mirror whose images to synchronize.

   ```
   Existing Mirror:
   ```

3. Select a number of milliseconds for the system to wait between successive I/O operations required to complete the synchronization. Choosing a non-zero value slows down the movement of data and can reduce contention for a device. Larger numbers slow down the synchronize operation. Choosing zero causes the synchronize operation to proceed at full speed, slowing down all other users' access to the device.

   ```
   Throttle Value in Milliseconds: (0-300) [0]
   ```

4. Specify the name of the image that will serve as the master, or primary, image for copying. The primary image is the one presumed to be up-to-date and correct.

   ```
   Master Image Virtual Disk:
   ```

5. Allow the operating system to determine the images to synchronize or specify explicitly the names of images to synchronize.

CAUTION: Be very careful when selecting the master, or primary, and secondary images for the synchronization operation. If you accidentally specify a good image as the secondary, the operation will destroy all data on the image.

```
Sync Master Image To: [All out-of-sync images]
```

The system then copies the entire master image block by block onto the secondary image(s), overwriting it entirely. The system displays a warning that the procedure will destroy the contents of the image(s) being synchronized. If you decide against resynchronization, answer "no" to continue. Otherwise, answer "yes," and the operation begins.

At the beginning and end of a synchronization session, the system logs a message using the **syslog** error logging facility. The message is from the **kern** facility and is level **warning**. By default, **kern.notice** messages go to the system console and to the file **/usr/adm/messages**. If synchronization fails, the system logs a **kern.err** message, which by default goes to the system console and to **/usr/adm/messages**.

An example of such a warning follows:

```
Oct 22 10:31:45 psycho dg/ux: Warning:  Image
'vdm(gth_image2,2CC7EE0E,0C052A9D,0)' on mirror
'vdm(gth_mirror,2CC7EE30,0C052A9D,0)' has failed (status=77005171).
```

Subsequently, when the mirror attempts to update the time stamp of both images, it gets a failure when writing to the failed one. The following warning is then displayed:

```
Oct 22 10:31:45 psycho dg/ux: Error:  Cannot store the attributes for
   virtual disk 'vdm(gth_mirror,2CC7EE30,0C052A9D,0)' on disk
   'sd(insc@7(FFF8A000,7),1,0)' (status = 77005171).
```

To change how the system handles **warning** messages, see information on logging system errors in *Managing the DG/UX™ System* and the manual page for **syslog.conf**(5).

Adjusting synchronization speed (throttling)

The throttle synchronization operation allows you to adjust the speed of a synchronization in progress by specifying a throttle value.

A throttle value is an amount of time (in milliseconds) that separates each I/O operation that comprises the synchronization operation. A throttle value of 0 means that the sync will be performed as fast as possible (I/O operations are continuous), which

can consume significant system resources. A non-zero throttle value will cause the synchronization to take longer (an amount of time separates each I/O operation), but the system as a whole will be slowed less. Releases before DG/UX 5.4R3.00 offered a slow speed for mirror synchronizations, which corresponds to a throttle value of approximately 100.

1. To change the throttle value for a synchronization operation that is in progress, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Throttle Sync
   ```

2. Specify a number of milliseconds for the system to wait between successive I/O operations required to complete the synchronization. Choosing a non-zero value slows down the movement of data and can reduce contention for a device. The larger the number, the slower the operation. Choosing zero causes the synchronize operation to proceed at full speed; thus, slowing down all other users' access to the device.

   ```
   Throttle Value in Milliseconds: (0-300) [0]
   ```

3. Specify the name of the image whose synchronization speed you wish to adjust.

   ```
   Ongoing Sync: [image-name]
   ```

   After you confirm your desire to change the synchronization speed, the system performs the operation and displays a message, such as the following:

```
Synchronization of image virtual disk "foobar-1" throttled.
```

Halting a synchronization in progress

1. To halt a synchronization that is in progress, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Halt Sync
   ```

   The image whose synchronization you halt is unusable.

2. Specify the name of the image whose synchronization you wish to terminate.

   ```
   Ongoing Sync: [image-name]
   ```

   The system performs the operation and displays a message, such as the following:

   ```
   Synchronization of image virtual disk "foobar-1"
   terminated.
   ```

Effectively expanding a mirrored virtual disk

The only way to expand the size of the mirror is to disassemble the images from the mirror and expand each mirror image. Afterward, you can reassemble the expanded images into a mirror.

During this operation, you may continue to access the data on the virtual disks forming the mirror, but without the benefits of mirroring until you reassemble the mirror.

1. Dismantle each image from the mirror using this **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Unmirror
   ```

   See the section in this chapter on unmirroring.

2. If the virtual disk being expanded contains a DG/UX file system, skip this step and go to step 3. Otherwise, to increase the size of a virtual disk that does not contain a DG/UX file system, select the **sysadm** operation.

   ```
   Device -> Disk -> Virtual -> Expand
   ```

   Refer to the section in this chapter on expanding a virtual disk.

3. To expand a virtual disk containing a DG/UX file system, select the **sysadm** operation:

   ```
   File -> Local Filesys -> Expand
   ```

   Refer to Chapter 11 for information on expanding a file system.

4. Reassemble and resynchronize the mirror's images using these mirroring operations:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Link

   Device -> Disk -> Virtual -> Mirrors -> Synchronize
   ```

   Refer to the section in this chapter on mirroring virtual disks for details.

Effectively shrinking a mirrored virtual disk

The only way to shrink the size of the mirror is to copy it to a new, smaller virtual disk.

1. The data on the virtual disk to be copied cannot be in use. Unmount the virtual disk's file system using this **sysadm** operation:

   ```
   File System -> Local Filesys -> Unmount
   ```

   See Chapter 11 for information on unmounting file systems.

2. Dismantle each image from the mirror using this **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> Unmirror
   ```

   See the section in this chapter on unmirroring.

3. Dump the contents of one mirror image (known to be up-to-date) to tape (refer to Chapter 3 for a discussion of options).

4. Create new, smaller virtual disks to be used as mirror images using this operation:

   ```
   Device -> Disk -> Virtual -> Create
   ```

   See Chapter 7 for details on creating virtual disks.

5. Restore the file system from backup tape to the virtual disk to be used the master image.

   See Chapter 3 for information on restoring the file system from backup to the new destination.

6. Rebuild the mirror virtual disk from the newly sized virtual disks you just created.

   See Chapter 7 for details on building a mirror.

Modifying a mirror's attributes

To alter mirror attributes, select the **sysadm** operation `Device -> Disk -> Virtual -> Mirrors -> Modify`.

After you select this operation, **sysadm** displays the same prompts that are delivered when mirroring a virtual disk.

See the section on creating a virtual disk mirror in Chapter 7 for explanations of each prompt.

Listing a mirror's statistics

1. To list information about disk mirrors and their images, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Mirrors -> List
   ```

2. Specify all virtual disk mirrors or the name(s) of those you wish to see.

   ```
   Mirrored Virtual Disks to List: [all]
   ```

3. Specify whether or not you wish to see a mirror's images in addition to the mirror itself. Answering **no** lists only the mirror; **yes** lists the mirrors' constituent images as well.

   ```
   List Images? [no]
   ```

A typical simple mirror listing follows:

```
Name                    Volume  Temp   Size Type
wrightbros                   V        10000 mirror with 2 images (1 good)
sam                          V         2100 mirror with 2 images (2 good)
```

Valid `Volume` values are:

`V`	Represents a virtual disk that is a volume.
`L`	Signifies that the virtual disk has only a long-name entry in **/dev/dsk**.
`T`	Signifies that the virtual disk is temporary and will be lost on system shutdown.
`F`	Means the virtual disk is floating (changes to virtual disk definition will be lost on system shutdown).

The display also includes the size of the mirror (that is, the usable size, not the cumulative size of the component images), and the number of images present. The display also includes the status of the mirror and its images, including the sync status of any images: whether the image is synchronized or corrupt and, if synchronization is in progress, the percentage of synchronization complete and the name of the image acting as the sync master.

A typical full (mirror and images) listing follows:

```
Virtual disk name:  wrightbros
It is usable, not in use, a volume, persistent, 10000 blocks.
Its type is:  mirror with 2 images (1 good).
Minimum required image count is 1, maximum lost image count is 0.
Auto-sync on system boot is enabled, throttle is 0 msec.
Image 1 of the mirror:
   It is in sync.
   Virtual disk name:  orville
   It is usable, in use, a volume, persistent, 10000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 24447 and is 10000 blocks long
   Bad block mapping is enabled.
Image 2 of the mirror:
   It is out of sync.
   Virtual disk name: wilbur
   It is usable, in use, a volume, persistent, 10000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 34447 and is 10000 blocks long
   Bad block mapping is enabled.
```

Maintaining cached virtual disks

This section covers the following cache maintenance operations:

- Linking one or more front-end devices to an existing cache
- Unlinking one or more front-end devices from an existing cache
- Uncaching (dismantling) a cache
- Effectively expanding a cache
- Effectively shrinking a cache
- Modifying a cache's attributes
- Listing a cache's statistics

See Chapter 7 for information on creating a cache.

Linking one or more front-end devices to an existing cache

You may wish to link more front-end devices to a cache to increase throughput. There is no limit on the number of front-end devices you can add to a cache.

1. To link one or more front-end devices (existing virtual disks on fast devices) to an existing cache, perform the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Caches -> Link
   ```

2. Specify the name of the existing cache to which front-end devices get linked.

   ```
   Existing Cache:
   ```

3. Specify the name of the virtual disk to serve as a fast, front-end device. You must select a pre-existing virtual disk. Type ? for a list of choices.

   ```
   Front End Virtual Disk:
   ```

4. Specify whether or not you want to add another front-end device to the cache. If you answer "yes," the preceding prompt repeats. Otherwise, no more prompts are delivered, and the selected number of front-end devices are linked to the cache.

   ```
   Specify Another Front End? [no]
   ```

Unlinking one or more front-end devices from an existing cache

1. To unlink (remove) one or more front-end devices from an existing cache, select the **sysadm** operation.

   ```
   Device -> Disk -> Virtual -> Caches -> Unlink
   ```

   When the front-end device is unlinked, all data in the front-end device is flushed to the back-end device automatically.

2. Specify the cache from which to unlink the front-end device.

   ```
   Existing Cache:
   ```

3. Specify the name of the front-end device to be unlinked. You must select a pre-existing virtual disk. Type ? for a list of choices.

   ```
   Front End Virtual Disk:
   ```

4. Specify whether or not you want to unlink another front-end device from the cache. If you answer "yes," the preceding prompt repeats. Otherwise, no more prompts are delivered, and the selected number of front-end devices are unlinked from the cache.

   ```
   Specify Another Front End? [yes]
   ```

   Following your confirmation, the desired front-end device is unlinked (removed) from the cache. The virtual disk still exists, but is not associated with the cache.

Uncaching (dismantling) a cache

1. To dismantle a cache (unlink its front-end devices), restoring the cache's name to the back-end virtual disk and remove the cache itself, select the **sysadm** operation.

   ```
   Device -> Disk -> Virtual -> Caches -> Uncache
   ```

   All data cached in the front end is first flushed to the back end before the system dismantles the cache. The operation deletes the cache, leaving behind its child virtual disks for use as file systems or by the application. The former back-end device resumes its original name — the cache's name.

2. Specify the cache to dismantle.

   ```
   Existing Cache:
   ```

   Following your confirmation, the desired cache is dismantled. The virtual disks still exist, but are not associated with the cache. After you have dismantled a cache, you can use its virtual disks just as you would any virtual disks.

Effectively expanding a cached virtual disk

The only way to expand the size of the cache is to dismantle the cache's front- and back-end devices and expand the back-end device. The expanded back-end device can then be linked to the front-end device.

During this operation, you may continue to access the data on the back-end device, but without the benefit of caching until the cache is reassembled.

1. Dismantle the cache using this **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Caches -> Uncache
   ```

   See the section in this chapter on uncaching.

2. If the back-end virtual disk being expanded contains a DG/UX file system, skip this step and go to step 3. Otherwise, to increase the size of a back-end virtual disk that does not contain a DG/UX file system, select the **sysadm** operation.

   ```
   Device -> Disk -> Virtual -> Expand
   ```

   See the section in this chapter on expanding a virtual disk.

3. To expand a virtual disk containing a DG/UX file system, select the **sysadm** operation:

   ```
   File -> Local Filesys -> Expand
   ```

   See Chapter 11 for information on expanding a file system.

4. Re-create the cache using the front-end devices that you previously uncached and the expanded virtual disk to be used as the back–end device.

   ```
   Device -> Disk -> Virtual -> Caches -> Cache
   ```

   See the section in this chapter on caching virtual disks.

Effectively shrinking a cached virtual disk

The only way to shrink the size of the cache is to copy it to a new, smaller virtual disk.

1. The data on the virtual disk to be copied cannot be in use. Unmount the virtual disk's file system using this **sysadm** operation:

   ```
   File System -> Local Filesys -> Unmount
   ```

   See Chapter 11 for information on unmounting file systems.

2. Dismantle the cache using this **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Caches -> Uncache
   ```

   See the section in this chapter on uncaching.

3. Dump the contents of the virtual disk being used as the back-end device to tape (refer to Chapter 3 for a discussion of options).

4. Create a new, smaller virtual disk to be used as a back-end device using this operation:

   ```
   Device -> Disk -> Virtual -> Create
   ```

   See Chapter 7 for details on creating virtual disks.

5. Restore the file system from backup tape to the virtual disk to be used the back-end device.

   Refer to Chapter 3 for information on restoring the file system from backup to the new destination.

6. Create the cache using the front-end device(s) that you previously uncached and the newly sized virtual disk to be used as the back-end device.

   ```
   Device -> Disk -> Virtual -> Caches -> Cache
   ```

   Refer to Chapter 7 for details on creating a software disk cache.

Modifying a cache's attributes

To alter cache attributes, select the **sysadm** operation `Device -> Disk -> Virtual -> Caches -> Modify`.

See the section on creating a cache in Chapter 7 for explanations of each prompt.

Listing cache statistics

1. To list information about caches and their front- and back-end devices, select the **sysadm** operation:

   ```
   Device -> Disk -> Virtual -> Caches -> List
   ```

2. Select all caches or specify by name those you wish to see.

   ```
   Cached Disk(s) to List: [all]
   ```

3. Specify whether or not you want to list front-end devices as well as back-end devices. Answering "no" lists only the caches; "yes," the caches' front and back ends.

   ```
   List Front and Back Ends? [no]
   ```

When you list caches only, the information displayed is a subset of the attributes you gave the cache when you created it. Besides the name, the "Volume" value specifies whether or not it has been made into a volume.

Valid `Volume` values are:

`V`	Represents a virtual disk that is a volume.
`L`	Signifies that the virtual disk has only a long-name entry in **/dev/dsk**.
`T`	Signifies that the virtual disk is temporary and will be lost on system shutdown.
`F`	Means the virtual disk is floating (changes to virtual disk definition will be lost on system shutdown).

It also includes the size of the front-end device used in the cache, and the back-end device name.

A typical listing for caches only follows:

```
Name                    Volume Temp  Size Type
foo                          V        100 cache for zoo,2CBDD094
home                         V        200 cache for home-back
notes                        V      40000 cache for notes-back
```

The listing for the cache and its front- and back-end devices is quite lengthy, including all the attributes you specified when creating it. A typical example follows:

```
Virtual disk name:  notes
It is usable, not in use, a volume, persistent, 40000 blocks.
Its type is:  cache for notes-back.
The cache is available.
Fraction of reads coming from a front end device (read hits): n/a
Fraction of writes going to a front end device (write hits): n/a
Caching reads, caching writes.
Not caching only file system metadata.
First write to cache triggers an asynchronous write to the back end.
Use of direct memory access by front-end device(s) is enabled.
Read retention weight: 1
Write retention weight: 1
Percentage of front-end(s) to search for available buffer: 10%
Flusher type:  none.
Read count: 0
Write count: 0
Read miss count: 0
```

```
Write miss count: 0
Read hit count: 0
Write hit count: 0
Read purge count: 0
Write purge count: 0
Allocate purge count: 0
Back end read count: 0
Back end write count: 0
Total buffers count: 1408
Total data blocks count: 46008
Dirty buffers count: 0
Allocated buffers count: 0
Total data blocks count: 46008
State stall count: 0
Flush stall count: 0
Cache back end:
   Virtual disk name:  notes-back
   It is usable, in use, a volume, persistent, 40000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 44747 and is 40000 blocks long
   Bad block mapping is enabled.
Cache front end 1:
   Virtual disk name:  notes-front
   It is usable, in use, a volume, persistent, 40000 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 84747 and is 40000 blocks long
   Bad block mapping is enabled.
Cache front end 2:
   Virtual disk name:  foolish
   It is usable, in use, a volume, persistent, 6100 blocks.
   Its type is:  partition on sd(insc(0),0,0).
   It starts at block 18347 and is 6100 blocks long
   Bad block mapping is enabled.
```

End of Chapter

11 Maintaining file systems

Physical and virtual disk oeprations differ from file system operations. For example, to expand or shrink a virtual disk, if it contains a DG/UX file system, you must instead expand or shrink the DG/UX file system. The file maintenance operations largely control the status of a particular file system — whether or not it is available for local or remote use, or whether or not it is available for continued use.

This chapter covers maintenance of local and remote file systems.

Maintaining local file systems

This chapter covers these local file system operations:

- Deleting a local file system
- Expanding a local DG/UX file system
- Shrinking a local DG/UX file system
- Modifying a local file system
- Mounting and unmounting a file system
- Exporting and unexporting a file system
- Listing file systems

For information on creating a DG/UX file system, creating a file system mount point, and using the **mkfs** command to make a file system for a diskette, see Chapter 8.

For information on backing up and restoring file systems, see *Managing the DG/UX™ System*.

Deleting a local file system

The delete operation removes one or more file systems from the file system table, **/etc/fstab**. Removal of a file system's entry in this file makes the file system inaccessible at boot time. Until boot time, however, the file system may still be accessible, if mounted. You may unmount the file system as well, which renders the file system unavailable now and at boot time. You may choose to keep the file system mounted until the next system boot. Deleting a file system without unmounting it does not interrupt any work sessions currently using the file system. If, however, you do elect to

unmount while the file is in use, the operation displays a warning message. You cannot unmount a file system while users are using it.

Deleting a file system does not destroy it. The data remains intact. You can make the file system again by mounting it.

If, when you created the file system, you identified the file system as exportable, the delete operation also removes the file name entries from the **/etc/exports** file, making the file unavailable for remote file system mounts.

You cannot unmount a `ramdisk` type file system if it contains any files or directories. Unmounting a `ramdisk` type file system removes the associated physical device entry in **/dev**.

1. To delete a file system from the **/etc/fstab** file, select the **sysadm** operation:

   ```
   Filesys -> Local Filesys -> Delete
   ```

2. Supply the names of the file system(s) to delete. Type ? for a list of mounted file systems. An example follows:

   ```
   File System(s) to Delete: /usr/opt/fredware
   ```

3. Specify whether or you wish to unmount the file system (make it inaccessible immediately) as well.

   ```
   Unmount after deleting? [yes]
   ```

   Answering "yes" makes the file system inaccessible immediately; "no" allows continued use of the file system until the next time the system boots. If the file system currently is in use, the file system cannot be unmounted.

   The operation asks for confirmation before deleting the file system entry.

Expanding a DG/UX file system

Keeping track of the size of your DG/UX file systems ensures good I/O performance. When a file system becomes 80% full, I/O performance begins to decline. When a file system becomes 90% full, operations requiring more space (such as creating new files or directories or increasing the size of a file) will fail. The system alerts you to such a failure with a message such as `no space left on device`. Such an event may require that you expand the full file system. Refer to *Managing the DG/UX™ System* for information on monitoring the size of your file systems.

IMPORTANT: The file system to be expanded must be a DG/UX file system on a virtual disk. You cannot expand a non-DG/UX file system nor can you expand a DG/UX file system residing directly on a physical disk, such as a DG/UX file system occupying an entire diskette. Furthermore, you cannot expand a striped virtual disk.

The operation expands the size of the underlying virtual disk, extending the DG/UX file system into the expanded area. The file being expanded may be mounted and on-line (in use) during this operation.

To expand a mirror, cache, or a virtual disk that does not contain a DG/UX file system, see Chapter 10 for instructions.

1. To increase the size of a file system, select the **sysadm** operation:

```
File System -> Local Filesys -> Expand
```

2. Specify the mount point directory for the file system being expanded; for example, **/usr/opt/fredware**.

```
File System to Expand:
```

3. You expand a file system by the same method used for creating a virtual disk. Refer to Chapter 7 for a discussion of the methods for creating a virtual disk.

```
Select Space by: [Size alone]
```

Keep in mind that file system internal data structures and the free space requirement will use some of the space that you are adding to the file system. Consequently, you should add 10% more space than the amount you intend to use. For example, if you need 30,000 more blocks of usable space in a file system, you should add 30,000 + (10% of 30,000) blocks, or 33,000 blocks.

The expand operation maintains the required percentage of free space for the file system. By default, a file system has a free space requirement of 10%.

Sysadm informs you that the virtual disk and DG/UX file system were expanded, as desired.

Expanding the / (root) and /usr file systems

To expand the root (/) or **/usr** file systems, remember that you cannot boot from a file system built on a virtual disk that spans multiple physical disks. You need to boot from the root file system because it contains your kernel, **/dgux**. You need to boot from the the **/usr** file system because it contains stand-alone **sysadm**,

/usr/stand/sysadm, which you may need to boot to recover after a failure. To increase the size of either of these file systems, ensure that the physical disk housing the / or **/usr** file system contains sufficient space for the creation of another partition. The expansion generally results in an aggregated virtual disk. You may perform this operation while the / and **/usr** file systems are mounted and the system is on-line and in use.

Shrinking a file system

IMPORTANT: The file system to be shrunk must be a DG/UX file system on a virtual disk. You cannot shrink a non-DG/UX file system nor can you shrink a DG/UX file system residing directly on a physical disk, such as a DG/UX file system occupying an entire diskette. Furthermore, you cannot shrink a striped virtual disk.

The operation shrinks the size of the underlying virtual disk and, correspondingly, the DG/UX file system. The file system being shrunk must be unmounted and off-line (not in use) during this operation.

The operation selects which blocks to remove based on internal criteria. The shrink operation maintains the required percentage of free space for the file system. By default, a file system has a free space requirement of 10%.

Shrinking a file system requires the operation to rearrange data within the file system, collecting unallocated blocks so that the system may free the desired number. If you specify that the shrink operation recover more space than is available from the file system, it returns an error.

To shrink a mirror, cache, or a virtual disk that does not contain a DG/UX file system, see Chapter 10 for instructions.

1. Unmount the DG/UX file system to be shrunk.

   ```
   File System -> Local Filesys -> Unmount
   ```

   Refer to a later section on unmounting a DG/UX file system for details.

2. To decrease the size of a file system, select the **sysadm** operation:

   ```
   File System -> Local Filesys -> Shrink
   ```

3. Specify the mount point directory for the file system to be shrunk; for example, **/usr/opt/fredware**.

   ```
   File System to Shrink:
   ```

4. Enter the number of 512-byte blocks by which to shrink the file system.

```
Number of blocks by which to shrink:
```

5. You may set the minimum percentage of the file system that should remain empty after the shrinking operation.

```
Percentage of blocks in the remaining DARs to keep unallocated:
(0-100) [20]
```

This figure refers to the percentage of unallocated blocks per DAR (Disk Allocation Region) The default is 20%. If the shrink operation cannot free the desired number of blocks while leaving this percentage of remaining space free, the operation fails.

6. You may set the minimum percentage file node slots that should remain available per DAR after the shrinking operation.

```
Percentage of file node slots in the remaining DARs to keep
available: (0-100) [20]
```

A DAR can hold a fixed number of files and directories (file nodes), and this percentage determines how many of them will remain available for directory or file creation. The default is 20%.

Sysadm informs you that the virtual disk and DG/UX file system were expanded, as desired.

If the shrink operation cannot free the desired number of blocks while leaving the desired percentage of file node slots available, the operation fails.

Under some conditions, the shrink operation will reduce the file system more than the requested number of blocks. This happens when the shrink operation causes the new end of the file system to fall in a DAR's metadata area (file node table or free space bitmap). The operation detects this condition and shrinks the file system further, to the beginning of the DAR. This condition does not constitute an error or a problem. When it occurs, the operation prints the following message:

```
The last DAR size was less than the minimal needed to support
the file node table and the DAR bitmap.  The new size of the
file system has been adjusted to nnnn blocks.
```

7. Remount the shrunk DG/UX file system.

```
File System -> Local Filesys -> Mount
```

Refer to a later section in this chapter on mounting a DG/UX file system for details.

For a more detailed discussion of the file system internals, see the **mkfs**(1M), **fs**(4), and **tunefs**(1M) manual pages.

Modifying a local file system

This operation changes the attributes of a local file system which you selected when creating the file system's mount point through the **sysadm** operation `File System -> Local Filesys -> Add`. The attributes you can change depend on the type of file system being modified.

After you have specified how you wish to modify the file system, **sysadm** asks if you want to apply the modifications immediately. If you choose to do so, the operation attempts to remount and then unexport or export (as appropriate) the file system so that the changes will be effective immediately.

To modify an attribute of a file system, select the **sysadm** operation:

```
File System -> Local Filesys -> Modify
```

See the section on creating a mount point for a local file system in Chapter 8 for details.

Mounting and unmounting a local file system

You must mount a file system to make it accessible to users. Creating a mount point for a file system (`File System -> Local Filesys -> Add`) and mounting a file system (`File System -> Local Filesys -> Mount`) are different operations. Creating a mount point places an entry in the **/etc/fstab** file, which the system consults at boot time; mounting a file system (which puts an entry in the **/etc/mnttab** file) gives users immediate access to that file system. The contents of these files do not have to be identical. You may have a file system currently mounted for use (it is listed in **/etc/mnttab**), but as soon as the system reboots (**/etc/fstab** lists file systems to be mounted at boot time), that file system will be unavailable.

Some physical disk, virtual disk and file system operations require that the file system be unmounted. Examples are disk copy, remove virtual disk, shrink virtual disk, and shrink file system. After the operation has completed, you must remount the file systems. Check the procedure to find out its file mount requirements.

Remote users to whom you have given access to the file system cannot mount it on their systems until you mount it locally on your own system.

1. Mount a file system with the **sysadm** operation:

   ```
   File System -> Local Filesys -> Mount
   ```

2. Supply the name of the mount point directory to mount:

   ```
   File System(s) to Mount:
   ```

   When the Mount operation prompts you for the file system to mount, specify the name of the mount point directory. If there are no local file systems in your file system table that are currently unmounted, the operation reports this condition before terminating. If an attempt to mount a file system fails, it may be because the file system is corrupted. Verify the integrity of the file system with `File System -> Local Filesys -> Check`. Refer to a later section in this chapter on file system checking.

1. Likewise, to unmount a file system, use the following **sysadm** operation :

   ```
   File System -> Local Filesys -> Unmount
   ```

2. Supply the name of the mount point directory to unmount:

   ```
   File System(s) to Unmount:
   ```

   You cannot unmount a file system that is in use. A file system is in use if it contains the working directory for any user or running process, or if it contains a program that is currently running or is open for reading or writing by a program that is currently running.

Exporting and unexporting a local file system

You may use these operations to make a local file system accessible or inaccessible to other systems on your network until the system reboots.

You can export only file systems that you have made exportable either through the add operation or the modify operation (see the sections on adding and modifying file systems in other sections in this chapter). If you choose to export a file system, the operation enters automatically the file system for export in the **/etc/exports** file, which the system consults each time the system boots. The add and modify operations also make an entry in **/etc/xtab**, which contains a list of currently exported file systems. The listed files are immediately accessible.

To unexport a file system, you may use the **unexport** operation. Unexporting a file removes its entry from the **/etc/xtab** file, making

it immediately unavailable. However, the next time the system boots, all file systems listed in **/etc/exports** will be exported for use on remote systems. The only way to make a permanent change to the **/etc/exports** file is to use the modify operation. The contents of these files do not have to be identical.

1. To unexport a local file system, use the **sysadm** operation:

```
File System -> Local Filesys -> Unexport
```

2. Specify the file systems to be unexported.

```
File System(s) to Unexport:
```

The specified file system is removed from the **/etc/xtab** file, making the file system unavailable immediately. However, the next time the system boots, file systems listed in the **/etc/exports** file are exported for use on remote systems.

Alternatively, you may use the **export –u** *file-system-name* shell command.

1. To export a local file system that had been unexported previously, use the **sysadm** operation:

```
File System -> Local Filesys -> Export
```

2. Specify the file system(s) to be exported.

```
File System(s) to Export:
```

The specified file system is added to the **/etc/xtab** file, making the file system available immediately. However, the next time the system boots, file systems listed in the **/etc/exports** file are exported for use on remote systems.

Alternatively, you may use the **export** *file-system-name* shell command.

Listing local file statistics

1. To list statistics about mounted local files (listed in **/etc/fstab**), select the **sysadm** operation:

```
File System -> Local Filesys -> List
```

2. Specify which entries you want to list:

```
Which local file systems: [all]
```

Choices are:

`all`	Lists all local file systems (from **/etc/fstab**).
`mounted`	Lists mounted local file systems (from **/etc/mnttab**).
exported	Lists exported file systems (from **/etc/xtab**).

A listing of all file systems follows:

```
                                          FS            NFS   Dump  Fsck
File System Source      Mount Directory   Type       RW Mount Frq  Pass
------------------      ---------------   ----       -- ----- ----- ----
/dev/dsk/root           /                 dg/ux      rw        d     0
/dev/dsk/usr            /usr              dg/ux      rw        d     0
/dev/dsk/usr_opt_x11    /usr/opt/X11      dg/ux      rw        d     1
/dev/dsk/usr_opt_aview  /usr/opt/aview    dg/ux      rw        d     1
/dev/ram2               /ram_dir/ram2     dg/ux(ram) rw        x     0
```

The columns in the display correspond to the columns in the **fstab** file. The `File System Source` column shows the physical device pathname. The `Mount Directory` column shows the mount point directory. The `FS Type` column shows the file system type, where the values are:

`dg/ux`	A typical disk file system.
`dg/ux(ram)`	A `ramdisk` (memory) file system.
`cdrom`	The file system on a CD-ROM drive.
`dos`	An MS-DOS file system, created by **mkfs** with the **dos** flag.

The `RW` column gives the write permissions, either `rw` (read/write) or `ro` (read only). The `NFS Mount` column applies only to remote file systems mounted via ONC/NFS. The field is blank for a local file system. The `Dump Freq` column gives the backup frequency, where possible values are:

`d`	Backup during daily, weekly, and monthly backups.
`w`	Backup during weekly and monthly backups.
`m`	Backup during monthly backups.
`x`	Do not back up.

The `Fsck Pass` column shows during which pass the file system checker, **fsck**, will check the file system. The value may be `0` through `9`.

For more information on the various **fstab** fields, see the section on the file system table in Chapter 8.

Maintaining remote file systems

This section covers these remote file system operations:

- Deleting a remote file system
- Modifying a remote file system
- Mounting and unmounting a remote file system
- Listing remote file systems

See Chapter 8 for information on remote file systems that appear in the **/etc/fstab file** and creating a mount point for a DG/UX file system.

To have access to a remote system's file systems, both the local and remote systems must have the ONC/NFS network package installed and set up. For more information, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

Deleting a remote file system

The delete operation removes one or more remote file systems from the file system table, **/etc/fstab**. Removal of a file system's entry in this file makes the file system inaccessible at boot time. Until boot time, however, the file system may still be accessible, if mounted. You are given an opportunity to unmount the file system as well, which renders the file system unavailable now and at boot time. You may choose to keep the file system mounted until the next system boot. Deleting a file system without unmounting it does not interrupt any work sessions currently using the file system. If, however, you do elect to unmount and the file currently is in use, the operation displays a warning message. You cannot unmount a file system while users are using it.

Deleting a remote file system does not destroy it. The data remains intact. If you later decide to make the file system available again, mount it.

1. To delete a remote file system from the **/etc/fstab** file, select the **sysadm** operation:

   ```
   Filesys -> Remote Filesys -> Delete
   ```

2. Supply the names of the file system(s) to delete. Type ? for a list of mounted file systems. An example follows:

   ```
   File System(s) to Delete: /usr/opt/fredware
   ```

3. Specify whether or you wish to unmount the file system (make it inaccessible immediately) as well.

```
Unmount after deleting? [yes]
```

Answering "yes" makes the file system inaccessible immediately; "no" allows continued use of the file system until the next time the system boots. If the file system currently is in use, the remote file system cannot be unmounted.

The operation asks for confirmation before deleting the file system entry.

Modifying a remote file system

This operation changes the attributes of a remote file system which you selected when creating the file system's mount point through the **sysadm** operation `File System -> Remote Filesys -> Add`. The attributes you can change depend on the type of file system being modified.

After you have specified how you wish to modify the file system, **sysadm** asks if you want to apply the modifications immediately. If you choose to do so, the operation attempts to remount the file system so that the changes will be effective immediately.

To modify an attribute of a remote file system, select the **sysadm** operation:

```
File System -> Remote Filesys -> Modify
```

For details, see the section on creating a mount point for a remote file system in Chapter 8.

Mounting and unmounting a remote file system

You must mount a remote file system to make it accessible to users. Creating a mount point for a remote file system and mounting a remote file system are different operations. You create a mount point with the **sysadm** operation `File System -> Remote Filesys -> Add`. You actually may mount a remote file system using either of two methods: answer "yes" when prompted to mount the remote file system as part of the add operation use the **sysadm** explicit mount operation `File System -> Remote Filesys -> Mount`.

Creating a mount point places an entry in the **/etc/fstab** file, which the system consults at boot time; mounting a remote file system (which puts an entry in the **/etc/mnttab** file) gives users immediate

access to that file system. The contents of these files do not have to be identical. You may have a remote file system currently mounted for use (it is listed in **/etc/mnttab**), but as soon as the system reboots (**/etc/fstab** lists remote file systems to be mounted at boot time), that remote file system will be unavailable.

IMPORTANT: Before you can mount a remote file system, the remote host must export that file system using this **sysadm** operation: `File System -> Local Filesys -> Export`. Refer to the section on exporting local file systems in this chapter for more information.

If you unmounted a remote file system, you have to mount it to make the file system accessible to users.

1. Mount a remote file system with the **sysadm** operation:

   ```
   File System -> Remote Filesys -> Mount
   ```

2. Supply the name of the mount point directory to mount:

   ```
   File System(s) to Mount:
   ```

   When **sysadm** prompts for the file system to mount, specify the name of the mount point directory, the second field in the **/etc/fstab** file. If there are no local file systems in your **/etc/fstab** file that are currently unmounted, the operation reports this condition before terminating. If an attempt to mount a file system fails, it may be because the file system is corrupted. The remote file's integrity must be checked on the remote host. Refer to a later section on file system checking in this chapter.

   If you know that the remote system is going to be shut down, or the file system on the remote system is going to be unexported or unmounted, then you should unmount the remote file systems on your local system. If you do not unmount, then some local processes might hang trying to access the remote file system.

   You cannot unmount a file system that is in use. A file system is in use if it contains the working directory for any user or running process, or if it contains a program that is currently running or is open for reading or writing by a program that is currently running.

1. Unmount a remote file system with the **sysadm** operation:

   ```
   File System -> Remote Filesys -> Unmount
   ```

2. Supply the name of the mount point directory to unmount:

   ```
   File System(s) to Unmount:
   ```

If your system's attempt to communicate with the remote system fails, your system will retry the mount request a number of times, possibly causing the mount operation to appear to hang. For more information on the **mount** command and how it handles request retries, see the **mount**(1M) manual page.

As an alternative to using **sysadm**, you may use the **mount** *remote-filename* command or the **mount –u** *remote-filename* command, as appropriate.

Listing remote file statistics

1. To list statistics about mounted remote files (listed in **/etc/fstab**), select the **sysadm** operation:

   ```
   File System -> Remote Filesys -> List
   ```

2. Specify which entries you want to list:

   ```
   Which local file systems: [all]
   ```

   Choices are:

`all`	Lists all local file systems (from **/etc/fstab**).
`mounted`	Lists mounted local file systems (from **/etc/mnttab**).

   A listing of all file systems follows:

```
                                          FS            NFS   Dump  Fsck
File System Source      Mount Directory   Type       RW Mount Frq  Pass
------------------      ---------------   ----       -- ----- ----- ----
sales03:/pdd/sales03    /pdd/sales03      nfs        rw hard   x     0
sales03:/sales/accounts /sales/accounts   nfs        rw hard   x     0
sales01:/udd/sales01    /udd/sales01      nfs        ro soft   x     0
```

   The columns in the display correspond to the columns in the **fstab** file:

   `File System Source`
   The mount point directory name on the remote host where the file system resides.

   `Mount Directory`
   The mount point directory on the local system.

   `FS Type` The file system type, where the value for a remote file system is always `nfs`.

`RW`	The write permissions mode, either `rw` (read/write) or `ro` (read only).
`NFS Mount`	How the file system is mounted, `hard` or `soft`.
`Dump Frq`	The backup frequency. For a remote file system, this field should always be `x` because you only back up local file systems.
`Fsck Pass`	The pass in which the file system checker, **fsck**, will check the file system. For remote file systems, this value should always be `0` because you check only local file systems.

For more information on the various **fstab** fields, see the section on the file system table in Chapter 8.

End of Chapter

12 File system checking

The file system checking procedure verifies the internal structure of a file system that you suspect is corrupt.

What makes a file system corrupt?

A file system may become corrupt as a result of a power outage, the physical relocation of the hardware on which the file system resides, a disk head crash that causes the system to halt unexpectedly, or an abnormal shutdown procedure that may cause service to the file system to terminate unexpectedly.

Typical examples of abnormal shutdown procedures are:

- Not using the **shutdown** command before halting the CPU
- Taking a mounted file system off line (powering it down without first unmounting it)

Such events can cause a file system's metadata to become inconsistent or the file system's data to be lost. Sometimes a file system may exhibit a behavior that makes you suspect that its data has become corrupt. A file system can show signs of corruption both during normal operation and during system booting. The following are typical symptoms of a corrupt file system:

- Following a system crash, difficulty mounting a file system during booting
- A system hang (the cursor disappears) when you try to access a locally mounted file that you have accessed successfully in the past

Checking and repairing file systems after a system failure

By default, the system invokes the **fsck** utility (pronounced f–s–c–k) to check the / file system each time the system reboots.

If the system failure damaged files that are necessary for bringing up the system, **fsck** may fail. If this happens, see "Repairing damaged DG/UX system files."

On a reboot, after the / file system has been checked, local file systems are checked according to their **fsck** pass numbers.

IMPORTANT: To review each file system's pass number use the **sysadm** operation `File System -> Local Filesys -> List` (see Chapter 11). To speed up local file system checking, you may elect fast recovery **fsck** through the **l** option to the **fsck** command, which is covered later in this chapter.

Fsck displays an error message when it has problems mounting a local file system. Refer to the section about reporting **fsck** errors for information on dealing with error messages.

Fsck records its activities in the file **/etc/log/fsck.log**, which you might check following the completion of **fsck**. When file systems are corrupt, the **fsck** log contains a message for each error found in the file system. Refer to *Managing the DG/UX™ System* for information on the **fsck.log** file.

The **fsck** utility can verify the structure of only a DG/UX file system. If you use a database product or some other software that does not use a DG/UX file system, if possible, use a third-party package that can check the files.

Should **fsck** fail to fix a local file system, check the file system's **lost+found** directory for possible fragments of data that need to be recovered.

Checking for file fragments in lost+found

One of the functions of **fsck** is to locate blocks of data that have become disconnected from their files. If the **fsck** utility cannot reconnect the data to its file, it puts the data in a file in a directory called **lost+found** in the file system's mount point directory.

For example, if you have a file system mounted at **/sales/accounts**, **fsck** puts any disconnected blocks in files in **/sales/accounts/lost+found**. The files in this directory have names reflecting where **fsck** found them.

It is good practice to check the **lost+found** directory of a file system after **fsck** has checked it. You could use a simple script like the following to list files in all mounted local **lost+found** directories:

```
#!/bin/sh
/etc/mount | /bin/grep ' type dg/ux ' | /bin/cut -d" " - |\
while read DIR
do
  if test -f $DIR/lost+found/*
  then
    echo Found lost file fragments in $DIR/lost+found:
    /bin/ls -l $DIR/lost+found/*
  fi
done
```

You may also use the **file** command to determine the nature of a file by classifying it as English text, data, binary executable code, and so on.

You might check the **lost+found** directory following a power outage that has caused a system crash. When you reboot the system, **fsck** begins checking file systems. After **fsck** has finished, you look in **fsck.log** (or **fsck_fast.log** for file systems mounted for fast **fsck** checking) and see that file system **/sales/accounts** required checking by **fsck**. When the file system is mounted and accessible, you can check **lost+found**.

```
# cd /sales/accounts/lost+found ↵
# ls -l ↵
total 8
-rw-rw-rw-  1 curly  sales    3273 Sep 23 1991 #177431
# file #177431 ↵
#177431:  English text
```

You see that **fsck** found a piece of a file belonging to user **curly**. The **file** command classifies the contents as English text. Now you can tell user **curly** that one of his files was damaged, and, using the fragment from the **lost+found** directory as a clue, he can begin to research what file was damaged so that he can repair it or have it restored from backup.

Depending on how you configure your system, rebooting after a power outage or other failure may occur without operator intervention. Thus, file system checking could occur without your ever realizing it.

As a general practice, check the **lost+found** directories periodically to be sure there are no file fragments there.

Repairing corrupt DG/UX system files

If **fsck** cannot repair the / or **/usr** file system, your DG/UX system cannot boot to init 1. One common error that makes it impossible to boot a system is when the **/etc/inittab** file is damaged. When this happens, you frequently see a message like this when **init** starts running:

```
... SAD autopush configuration failed ...
```

If a system or disk failure damages **inittab** or other DG/UX system files (those in the **/usr** or / file systems), you need to repair the file systems and restore the damaged files from backups. If you cannot repair the file system, you need to reload the system software from

either the release media or a bootable copy of the file systems made with the **systemtape** utility.

Your options follow, from least to most drastic.

- Repairing a file system using the release CD
- Booting stand-alone **sysadm** from the system disk
- Booting stand-alone **sysadm** from the release tape
- Reloading the system software
- Booting a copy of the DG/UX system made with the **systemtape** utility
- Re-installing the DG/UX system

Repairing with the release CD

If you are running DG/UX 5.4R3.00 or later releases, you can use the CD-ROM release disk to repair your system. The CD can be booted into repair mode, under which a complete working environment is available from file systems on the CD. To boot the CD into repair mode, specify the **-R** option in the SCM boot command line:

```
SCM> b sd(cisc(),3) -R
```

After you have the repair mode environment running, repair the damaged / and **/usr** file systems' virtual disks with the **fsck** command and then mount them on different mount point directories. Refer to a later section in this chapter for details on running **fsck**. You can then copy files from the CD or from system backups to repair any damage.

Booting stand-alone sysadm from the system disk

To boot from **/usr**, use a command line like the following, specifying the physical disk on which the **/usr** file system resides:

```
SCM> b sd(cisc(),0)usr:/stand/sysadm ↵
```

Use the **sysadm** operation `File System -> Check a File System`. For information on invoking **fsck**, refer to a later section in this chapter.

If the **/usr** file system is corrupt or if the **/usr** file system is built on a virtual disk consisting of multiple partitions spanning multiple physical disks, an attempt to boot stand-alone **sysadm** will fail. If

stand-alone **sysadm** fails or has been deleted from **/usr**, then boot the release tape for the revision your system is running

Booting stand-alone sysadm from the release tape

If you cannot boot from the system disk, try booting from the release tape, using a command line like the following:

```
SCM> b st(cisc(),4)↵
```

If stand-alone **sysadm** boots successfully, check the / and **/usr** file systems with the **sysadm** operation `File System -> Check a File System`. For information on invoking **fsck**, refer to a later section in this chapter. When prompted for **fsck** options, specify **-xlp**.

If you still cannot boot the system, use `Check a File System`, specifying the **-y** option. The **-y** option repairs all non-fatal flaws in the file system, even if the repair results in lost files or data.

If **fsck -y** fails, attempt to repair file systems from the shell (go to the next section). If **fsck -y** succeeds, reboot the system. If it fails again, reinstall the DG/UX system using the instructions in a later section in this chapter.

Repairing file systems from the shell

Use these instructions to attempt repairing file systems from the shell.

1. Invoke the software installation menu using the stand-alone **sysadm** operation:

   `Install Software -> Prepare Virtual Disks`

2. When asked to modify the information presented in the table, answer **no**.

```
2.  Prepare required virtual disks

Run this step now?  [yes]↵
Required File System Mount Points:

     File System    Virtual  Current   Action     Blocks  Physical
     Mount Point    Disk     Blocks    Required   To Add  Disk
     ------------  --------  -------   --------   ------   --------
     -none-         swap      50000      None        -    sd(insc(0),0,0)
     /              root      40000      None        -    sd(insc(0),0,0)
     /usr           usr       240000     None        -    sd(insc(0),0,0)

Modify this information? [no] ↵
```

3. Escape to the shell by entering **!** at the **sysadm** prompt.

   ```
   Enter a number, a name, ? or <number>? for help, <NL>
   to redisplay menu or q to quit: [All Steps]: !
   ```

   The damaged / file system is mounted as **/mnt/root**; the damaged **/usr** file system is mounted as **/mnt/usr**.

4. To gain access to the normal commands, set your path as follows:

   ```
   # PATH=/mnt/root/sbin:/mnt/usr/bin:/mnt/root:sbin↵
   # export PATH ↵
   ```

   Note that some commands may not work if they use shared libraries, since the real **/usr** file system that contains them is mounted in a different place than normal.

5. Go to **/mnt/root** and **/mnt/usr** to examine and repair the damaged file systems.

   ```
   cd /mnt/root
   cd /mnt/usr
   ```

   If you are repairing a damaged **inittab** file, there may be an undamaged backup of it in **/etc/inittab.backup**. The prototype file is in **/etc/inittab.proto**.

6. When you finish fixing the file systems, return to **sysadm** by exiting the shell.

   ```
   # exit ↵
   ```

7. Exit **sysadm**.

   ```
   Enter a number, a name, ? or <number>? for help, <NL>
   to redisplay menu or q to quit: [All Steps]: q ↵
   ```

   Exiting **sysadm** shuts down the system. You can now reboot your repaired / and **/usr**.

8. Reboot the repaired DG/UX system.

   ```
   SCM> b ↵
   ```

   If you still cannot boot your system, you have three more options: reload the system software, reboot the system from a copy made with the **systemtape** utility or re-install the system completely. The next three sections cover these options.

Reloading the system software

If all else fails, you must boot the release tape and reload the damaged file systems.

1. Boot stand-alone **sysadm** from the release tape using a command like the following:

   ```
   SCM> b st(cisc(),4)↵
   ```

2. From the Standalone Sysadm Main Menu, select `Install Software`.

3. From the Install System Software Menu, select `Load Software`.

   After you load, you must also set up packages, rebuild and reboot the kernel. After you reboot, you may replace any files you wish by restoring them from backup. For information on restoring selected files, refer to *Managing the DG/UX™ System*.

Booting a copy of the DG/UX system made with the systemtape utility

You may use a bootable copy of your / and **/usr** file systems made with the **systemtape** utility. In addition to normally scheduled backups, you can use **systemtape** to create a bootable tape of these file systems at regular intervals. You can then use that tape if you need to restore your system. See the **systemtape**(1M) manual page for more information.

Reinstalling the DG/UX system

If all of these options fail, you must re-install the DG/UX system completely.

1. From the Standalone Sysadm Main Menu, select `Virtual Disk`.

2. From the Virtual Disk menu, select `Remove`.

   For procedures to remove the **root** and **usr** virtual disks, see Chapter 10.

3. After you have removed the **root** and **usr** virtual disks, you must reinstall your DG/UX system — prepare the **root** and **usr** virtual disks, load and set up all packages contained on those virtual disks, and build and reboot the kernel.

   From the Standalone Sysadm Main Menu, select `Install Software`.

4. From the Install System Software Menu, select `All steps`.

   Refer to *Installing the DG/UX™ System* for details.

   After you have re-installed the DG/UX system, you may replace any files you wish by restoring them from backup. For information on restoring selected files, refer to *Managing the DG/UX™ System*.

Tuning fsck

Before you use **fsck**, you may tune it according to your system's requirements. You may tune these aspects of **fsck**:

- **Fsck** kernel parameters
- Fast recovery **fsck**
- **Fsck** logging

Fsck kernel parameters

You may tune three kernel parameters to control the behavior of **fsck**.

`RUNFSCK` Specifies whether or not the kernel should run **fsck** to check the / file system before trying to mount it at system boot. The value 1, the default, enables the RUNFSCK variable; 0 does not.

`FSCKFLAGS` This parameter is set only if RUNFSCK is set. By default, both variables are set. The default options selected for use with **fsck** are **xlq**. If the file system is marked unmountable, the **x** option checks it. The **l** option performs a fast recovery **fsck**, and the **q** option repairs inconsistencies automatically, bypassing your approval. See the section on invoking **fsck** from the shell for details.

`ROOTLOGSIZE` This parameter specifies the number of blocks to be used for **fsck**'s fast recovery log on the / file system. A value of 0, by default, disables fast recovery logging. For example, a value of 32 sets the **fsck** log size for the / file system to 32 blocks.

Refer to Chapter 2 for information on kernel building.

Invoking fast recovery fsck

Decide which form of **fsck** to use. The two flavors follow:

- Multi-pass **fsck**
  Checks normal file systems by performing five passes through the file system, each pass verifying different aspects of the file system, in a particular order.

- Fast recovery **fsck**
  Performs single-pass verification of file systems that were mounted to take advantage of the **fsck** logging feature. This method does not follow a particular order when file checking.

  You can perform fast recovery **fsck** only if you enabled **fsck** logging when you created the mount point for the file system (see the next section), and this is the first time that **fsck** has checked the file system since that mount.

  A fast recovery file system check is preferable primarily for file systems where rapid recovery and high availability are crucial. The fast option reduces downtime by using file system logging. Logging keeps records about file system updates, which permits a quick reconstruction of file systems from the log's records. In the event of a file system failure, the file system checker can use the log to speed the process of verifying and restoring file system integrity. The downside of logging is that it has some negative impact on run time write performance in the file system.

  To enable fast recovery **fsck**, you must select the **l** option when invoking **fsck**. Information on invoking **fsck** and supplying options is supplied in a later section.

fsck logging

If you select fast recovery **fsck**, you must also enable logging. You enable logging when you create a mount point for the file system with the **sysadm** operation:

```
File System -> Local Filesys -> Add
```

If you opted against logging when mounting the file system, you may modify your request through this **sysadm** operation:

```
File System -> Local Filesys -> Modify
```

The file system checking and logging attributes follow:

`Pass Number` The **fsck** pass number indicates the order in which the DG/UX system checks for corrupted file systems when the system boots. When you create a mount point for a file system, you may assign a pass number to the file system in the range of 0–9 or accept the default, 1. Values range from 0 to 9; 0 means never check the file system is never checked; 1 means check it first; and 9 means check it last. Multiple file systems can be checked in one pass. In general, you can accept the default pass number.

`Logging` **Fsck** logging enables or disables file system logging. The log collects file system modifications, which contributes to a quick recovery of the file system when the system crashes. Logging, however, slows runtime write performance. Select logging when rapid recovery and high availability are crucial.

`Log Size` You specify the log size in 512-byte blocks. There is a tradeoff in performance between log files of different sizes. A large log file improves run time performance but prolongs recovery time. A small log file reduces recovery time but degrades run time performance. Recovery time depends on the size of the intent log, not the size of the file system.

IMPORTANT: As an alternative, you may wish to hand-edit the **/etc/fstab** file, specifying the **fsck** log file size. An example follows:

```
/dev/dsk/sales /sales dg/ux rw,fsck_log_size=64 d 1
```

Refer to Chapter 8 for details on defining the attributes of file system checking and logging when creating or modifying a mount point for a DG/UX file system (adding a file system to the **/etc/fstab** file).

Invoking fsck

Fsck can be invoked in any of these ways:

sysadm Both stand-among and stand-alone versions. Select the stand-among **sysadm** operation `File System -> Local Filesys -> Check` or the stand-alone **sysadm** operation `File System -> Check a File System`, as appropriate.

command line From the command line, type:

fsck [*options*] [*file_system_names*]

where *options* are single character flags that modify the behavior of the command and *file_system_names* refer to the file systems that you want to check. Refer to the **fsck**(1M) manual page for details on options. If you do not specify which file systems to check, **fsck** will check those having appropriate entries in the **/etc/fstab** file. An appropriate entry is one having a nonzero pass number and a "rw" or "ro" mounting status. The **fstab** file collects entries when you create a mount point for a local file system. See Chapter 8 for information on the **fstab** file and creating mount points.

initialization The system includes the initialization version of **fsck** in the operating system kernel and runs it automatically over the (/) root file system when you boot your system.

rc script You can set up any of your system's **rc** scripts to invoke **fsck** when you change run levels. For more information on **rc** scripts and how to set them up, see *Managing the DG/UX™ System*.

Invoking fsck through sysadm

1. You must unmount the file system before you check it. If it is not already unmounted, do so using the following **sysadm** operation:

   ```
   File System -> Local Filesys -> Unmount
   ```

   For details on unmounting a local file system, see Chapter 11.

   If you cannot unmount the file system, you may need to take the system down to single user mode, which attempts to unmount all file systems except the / and **/usr** file systems. Verify that all file systems are unmounted using the **mount** shell command. If there are any mounted file systems, unmount them with the **umount** shell command. Then use the **fsck** shell command. Refer to *Managing the DG/UX™ System* for details on taking the system down.

2. To invoke **fsck**, select the **sysadm** operation:

   ```
   File System -> Local Filesys -> Check
   ```

3. Supply the name of the file system to check. An example is provided as follows: (To recall file system mount point names, select the **sysadm** operation `File System -> Local Filesys -> List`.)

   ```
   File System(s) to Check: /usr/opt/fred ↵
   ```

4. Customize **fsck** to perform the checks you desire. In particular, selecting the **l** option invokes the fast recovery **fsck**. If you choose no options, **p** is selected by default. The **p** option detects all possible inconsistencies, but corrects only those that may result from an abnormal system halt. For a complete list of options, refer to the section on invoking **fsck** from the shell.

   Fsck performs a multi-pass check under two conditions:

   - Fast recovery **fsck** cannot repair the file system or
   - You specify the **-l** option for a normal-recovery file system.

```
Fsck Options: ↵
```

Fsck then treats the problem as necessary.

Invoking fsck from the shell

You can invoke **fsck** from the shell using the following command format:

fsck [*options*] [*arguments*]

By default, **fsck** checks all file systems listed in the **/etc/fstab** file.

All options are represented by single-character flags; options must begin with a hyphen. All options except for **–t** are Boolean flags, and may thus be combined. For example, you can combine the options **–p**, **–x**, and **–D** as follows: **fsck –pxD**.

The following options are interpreted by **fsck**:

–l Perform fast recovery using the **fsck** log, if possible. The **fsck** program can perform fast recovery only if:

- You specified the **fsck_log_size** option, which turns on fast recovery logging, the last time you mounted the file system (unless it was the / (root) file system, for which fast recovery logging is in effect if you set the ROOTLOGSIZE parameter when you built the kernel), and
- This is the first time that **fsck** has checked the file system since that mount.

Fsck performs a multi-pass check under two conditions:

- Fast recovery **fsck** cannot repair the file system or
- You specify the **–l** option for a normal-recovery file system.

–p Detect all possible inconsistencies, but correct only those inconsistencies that may be expected to occur from an abnormal system halt. For each corrected inconsistency, one or more lines will be printed identifying the file system and the nature of the correction. Any other inconsistencies will cause the check of that file system to fail. **Fsck** corrects the following 15 inconsistencies (and only those listed) for the specified file systems:

1. An inode has an incorrect count of the blocks it uses. The count is corrected.

2. An inode is partially truncated. Partial truncation can occur if the system is abnormally halted while a file is being truncated, leaving the file claiming more data blocks than its size in bytes would require. The extra blocks are freed.
3. A directory has an incorrect child count. The count is corrected.
4. A directory entry exists for an inode which is unallocated. The directory entry is removed.
5. A directory entry's file name length is incorrect. The length is corrected.
6. An inode is unreferenced (has no directory entries anywhere in the file system). The inode is reconnected in the **lost+found** directory.
7. No **lost+found** directory exists, but an inode needs to be reconnected there. The **lost+found** directory is created.
8. The root directory needs to be expanded in order to make room for a **lost+found** directory entry. The directory is expanded.
9. The **lost+found** directory needs to be expanded in order to make room for a directory entry for an inode being reconnected there. The directory is expanded.
10. An inode's link count is incorrect. The count is corrected.
11. The root control point directory's resource accounting (blocks, inodes) is incorrect. The counts are corrected.
12. A disk allocation region (DAR) has an incorrect free-block bitmap. The bitmap is corrected.
13. A DAR has an incorrect free-inode list. The list is corrected.
14. A DAR has incorrect summary counts of used blocks, inodes or directories. The counts are corrected.
15. The summary counts in the superblock are incorrect. The counts are corrected.

–q Repair the inconsistencies listed under the **–p** option automatically, without asking for user approval. Unlike **–p** however, more serious inconsistencies will not cause **fsck** to fail; the user must still answer the resulting queries.

–y Audit and repair all file system inconsistencies assuming a "yes" response to all questions asked by **fsck**.

CAUTION: Use this option with great care, since it could lead to irreversible changes to the file system.

–n	Audit all file system inconsistencies, assuming a "no" response to all questions asked by **fsck**. This option also means that all file systems will be opened with read-only intent.
–x	Look at a file system's superblock to see if it is marked mountable. If so, do not check the file system for inconsistencies. If the file system is marked unmountable, check it.
–s	Ignore the actual free-block bitmap and unconditionally reconstruct a new one.
–S	Conditionally reconstruct the free-block bitmap. A free-block bitmap is reconstructed if and only if the file system is consistent. This option also forces a "no" response to all **fsck** questions.
–D	Directories are checked for bad blocks.
–f	Fast check: blocks and sizes are checked; the free block bitmap is reconstructed if necessary.

The following options are mutually exclusive, and use of more than one per invocation is not allowed: **–l**, **–y**, **–n**, **–p**, **–q**, and **–S**.

Arguments to fsck options

If you do not specify which file systems to check on the **fsck** command line, **fsck** will check those having appropriate entries in the **fstab** file. An appropriate entry is one having a nonzero pass number and a "rw" or "ro" mounting status. If you specified the **–p** option, the checking occurs in order of pass number. **Fsck** checks simultaneously file systems that have the same pass numbers. Otherwise, checking occurs in order of appearance in **fstab**.

If you specify arguments (the rest of the command line after the option flags), fsck checks those file systems, and only those file systems, in sequential order.

You may specify file systems as arguments to **fsck** in one of two ways: by the special device file (in **/dev/dsk** or **/dev/rdsk**) containing the file system; or by the directory that **/etc/fstab** indicates will serve as the mount point for the file system.

Behavior of fsck

If **fsck** finds no errors, file system checking proceeds without any input from the user. If it finds a fatal inconsistency, it does no further checking on that file system; the **fsck** program either exits

or proceeds to the next specified file system. If the **-p** option discovers an inconsistency and that error is one of those listed under **-p**, or if the **-y** option discovers the inconsistency, **fsck** fixes the inconsistency without your intervention. Any other discoveries of inconsistencies require that you make a decision. The **fsck** program prompts with its recommended action. If you answer **yes**, **fsck** takes the recommended action. In the case of **-p**, **fsck** takes no damaging action without approval.

IMPORTANT: You may invoke **fsck** with the **-y** and **-n** options, respectively, to grant advance approval or disapproval.

Typical output follows for a problem that **fsck** can fix without your intervention:

```
/dev/dsk/usr_opt_fred:  File System is now mountable.

/dev/dsk/usr_opt_fred:  20 of 330 blocks used (310
free); 1 of 62 inodes used (61 free).

/dev/dsk/usr_opt_fred:  Time to fix
/dev/dsk/usr_opt_fred was 0 seconds.
```

Typical output follows for a problem that does require your intervention:

```
# fsck /dev/dsk/usr_opt_fred ↵

** /dev/dsk/usr_opt_fred:
** Phase 1 - Check Blocks and File Sizes
** Phase 2 - Check Directory Contents
** Phase 3 - Check Connectivity
** Phase 4 - Check Link Counts and Resource Accounting

Inode 67 (owner: 2 [bin]; group: 2 [bin]; size: 52736
bytes;
type: Ordinary; mode: 755; mtime: Fri Nov 20 17:54:36
1987) has incorrect link count (2 should be 1) -- fix?
```

If you respond **yes**, the **fsck** program corrects the incorrect link count that it found for inode 67 and moves on to Phase 5. After it finishes, it reports that the file system is mountable.

```
** Phase 5 - Check Disk Allocation Region Information
File system is now mountable.

13936 of 50000 blocks used (36064 free); 288 of 5822
inodes used (5534 free).

#
```

If you respond **no**, **fsck** does not fix the inconsistency, and the file system remains unmountable.

The **fsck** program will refuse to check any file system for which any of the following conditions hold true:

- The file system is mounted (except when you specified the **-n** option, which opens the file system read-only).
- The special file associated with the file system cannot be opened.
- The specified pathname (or its device node associate in **/etc/fstab**) is not a block-special, character-special, or regular file whose size can be determined.

For complete descriptions of **fsck** error messages and recovery actions, see a later section on reporting **fsck** errors.

What makes a file system internally consistent?

Every time a file is modified, the DG/UX operating system performs a series of file system updates. When written to disk, these updates yield a consistent file system.

There are four types of file system updates. The updates involve the following areas:

- Superblock and disk allocation region (DAR) information, which includes the free-block bitmap and the inode table
- Inodes
- Index (indirect) blocks
- Data blocks (directories and other files)

The next sections discuss each area in detail.

Superblock and disk allocation region information

The superblock and disk allocation region (DAR) information are some of the most commonly corrupted items. Every change to the file system's blocks or inodes modifies the superblock and the disk allocation region information. The superblock and disk allocation region are most often corrupted when the system was not properly shut down with the **shutdown** command.

Superblock and disk allocation region inconsistencies can involve file system size, the number of available inodes and blocks, the free-block bitmaps and the free inode lists. A brief discussion of some of these types follows:

Free-block bitmap

Each DAR contains a bitmap representing all the blocks in the DAR. Multi-pass **fsck** compares that information with its own map of allocated blocks, looking for inconsistencies that suggest DAR corruption.

Free-inode list

Each DAR contains a linked list of free inodes in that DAR. The **fsck** program ensures that all free inodes in the DAR appear in the list, and that no allocated inodes appear in the list.

Summary counts

The superblock and each DAR contain several counts: the number of used inodes, the number of used blocks, the number of directories. The **fsck** program compares these counts to the information it has compiled.

Inodes

An individual inode is less likely than the superblock to be corrupted. However, because of the great number of active inodes, the free inode lists are as susceptible as the superblock to corruption. Multi-pass **fsck** checks for inconsistencies involving format and type, link count, duplicate blocks, and inode size. Fast recovery **fsck** checks for inconsistencies involving link count, duplicate data element pointers, and inode size.

Format and type

Each inode contains mode information. This information describes the type of the inode. Inodes may be one of eight types: regular, directory, control point directory, special block, symbolic link, special character, FIFO, or socket. Any other type is illegal.

Link count

Each inode contains a count of the directory entries linked to the inode. Multi-pass **fsck** verifies the inode count by checking down the total directory structure, starting from the root directory, and calculating an actual link count for each inode. Fast recovery **fsck** verifies link counts by comparing them to link count records in the log.

In multi-pass checking, when the link count (which is stored on the disk) is nonzero and the actual link count (kept by **fsck**) is zero, no

directory entry appears for the inode. If no entry appears, **fsck** may link the disconnected file to the **lost+found** directory. Fast recovery has records that enable it to match lost files with their original directories; consequently, it does not leave files or file fragments in **lost+found**.

If the stored and actual link counts are nonzero and unequal, **fsck** may replace the link count on the disk by the actual link count. When this situation arises, a directory entry may have been added or removed without the inode being updated.

Duplicate blocks

Each inode contains a list and sometimes pointers to lists (index blocks) of all the blocks claimed by the inode.

Multi-pass **fsck** checks these lists for duplicate blocks. Duplicate blocks can occur when a file system uses blocks claimed by both the free-block bitmap and other parts of the system or when two or more inodes claim the same block. Any block claimed more than once is flagged by **fsck** as a duplicate block. If there are any duplicate blocks, **fsck** makes a partial second pass of the inode list to find the inode of the duplicated block. If the files associated with these inodes are not examined for correct content, **fsck** will not have enough information to decide which inode is corrupted and should be cleared. Usually, the inode with the earliest modification time is incorrect and should be cleared.

Fast recovery **fsck** does not check for duplicate blocks. It compares inodes and index blocks with log records listing which blocks were allocated for each file.

Size checks

Each inode contains a size field. This field's size indicates the number of bytes in the file associated with the inode.

The **fsck** program can check the size for inconsistencies, such as directory sizes that are not a multiple of 512 bytes, or a mismatch between the number of blocks actually used and the number indicated by the inode size. The **fsck** program also checks for directory corruption, where conflicting information is found within the directory entries.

The **fsck** program can also perform a check of the size field of an inode. Multi-pass **fsck** uses the size field to compute the number of blocks that should be associated with the inode, and then compares that number to the actual number of blocks claimed by the inode. Fast recovery uses log records to track changes to the file size and number of blocks claimed by the inode.

Control point directories

The root inode of a file system is a special type of directory known as a *control point directory.* A control point directory is like an ordinary directory except that it has resource limits associated with it for inodes and for data blocks. Effectively, these limits determine how many files you may create in the file system and how large the file system (total size of all files and directories) may become. The total resources consumed by the control point directory and all its descendants (to which it is the *space parent*) may not exceed the size limits. The limit on data blocks (size) is determined by the size of the virtual disk on which the file system resides. To change the size of the file system, use the Expand and Shrink operations in the **sysadm** `File System -> Local Filesys -> Expand` or `Shrink` operations, as necessary.

Index blocks

Index blocks (also known as indirect blocks) are owned by an inode. Therefore, inconsistencies in an index block directly affect the inode that owns the block.

Multi-pass **fsck** can check inconsistencies involving blocks already claimed by another inode and block numbers outside the range of the file system.

Fast recovery **fsck** uses the log to track changes to the contents of index blocks.

Data blocks

There are two types of data blocks: plain data blocks and directory data blocks. Plain data blocks contain the information stored in a file. Directory data blocks contain directory entries. The **fsck** program does not try to check a plain data block.

There are several checks that **fsck** makes for directory data blocks. Multi-pass **fsck** checks all entries in all directory data blocks, but fast recovery **fsck** checks only those entries that have been modified. The directory data block check searches for:

- bad self-identification information
- directory entries for unallocated inodes
- directory entries for inodes which do not exist in the file system
- directories that are disconnected from the file system

If a directory entry inode number points to an unallocated inode, **fsck** may remove that directory entry. Directory entry inode

numbers can point to unallocated inodes when the data blocks containing the directory entries are modified and written out, but the inode is not written out.

If a directory entry inode number is pointing to a nonexistent inode, **fsck** may remove that directory entry. This condition occurs if bad data is written into a directory data block.

The **fsck** program checks that all directories are linked into the file system; i.e., they have a parent directory pointing to them (except for the root). Multi-pass **fsck** links unlinked directories to the file system's **lost+found** directory. Fast recovery **fsck** links unlinked directories back to their original places in the directory tree. When inodes are being written to the file system without the corresponding directory data blocks being written to the file system, the directories are not linked into the file system.

Both modes of **fsck** report inconsistencies that they find. It is your option to fix the inconsistencies or ignore them.

Reporting fsck errors

When **fsck** detects an inconsistency, it reports the error condition on the console screen. If a response is required, **fsck** prints a prompt message and waits for a response. This section explains the meaning of each error condition, possible responses, and related error conditions.

In "Error messages for phased checking," the error conditions are organized by the phase of the **fsck** program in which they can occur. The error conditions that may occur in more than one phase are discussed under "General error messages."

Error messages that occur only during fast recovery file system checking (**fsck** logging is enabled) appear in "Error Messages Exclusive to Fast Recovery Checking."

The following error messages are presented in their basic form. Fatal errors (such as during **fsck -p**) cause the error message to be prefaced by the string `Fatal Error`. Running with **-p** also causes messages to be preceded by the name of the file system to which the message applies. The following abbreviations appear in the description of error messages:

B	A (decimal) disk block number.
N	A decimal number.
O	An octal number.

C	A character.
D, F	A directory name, file name or pathname string.
I	An inode description string. At the very least, this will consist of the inode number. If possible, the inode's size, file type, file mode, UID, GID, time of last modification, owner name, group name and pathname will also be present.

Error messages for phased checking

The **fsck** program checks file systems in phases only if **fsck** logging was enabled. This section lists errors you may see during the typical, phased **fsck** check. Some of these error messages may also appear during file system checking for which **fsck** logging was enabled.

The section "Error messages exclusive to fast recovery checking" lists errors that can appear during file system checking for which **fsck** logging was enabled.

General error messages

The messages described in this section may appear at any time during an **fsck** session.

Cannot allocate memory for internal tables (N bytes requested)

The **fsck** program cannot allocate enough memory; this can occur only when invoked through stand-alone **sysadm**. The **fsck** program will abort. Bring up your system and use the **fsck** shell command instead.

Cannot read block B

A disk read of block number *B* has failed. The **fsck** program treats the block it could not read as if it were filled with all zeroes, and continues execution, but the file system is not marked as mountable upon conclusion of checking. Use **sysadm** (`Device -> Disk -> Physical -> Bad Blocks -> Map`) to remap the bad block *B* and run **fsck** again.

Cannot write block B

A disk write of block number B has failed. The **fsck** program continues execution, but the file system being checked is not marked as mountable upon conclusion of checking. Use **sysadm**

(`Device -> Disk -> Physical -> Bad Blocks -> Map`) to remap the bad block *B* and run **fsck** again.

Fork failed

The **fsck** program has failed in an attempt to spawn a child process. This will occur only when running **fsck** with the **-p** option. The only file system affected will be the one for which the child **fsck** process was being created; no check will occur.

Internal Software Error: Cannot seek to block B — aborting

A disk seek to block number *B* has failed; this should never happen. Contact your Data General support representative if this message is displayed. The **fsck** program terminates.

Invalid response; please answer yes or no

An invalid answer has been entered in response to one of **fsck**'s questions. The **fsck** program will not continue until a valid response has been entered. The following strings are valid responses: **y, Y, yes, YES, n, N, no** and **NO**.

Errors during fsck invocation

Before starting to check a file system, **fsck** must parse its command line and determine which files to check. The following messages result from command line errors or information in the file **/etc/fstab**.

The directory D is the mount point for F

The **fsck** program has been given a directory *D* to check and has determined that *D* is the mount point for the file system *F*. This message is purely advisory.

The flags -y, -n, -p, -q and -S are all mutually exclusive

More than one of the above flags has been specified on the **fsck** command line. Only one is allowed. The **fsck** program will abort.

Unknown option: -C

An unknown option flag, *C*, has been specified on the **fsck** command line. Valid flags are as follows: **-l, -y, -n, -p, -q, -t, -D, -f, -s, -S**, and **-x**. When you give it an invalid option, **fsck** will abort.

Errors during fsck initialization

Before a file system check can be performed, **fsck** must set up certain tables and open certain files. The following messages can result from errors during this phase.

Block B is invalid Inode Table Block — rewrite as empty block?

The inode table block *B* does not contain the proper self-identification information. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `rewrite as empty block?` prompt are:

YES	Fix this error condition by rewriting this block as an empty file node table block. Any inodes that formerly occupied slots in this block will be cleared.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Cannot determine disk size of F

The **fsck** program has been given a file system *F* to check, but the size of *F* cannot be determined. The **fsck** program will abort checking this file system. This should never happen. Contact your Data General support representative if this message is displayed.

Cannot find a readable copy of the superblock

Neither of the two copies of the superblock can be read. The **fsck** program will abort checking this file system.

Cannot find a valid copy of the superblock

Neither of the two copies of the superblock contain the required self-identification information. The **fsck** program will abort checking this file system.

Cannot open F for reading

The **fsck** program has been given a file system *F* to check, but *F* cannot be opened for reading. Check the mode of *F*. The **fsck** program will abort checking this file system.

Cannot open F for writing

The **fsck** program has been given a file system *F* to check, but *F* cannot be opened for writing. Check the mode of *F* and make sure

that no disks containing the file system are physically write-disabled. The **fsck** program will abort checking this file system.

Cannot read superblock copy N

One of the two superblock copies cannot be read. The **fsck** program will attempt to use the other copy and continue.

F is not a regular file, block-special file, character-special file or valid mount point

The **fsck** program has been given a file system *F* to check, but *F* is not of the correct type. *F* must be an ordinary file type, block-special or character-special, or else it must be listed in the file **/etc/fstab** as a valid mount point directory. The **fsck** program will abort checking this file system.

File system is too large to check

Stand-alone **fsck** cannot allocate enough memory for its internal tables to begin checking the file system. The **fsck** program will abort checking this file system. Bring up your system and use the **fsck** shell command instead.

File system size stored in superblock is incorrect (N1 blocks should be N2) — fix?

The superblocks contain an incorrect file system size figure. If run with the **-p** or **-q** options, **fsck** will automatically correct this. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the file system size to N2, the actual size of the disk containing the file system.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Invalid default Data Element Size exponent: N — fix?

The default data element size for files (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the value to the default of 4 (meaning an element size of 16 blocks).

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the default data element value to 4.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Directory Data Element Size exponent: N — fix?

The default data element size for directories (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the value to the default of 4 (meaning an element size of 16 blocks).

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the default directory value to 4.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Directory Index Element Size exponent: N — fix?

The default index element size for directories (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the value to the default of 0 (meaning an element size of 1 block).

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the default directory index value to 0.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Index Element Size exponent: N — fix?

The default index element size for files (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the value to the default of 0 (meaning an element size of 1 block).

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the default index element value to 0.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid Disk Allocation Region size: N blocks

The DAR size stored in the superblocks is invalid. The **fsck** program will abort checking this file system.

Invalid first allocation threshold file size: N — fix?

The superblocks contain an invalid first allocation threshold file size (the number of blocks a file can allocate in its initial DAR before all subsequent allocations are made from a different DAR). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the first allocation threshold file size to the default limit for DARs of the size specified in the superblock.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid number of inodes per Disk Allocation Region

The number of inodes per DAR stored in the superblocks is invalid. The **fsck** program will abort checking this file system.

Invalid second allocation threshold file size: N — fix?

The superblocks contain an invalid second allocation threshold file size (the number of blocks a file can allocate in a noninitial DAR before all subsequent allocations are made from a different DAR). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the second allocation threshold file size to the default limit for DARs of the size specified in the superblock.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

No check necessary for F

The file system F is already marked mountable and **fsck** was invoked with the **-x** flag. The **fsck** program will not check this file system.

Superblock copies differ; using newer copy

Both copies of the superblock are readable and both contain the required self-identification information, but they differ. The **fsck** program will use the first copy (which is guaranteed to be more recent) and continue.

Superblock copy N is invalid

One of the two superblock copies does not contain the required self-identification information. The **fsck** program will attempt to use the other copy and continue.

Superblock has invalid contents in reserved area — fix?

A copy of the superblock has nonzero contents in a reserved area. If running with the **-p** flag, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES	The superblock's reserved area is initialized so that it contains all 0s.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors during phase 1 - check blocks and file sizes

This phase of **fsck** operation is concerned with inodes. The following messages result from errors in inode types, inode format, file sizes and the data element pointers and index element pointers that make up a file's structure.

Incorrect block count in Inode I (N1 should be N2) — fix?

The inode *I*'s count of the blocks it uses is incorrect. If run with the **-p** option, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting inode *I*'s block count to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I claims an invalid block (B) — clear bad pointer?

The inode I claims block B, which does not exist. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES	Fix this error condition by clearing the pointer in inode I that claims the nonexistent block. This may result in a "hole" in the file if the cleared pointer preceded the last block of the file.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I claims a system block (B) — clear bad pointer?

The inode I claims block B, which is a system block (a bitmap block, file node table block, DAR entry table block or superblock). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the system block.

Possible responses to the `clear bad pointer?` prompt are:

YES	Fix this error condition by clearing the pointer in inode I that claims the system block. This may result in a "hole" in the file if the cleared pointer preceded the last block of the file.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has an Index Block (B) with invalid format — clear bad pointer?

The inode I claims block B as an index block, but block B does not contain the proper self-identification information. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES	Fix this error condition by clearing the pointer in inode I that claims the index block. This may result in a "hole" in the file if the cleared pointer preceded the last block of the file.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has invalid contents in its reserved area — fix?

The inode I does not contain the proper self-identification information. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by initializing inode *I*'s reserved area.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has invalid fragment size exponent (N) — clear?

The inode I has an illegal exponent representing the size of the file's fragment. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Inode I is of unknown file type (O) — clear?

The inode I is of type *O*, which is an unrecognized octal number. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I.
NO	Ignore this error condition. The **fsck** program will abort checking this file system.

Inode I is partially truncated — fix?

The inode *I*'s size is shorter than the number of blocks allocated to it. If run with the **-p** option, **fsck** will complete automatically the truncation. Otherwise, **fsck** will ask to complete truncating inode I.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by completing the truncation of inode I down to the size stored in the inode.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors during phase 1b – resolve duplicate claims

When **fsck** finds a block claimed by two or more files, it rescans the file system to find the original claimant of that block. This section lists the error messages that result from settling the claim to the disputed block.

Inode I claims another file's blocks — clear?

The inode I claims some blocks that belong to another file. **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I.
NO	Ignore this error condition. This will result in the same question being asked about the next claimant of the disputed block. As long as enough files are eventually cleared to resolve the duplicate claims on block B, **fsck** will continue normally. However, if at the end of Phase 1b any duplicate claims still exist, **fsck** will not mark this file system as mountable upon completing the check.

Errors during phase 2 – check directory contents

This phase is concerned with the contents of directories. The messages in this section result from improperly formatted directory blocks, an improperly formatted root (/) directory, and bad directory entries. During this phase, all bad entries and inodes are removed from the file system tree.

Directory inode I has a hole — fix?

The directory inode *I* has at least one "hole" in its file structure (gaps before the end of file). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rearrange the directory blocks to fill in the hole.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rearranging the blocks in the directory to eliminate the hole.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I has incorrect child count (N1 should be N2) — fix?

The directory inode *I*'s count of children, *N1*, is incorrect. If run with the **-p** or **-q** options, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct the child count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting inode *I*'s child count to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I has an invalid block (B) — rewrite as empty block?

The directory inode *I* has a block (address *B*) which does not contain the proper self-identification information. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `rewrite as empty block?` prompt are:

YES	Fix this error condition by rewriting block *B* as an empty directory block. Any directory entries that formerly occupied this block will be destroyed.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 of invalid size — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry is too long, too short, or is not a multiple of 4 bytes in size. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 which is out of order — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2* which has a bad sequence number, meaning that the entry is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with filename of invalid size — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's file name is too long or too short. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with an illegal filename: F — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name *F* is **.** (dot) or **..** (dot-dot). These two names are reserved for the directory's links to itself and to its parent, respectively. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry? prompt are:`

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has a filename with an illegal character, octal value O — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name includes the illegal character *O*. Neither a non-ASCII nor the slash character is allowed. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has an illegally long pathname — remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the pathname for that entry relative to the root of the file system would exceed **MAXPATHLEN** (1024) bytes. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there

will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO — Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has invalid contents in its reserved area — fix?

The directory inode *I1* has a directory entry for inode *I2*, which has nonzero information in its reserved area. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the contents of the reserved area of inode *I2*.

Possible responses to the `fix?` prompt are:

YES — Fix this error condition by initializing the reserved area of inode *I2*.

NO — Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode number I2, which is invalid — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is not a valid inode number. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES — Fix this error condition by removing the directory entry for inode *I2*.

NO — Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode number I2, which is unallocated — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is not an allocated inode. If run with the **-p** option, **fsck** automatically will remove the directory entry for inode *I2*. Otherwise, **fsck** will ask to remove the directory entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry which is an extraneous link to directory inode I2 — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is a directory that does not list *I1* as its parent directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry which is an extraneous link to symbolic link inode I2 — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is a symbolic link which already has another hard link. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has an entry (for inode I2) which crosses a control point directory boundary — remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I1* and *I2* have different space parent control point directories.

If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Incorrect filename length in directory inode I1 for directory inode I2 (N1 should be N2) — fix?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name length, *N1*, is incorrect. If run with the **-p** option, **fsck** will automatically correct the directory entry's name length to *N2*. Otherwise, **fsck** will ask to correct the name length.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the directory entry's length to *N2* bytes.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode is of wrong file type — fix?

The root inode (inode 2) is not a control point directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the incorrect file type.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the file type of inode 2 to type control point directory.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode is not allocated — fix?

The root inode (inode 2) is not allocated. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to allocate inode 2.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by allocating inode 2 as the root.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's parent directory is not the root — fix?

The root inode's parent directory is not the root (itself). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to list the root inode as its own parent.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the root inode's parent directory to itself.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's space parent control point directory is not the root — fix?

The root inode's space parent control point directory is not the root (itself). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to list the root inode as its own space parent.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the root inode's space parent control point directory to itself.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's space usage limit is too large (N1 should be N2) — fix?

The root inode's space usage limit, *N1*, is bigger than the size of the file system, *N2*. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset the limit to *N2* blocks.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by setting the root inode's space usage limit to *N2* blocks.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors during phase 3 – check connectivity

Phase 3 of **fsck** deals with the reconnection of unreferenced files and directories onto the file system tree. The messages in this section result from attempts to connect unreferenced files into the **lost+found** directory. Note also that any of the Phase 2 messages may be seen in this phase, as the contents of any reconnected directories must be checked.

Cannot find enough contiguous free blocks to expand directory inode I

The **fsck** program could not find enough contiguous free blocks to expand the directory inode I. Some unreferenced files may not reconnected as a result of this failure; they can be reconnected during a later **fsck** session after enough space has been freed in the file system.

Control point directory inode I has an entry named 'lost+found' which is not a directory — remove bad directory entry?

The control point directory inode I already has an entry named **/lost+found**, but which is not of type directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the bad entry from inode I. The bad entry's inode will itself be reattached in the new **/lost+found** directory which will be created in directory I1.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Could not reconnect inode I

The **fsck** program was unable to reconnect the unreferenced inode I because it could not allocate enough blocks to expand the **/lost+found** directory, or because it could not allocate a free inode to use as the **/lost+found** directory.

Directory inode I is already as large as it can become

The **fsck** program has discovered that a directory it was attempting to expand is already the maximum size a directory can become.

Inode I1 lists as its space parent inode number I2, which is not a valid control point directory — reset space parent to root?

The inode I1 has the noncontrol point directory inode I2 listed as its space parent. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset I1's space parent to inode 2, the root of the file system.

Possible responses to the `reset?` prompt are:

YES	Fix this error condition by resetting I1's space parent to inode 2, the root of the file system.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I needs to be expanded — fix?

The directory inode I needs to be expanded so that another directory entry can be added to it; I is either the root directory or the **/lost+found** directory. If run with the **-p** or **-q** options, **fsck** will automatically attempt to expand the directory. Otherwise, **fsck** will ask to expand it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by attempting to expand inode I.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I is unreferenced — clear?

The inode I has no links in the file system and an earlier reconnection failed or was refused.

Possible responses to the `clear?` prompt are:

YES	Fix this error condition by clearing inode I. The contents of the file will be destroyed.
NO	Ignore this error condition. Inode I will remain unattached and can be reattached during a later **fsck** session provided that enough blocks and/or inodes are free.

Inode I is unreferenced — reconnect?

The inode I has no links in the file system. If run with the **-p** option, **fsck** will automatically attempt to reconnect the file. Otherwise, **fsck** will ask to reconnect it.

Possible responses to the `reconnect?` prompt are:

YES	Fix this error condition by reconnecting inode *I* in the **/lost+found** directory, with the name "#*N*", where *N* is the inode number of I.
NO	Ignore this error condition.

The lost+found directory inode I already has an entry named 'F' — remove bad directory entry?

The **/lost+found** directory inode I has discovered that it already has an entry of the name F when it was trying to reconnect an unreferenced file which would have had the same name. If run with the **–p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the spurious entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES	Fix this error condition by removing the entry for F; the inode referenced by that entry will be reattached with a name constructed from its inode number.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors during phase 4 – check link counts and resource accounting

This phase checks the link counts of individual inodes and the resource counts (blocks and inodes used) of control point directories. The messages result from errors in these counts.

Control point directory inode I has incorrect inode allocation count (N1 should be N2) — fix?

The control point directory inode I has a bad count of the inodes used by it and all its space descendants. If run with the **–p** or **–q** options, **fsck** will adjust automatically the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by adjusting the inode count for inode *I* to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Control point directory inode I has incorrect space allocation count (N1 should be N2) — fix?

The control point directory inode I has a bad count of the blocks used by it and all its space descendants. If run with the **-p** or **-q** options, **fsck** will adjust automatically the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by adjusting the space count for inode I to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has incorrect link count (N1 should be N2) — fix?

The inode I has a bad link count. If run with the **-p** or **-q** options, **fsck** will adjust automatically the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by adjusting the link count for inode I to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors during phase 5 - check disk allocation region information

This phase deals with the disk allocation regions. Messages in this section result from errors in the components of the DARs: the bitmap, the free inode list, and various resource counts.

Block B of the Disk Allocation Region Information Area is invalid — fix?

The disk allocation region information area block B does not contain the proper self-identification information. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting this block as an empty disk allocation region information area block. The DAR information in the block will be corrected later in this Phase.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect Bitmap — fix?

The bitmap for DAR *N* is incorrect. If run with the **-p** option, **fsck** will correct automatically the bitmap. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the bitmap correctly.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect count of blocks used (N1 should be N2) — fix?

The block count for DAR *N* is incorrect. If run with the **-p** or **-q** options, **fsck** will correct automatically the count to *N2*. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by changing DAR *N*'s block count from *N1* to *N2*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect counts of directories and inodes used — fix?

The counts of used files and directories for DAR *N* are incorrect. If run with the **-p** or **-q** options, **fsck** will correct automatically the counts. Otherwise, **fsck** will ask to correct them.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the counts of used inodes and directories correctly.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect free inode list — fix?

The linked list of free inodes in DAR *N* is incorrect: it contains allocated inodes, duplicates, or it does not contain some inodes that

are actually unallocated. If run with the **-p** or **-q** options, **fsck** will correct automatically the free list. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by rewriting the free list for DAR *N*.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has invalid contents in its reserved area — fix?

Disk allocation region number N has nonzero contents in its reserved area. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to zero out the reserved area.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by initializing the contents of the reserved area.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Incorrect summary counts in superblocks — fix?

The counts of used blocks and files in the two copies of the superblock are incorrect. If run with the **-p** or **-q** options, **fsck** will correct automatically the counts. Otherwise, **fsck** will ask to correct them.

Possible responses to the `fix?` prompt are:

YES	Fix this error condition by correctly rewriting the counts of used blocks and files.
NO	Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Advisory messages during file system cleanup

After a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system.

File System is now mountable

The **fsck** program has completed successfully checking the file system and it has been marked as mountable.

File System is still inconsistent and not mountable

The **fsck** program has completed checking the file system, but inconsistencies remain and the file system is still marked as unmountable. Re-run **fsck** to fix the remaining inconsistencies.

N1 of N2 blocks used (N3 free); N4 of N5 inodes used (N6 free)

The indicated number of blocks and inodes have been used, leaving the indicated number unallocated.

Unconnected files still remain. Mount the file system and remove files to free data blocks and inodes

The **fsck** program has completed successfully checking the file system and it has been marked as mountable. However, there are still unreferenced files in the file system. These unreferenced files can be recovered by running **fsck** again after enough blocks and inodes have been freed to allow them room to be reconnected.

Error messages exclusive to fast recovery checking

This section lists errors that can appear only during checking of a fast recovery file system, one for which **fsck** logging was enabled. These messages do not appear with phased **fsck** messages in the previous section because fast recovery checking does not occur in phases.

While normal phased checking will return only the messages in the previous section, fast recovery file system checking may return the messages in this section as well as many (but not all) of those appearing in the previous section.

Bitmap block N in Disk Allocation Region N has an invalid format

This message is printed when **fsck** discovers a bitmap block with a bad self-id. The arguments are the lda of the bitmap block and the number of the DAR it belongs to.

Block N is invalid Inode Table block

This message is printed when an Inode Table block fails to self-ID. The numeric argument is the invalid block.

Block N of the Disk Allocation Region Information Area is invalid

This message is printed when **fsck** discovers a DARIA which does not self-ID. The argument is the ordinal number (within the DARIA) of the faulty block.

Cannot allocate memory for internal tables (N bytes requested)

This message is printed when an attempt to allocate memory fails. The numeric argument is the number of bytes requested.

Cannot find enough contiguous free blocks to allocate a missing index element for inode I

This message is printed when **fsck** fails to allocate an index element. The argument describes the inode in question.

Cannot open fast recovery dump file F

This message is printed when the open of the file to hold the human-readable version of the fast recovery log fails. The argument is the pathname of the dump file.

Cannot read DAR table

This message is printed when the DAR table cannot be read.

Cannot read fast recovery log at lda B

This message is printed when the read of one of the fast recovery disk logs fails. The argument is the logical disk address of the log.

Cannot recover from log. Run full fsck

This message is printed when a fatal error occurs while trying to recover with the log.

Cannot write DAR table

This message is printed when the DAR table cannot be written.

Directory entry D in Directory inode I contains the wrong inode number (B should be B)

This message is printed when a directory entry is discovered to contain an incorrect file node number. The arguments are the directory entry file name, the file node number of the directory containing the entry, the entry's incorrect file node number, and the correct file node number.

Directory inode I has a spurious entry for file name F with file node number B

This message is printed when an unneeded directory entry is found. The arguments are the directory's file node number, the name of the missing entry and the file node number that should be in that entry.

Directory inode I has an incorrect parent inode number (B should be B)

This message is printed when a directory file node containing the wrong parent inode number in its din is found. The arguments are the directory's inode number and the incorrect and correct parent inode numbers.

Directory inode I has incorrect child count (N should be N)

This message is printed when **fsck** discovers an erroneous child count in a directory. The string argument describes the inode in question. The first numeric argument is the old, incorrect count; the second numeric argument is the correct count.

Directory inode I is missing an entry for file name F with file node number B

This message is printed when a directory entry is found to be missing. The arguments are the directory's file node number, the name of the missing entry and the file node number that should be in that entry.

Disk Allocation Region N has an incorrect Free Inode List

This message is printed when **fsck** discovers a DAR with incorrect Free File Node List. The first argument is the DAR number.

Disk Allocation Region N has incorrect Bitmap

This message is printed when **fsck** discovers an incorrect bitmap. The argument is the number of the DAR with the bad bitmap.

Disk Allocation Region N has incorrect count of blocks used (N should be N)

This message is printed when **fsck** discovers a DAR with incorrect space count. The first argument is the DAR number. The second argument is the incorrect, old count. The third argument is the correct count.

Disk Allocation Region N has incorrect counts of directories and inodes used

This message is printed when **fsck** discovers a DAR with incorrect directory or file node counts. The first argument is the DAR number.

File node and space accounting partially recovered. Run full fsck to recover accounting information

This message is printed when **fsck** completes but accounting recovery from the log was not complete. This error does not prevent the file system from being mounted, but at some point, full **fsck** should be run to fully correct accounting information.

Inode I has a bad fragment size for data element #0 (B should be B)

This message is printed when a file node's `fragment_size_exponent` field is found to contain an incorrect value. The arguments are the inode number, the bad fragment size exponent and the correct fragment size exponent.

Inode I has an Index Block (N) with invalid format

This message is printed when **fsck** discovers an element pointing to an index block which does not self-ID. The string argument describes the inode in question. The numeric argument is the bad block's number.

Inode I has an incorrect file size (N bytes should be N bytes)

This message is printed when **fsck** discovers a node having an incorrect file size. The string argument describes the inode in question. The first numeric argument is the old, incorrect size. The second numeric argument is the correct size.

Inode I has an incorrect space parent inode number (B should be B)

This message is printed when a file node containing the wrong space parent inode number is found. The arguments are the file's inode number and the incorrect and correct space parent inode numbers.

Inode I has bad element pointers

This message is printed when an element pointer in the file node is found to contain the incorrect address and the element is allocated. The argument is the inode number.

Inode I has incorrect inode allocation count (N should be N)

This message is printed when **fsck** discovers a node having an incorrect node count. The string argument describes the inode in question. The first numeric argument is the old, incorrect count. The second numeric argument is the correct count.

Inode I has incorrect link count (N should be N)

This message is printed when **fsck** discovers a node having an incorrect link count. The string argument describes the inode in question. The first numeric argument is the old, incorrect count. The second numeric argument is the correct count.

Inode I has incorrect space allocation count (N should be N)

This message is printed when **fsck** discovers a node having an incorrect space count. The string argument describes the inode in question. The first numeric argument is the old, incorrect count. The second numeric argument is the correct count.

Inode I is incorrectly marked as allocated

This message is printed when a file node allocation has to be backed out or the results of a deallocation did not make it to disk. The argument is the file node number.

Inode I is partially truncated

This message is printed when **fsck** discovers a partially truncated file. The string argument is the file in question.

Internal software bug O

This message is printed when a sanity check fails, indicating a software bug. The argument is a status encoded with FSCK_ENCODE_BUG_STATUS.

Missing log entry for file system fnode accounting

This message is printed when it is discovered that there are file node allocations or frees on the file system, but that no log entry giving the initial value of the file system's `number_of_used_file_nodes` was made. This error indicates that there is a bug in the kernel logging code.

System buffers containing data from the following files may not have been written to disk:

This message is printed when **fsck** fails to allocate an index element. The argument describes the inode in question.

Unexpected child count N in inode I

This message indicates a bug in the log recovery code.

Unknown kind of log entry

This message is printed when an unrecognized type of log entry is found in the log.

End of Chapter

13 Retrieving information about files and file systems

This section shows you how to display disk usage, check for security breaches, and find files.

Displaying disk space usage

This operation displays a table showing the number of blocks and inodes in use on mounted file systems that you specify.

1. To display disk space usage information, select the **sysadm** operation:

   ```
   File System -> File Information -> Disk Use
   ```

2. Specify one or more directories to check. You may enter multiple file system names. If you specify no file system names, the operation lists information for all file systems.

   ```
   Directories: ↵
   ```

3. Specify whether or not to restrict the list to local file systems.

   ```
   Restrict to Local File Systems? [yes] ↵
   ```

   Typical output follows:

```
             Free      Total     Pct      Free      Total     Pct
Directory    nodes     Inodes    Used     Blocks    Blocks    Used
----------   -------  --------   ----   --------  --------   ----
/             4758       5760    17%      12537     40000    68%
/usr         24996      34560    27%      16682    240000    93%
/usr/opt/X11 16884      20160    16%      14326    140000    89%
/usr/opt/fred   63         64     1%        310       330     6%
```

Checking for security breaches

This operation searches a directory tree for files that have suspicious ownership and permission settings. If you specify no directories, the operation checks the system's entire directory tree. This operation may be time consuming.

1. To check for breaches, select the **sysadm** operation:

   ```
   File System -> File Information -> Check
   ```

2. Specify one or more directories to check. You may enter multiple file system names. If you specify no file system names, the operation lists information for all file systems.

   ```
   Directories: ↵
   ```

3. Specify whether or not to restrict the list to local file systems.

   ```
   Restrict to Local File Systems? [yes] ↵
   ```

   The check operation finds files that may indicate that a security breach has occurred. This operation searches the directory you specify and reports files that have the following problems:

- Device files that exist outside **/dev**. No device files should reside outside **/dev** unless you created or moved device files for a special purpose, such as a test.
- Non-system files that are owned by the superuser (user with UID of 0, **sysadm** and **root** by default) and have the setuid bit set.

  The setuid bit is a special kind of permission attribute that all files have but which is useful only for executable files (such as programs and shell scripts—not directories or text files). When a user runs an executable that has the setuid bit set, the process runs with the effective user ID of the executable's owner—not, as is normally the case, with the user ID of the person running the executable.

  For example, if user **fred** executes a program owned by user **bob**, and this program does not have the setuid bit set, **fred**'s process runs with the effective user ID of **fred** (as normal). On the other hand, if user **fred** executes a program owned by user **bob**, and this program does have the setuid bit set, **fred**'s process runs with the effective user ID of **bob**. This means that the program, if it is so written, can access files to which **fred** may not normally have access, but to which **bob** does. User **fred** does not even need to know **bob**'s password for these accesses to occur.

  The rationale behind the setuid bit is to allow users to perform some action that they should not be able to perform under normal circumstances. The **lp** command, for example, has the setuid bit set so that any user who invokes **lp** to queue up a print job can, through the agency of the **lp** program, do things such as copy files to the LP system's directories and add requests to the LP scheduler's queue file.

  When you execute a program that has the setuid bit set, your actions are determined by the scope of the program and the user ID of the program's owner. Thus, there is a danger inherent in the setuid bit feature: the combination of a permissive program that has the setuid bit set, owned by a privileged user ID, may give a

user too much freedom on the system. This situation can result in a breach of security. The extreme case would be a shell program owned by **root** that has its setuid bit set; any user could execute the program and enjoy the use of a superuser shell throughout the system.

Among its various functions, the check operation includes a search for suspicious programs, ones owned by **root** that have the setuid bit set.

The following example file listing shows **ls -l** output for several files that have the setuid bit set (which is indicated by the **s** flag in the group-execute permission and/or owner-execute permission in the permissions line). The file **/sales/tom/nasty** is suspicious because it is owned by **root** but is obviously not a normal system program. It appears to belong to user **tom**. Depending on what Tom's program does, it may constitute a security breach.

```
-rwsr-sr-x  1 root  bin    44924 Mar 28 18:16 /usr/bin/at
-rwsr-sr-x  1 root  bin    29500 Mar 28 18:16 /usr/bin/crontab
---s--x---  1 root  users  95376 Aug 18 11:08 /sales/tom/nasty
```

Tom would never have been able to create such a program without superuser access. Thus, the program may not only constitute a security breach itself, but it also indicates the presence of a breach elsewhere, the one that allowed Tom to become superuser and produce the suspicious executable.

To protect your system, never leave your logged-in terminal unattended (particularly if you are logged in as **root** or **sysadm**). Another user could move or copy files, or commit any manner of destructive acts, all with your user ID.

When you find a setuid bit set, investigate further. You may need to correct setuid permissions with the **chmod** command.

If you do have a test device file outside **/dev**, you should either move or delete the device file.

For environments where you require a greater degree of security, there are the Trusted DG/UX systems, which provide B1 and C2 levels of security. For more information, see *Managing Security on the Trusted DG/UX™ System*. For more information on the Trusted DG/UX system product, contact your Data General representative.

Finding files

Use the find operation to search a specified directory and list all files or directories under it that satisfy specified criteria. The

operation can also sort the output data in various ways. (In this discussion, the term "file" also refers to directories.)

The find operation is useful as a part of everyday maintenance. You can use it to find:

- files whose names match a specific pattern,
- files belonging to particular users or groups,
- files of a given type,
- files that have not been accessed or modified in a long time and are no longer necessary, or
- files that take up too much space.

You can also determine the number and order of files found. You can sort the information by name, size, access date, and modification date.

1. To search a specified directory and list all files or directories under it that satisfy specified criteria, select the **sysadm** operation:

   ```
   File System -> File Information -> Find
   ```

2. List the directories you want to search. The operation searches the named directories as well as any directories beneath them. If you specify no directories, the operation searches the entire system, which can be quite time consuming.

   ```
   Directories:
   ```

3. Select this option to restrict the search to file systems that reside physically on your system. Without this option, the operation searches local directories as well as file systems mounted from remote systems.

   ```
   Restrict to Local File Systems:
   ```

4. Select this option to restrict the search to the file systems containing the named directories. Without this option, the operation searches all directories under the named directories as well as any file systems mounted under the directories.

   ```
   Restrict to This File System:
   ```

5. Specify name or name pattern to match. You may use the metacharacters (wildcards) accepted by the Bourne shell, **sh** (refer to the **sh**(1) manual page.

```
File Name:
```

These metacharacters are:

?	Matches any one character.
*	Matches any number of any characters.
[ ]	When surrounding a group of characters (not including the hyphen,–), matches any one character in the group. For example, **[abc]** matches an occurrence of **a**, **b**, or **c**. Use the hyphen to represent a range of characters. For example, **a–z** represents any lowercase letter. Follow the closing bracket with **!** to negate the set, causing the expression to match any character *not* in the set. For example, **[ag–iA12]!** matches any character except **a**, **g**, **h**, **i**, **A**, **1**, or **2**.

If the name contains metacharacters, surround it with quotation marks. For example, the pattern "**c*.[1–4ab]**" matches any file or directory name starting with **c** and ending with a dot followed by one of the characters **1**, **2**, **3**, **4**, **a**, or **b.**

6. Specify the login name or user ID number of an existing user. The operation finds only files that the user owns. Remember that usernames are case sensitive.

   ```
   Owner Name or ID:
   ```

7. Specify the group name or group ID number of an existing group. The operation finds only files whose group ownership is for the specified group. Like usernames, group names are case sensitive.

   ```
   Group Name or ID:
   ```

8. Specify the type of file to find. You may specify more than one type.

   ```
   File Type:
   ```

   The DG/UX file system and the find operation recognize these file types:

`any`	Any of the following types.
`regular`	A typical file such as a text file that is not a directory or any other type specified here.
`directory`	A directory.
`block special`	A device file created for block data access.
`character special`	A device file created for character (raw) data access.
`fifo (named pipe)`	A named pipe.

9. When a user modifies a file, the system records the date in the file's inode, the block containing data about the file. The find operation can use this information to search for files that have not been modified since a particular date.

   ```
   Days Since Last Modification:
   ```

10. A file's inode also includes the date the file was last accessed (read). The find operation can use this information to search for files that have not been accessed since a particular date.

    ```
    Days Since Last Access:
    ```

11. Specify a size limit for files. The operation searches for files larger than this size.

    ```
    File Size (bytes):
    ```

12. The operation lets you organize the data about files that it finds. You may choose to sort files by name, size, access time, or modification time. You can also specify increasing order or decreasing order for the sort, or you can specify no sorting at all.

    ```
    File Sorting Method:
    ```

13. To limit the number of files in the display, specify an upper limit. Specifying zero removes the upper limit.

    ```
    Maximum Number of Files to Report:
    ```

End of Chapter

A DG/UX mass storage device names

This appendix gives full descriptions of DG/UX device names for SCSI disk and tape drives, SCSI controllers, SMD and ESDI disk controllers, and NVRAM boards.

SCSI disk and tape device names

This section explains SCSI disk and tape drive and controller names, including disk-array storage system drive and disk-array storage system drive and controller names.

Disk and tape drive names

The format for SCSI disk and tape drives follows.

device (*controller-name*,[[*vme-controller*] *controller-number controller-SCSI–ID*], *device-SCSI-ID*, *LUN*)

where:

device specifies the disk-array or the standard integrated SCSI device. The valid values, alphabetically, are

- **da** **hada**-type, 30-module disk-array subsystem (this is not the same as a CLARiiON™ disk-array storage system)
- **sd** SCSI disk (on an AViiON CSS or PHU, or CLARiiON disk-array storage system)
- **st** SCSI tape (on an AViiON CSS or PHU, or CLARiiON tape-array storage system)

controller-name specifies the associated controller, such as **ncsc** or **dgsc**. See later section "SCSI controller names," for these values.

vme-controller specifies the VME controller channel, for example (**vme(1)**); needed only if the computer has more than one VME channel and this device is on a channel other than the first one.

controller-number is used for the short form, and the controller's address is used for the long form. Table A–2

lists the device codes, valid controller numbers supported (short form), and controller addresses (long form).

controller-SCSI–ID is necessary only if you connect two AViiON computers to a single drive in a dual-initiator configuration. See *Achieving High Availability on AViiON® System* for details about this configuration. Table A–1 lists default SCSI IDs.

IMPORTANT: Be careful to specify the correct SCSI adapter SCSI ID in the boot path. If your host's boot path specifies the SCSI ID of the SCSI adapter installed in the other system, the boot will fail and the SCSI bus will hang. If the SCSI bus hangs, an attempt by either system to access the SCSI bus will hang the system. To recover, reset the hardware and reboot.

device-SCSI-ID For a **sd** disk that is not part of a CLARiiON disk-array storage system or for a **st** tape that is not part of a CLARiiON tape-array storage system, this is the SCSI ID number on the disk or tape drive which is jumpered originally at the factory. You may have reset the SCSI ID in the meantime. Table A–1 lists the default SCSI IDs.

For a CLARiiON disk-array storage system, this field indicates the SP (storage-control processor) SCSI ID. For a CLARiiON tape-array storage system, this field indicates the SP SCSI ID. The disk-array storage system can have up to two SPs. You may select the SP SCSI IDs, as explained in the 014-series manual that accompanies the storage system.

For a **hada**-type, 30-module disk-array subsystem (**da** device), this field indicates the unit number attached to a physical disk in the disk-array subsystem. A single controller (I/O processor) supports up to 30 disk modules in such a subsystem, numbered 6-1F (hexadecimal).

Table A–1 Default SCSI ID numbers

Device Type	SCSI ID Number
First disk drive	0
Second disk drive	1
Third disk drive	2
Fourth disk drive	3
First SP (disk-array storage-control processor)	0
First cartridge tape or tape array SP	4
Second cartridge tape or tape array SP	5
Third cartridge tape if using a single controller	6
First controller or SCSI-2 adapter channel	7
Second controller or SCSI-2 adapter channel	6
	F
	E
	D
	C
	B
	A
	9
	8

See the next section on SCSI ID numbering for more information on wide SCSI devices.

IMPORTANT: SCSI IDs F–8 can be used only if you are using DG/UX 5.4R3.10 and have wide SCSI devices. Releases prior to DG/UX 5.4R3.10 do not support wide SCSI devices.

These default SCSI ID number assignments are conventions only. Any 16-bit (wide) adapter can support SCSI IDs 0–F as long as all devices and controllers have unique SCSI IDs; 8-bit host adapters can support SCSI IDs 0–7 only.

Disk drives include the following: Winchester hard disks (which include customer replaceable disk-array disk modules), 3.5-inch diskettes, 5.25-inch diskettes, compact disk read-only memory (CD-ROM), write-once read-many (WORM), and erasable magneto-optical disk.

LUN — For the **sd** and **st** drives, this is the logical unit number of the drive that shares the controller SCSI ID. For example, a controller on SCSI ID 0 may support up to 8 logical units, LUNs 0 through 7. The *LUN* field serves primarily for distinguishing disk-array storage system disk drives, diskette drives (except Model 6880 drives; see the Model 6880 manual), and optical disk drives. For the first disk-array storage system on a **dgsc**, you can assign physical unit numbers in the range 0 through 1F hex.

For examples of disk and tape device names, see the section on SCSI controller names.

SCSI ID numbering

DG/UX 5.4R3.10 introduces use of the wide SCSI bus (cable), which supports eight more devices at SCSI ID locations 8 through F. Wide SCSI refers to the 16-bit data bus, high-density, single cable interface. Narrow SCSI refers to the 8-bit SCSI data bus, low-density single cable interface that traditionally supported devices at SCSI ID locations 0 through 7.

Correspondingly, you may charactertize devices (including initiator devices) as wide and narrow. You may arrange both narrow and wide devices on the same bus (wide or narrow), but you must adhere to certain cabling and device ordering rules to produce a reliable interface. Your hardware installation documentation explains these rules.

If you need to rejumper a device at some later date, be mindful of the following guidelines:

- The use of a narrow initiator on a bus restricts all other SCSI IDs on that bus to 0 through 7.
- The system services device I/O requests according to the winner of a priority-based arbitration scheme. If four devices, for example, submit an I/O request at the same time, the system awards the request for service to the device whose SCSI ID has highest priority. The system assigns SCSI IDs the following priority, from high to low: 7, 6, 5, 4, 3, 2, 1, 0, F, E, D, C, B, A, 9, 8. By default, the initiator at SCSI IDs 7 and 6 has the highest priority. The next highest priority goes to SCSI tape devices (4 and 5), which traditionally are slower than disks. Disks get the lowest priority because they are fast devices. When assigning SCSI IDs, be aware of the service priority you attach to the device. If you have tape

devices, be sure to assign them the second (to initiators) highest priority possible. Otherwise, fast devices usually will win a contest for service, and tape devices may have a longer wait before the system grants it requests for service.

SCSI controller names

The format for SCSI controller names follows.

device[**@***device-code*] ([*vme-controller*] *controller-number* [,*controller-SCSI-ID*])

where:

device is a mnemonic that identifies the standard SCSI adapter/controller. The valid values are

- **cisc** Ciprico VME SCSI adapter.
- **dgsc** Data General VME SCSI-2 adapter. When used with a disk-array storage system, it is called a SCSI-2 adapter.
- **hada** H.A.D.A., 30 disk-module disk-array I/O processor.
- **insc** Integrated SCSI adapter.
- **ncsc** NCR integrated SCSI-2 adapter.

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it is the interrupt vector. It is always expressed in long form (see Table A–2 below).

vme-controller specifies the VME controller channel, for example, **vme(1)**; needed only if the computer has more than one VME channel and this device is on a channel other than the first one. If you omit this, **vme(0),** is implied. For example, the device name **sd(dgsc(vme(0),2),3,1)** is functionally the same as **sd(dgsc(2),3,1)**. This item does not apply to the integrated controllers **insc** and **ncsc**.

controller-number is used for the short form, and the controller's address is used for the long form. Table A–2 lists the device codes, valid controller numbers supported (short form), and controller addresses (long form).

controller-SCSI-ID is an optional argument. The ID is required for a disk-array SCSI-2 adapter or in any other

controller in a dual-initiator (shared bus) configuration.

Table A–2 lists the standard SCSI adapter/controller device names alphabetically by device name. Devices with values outside the ranges shown are nonstandard.

Table A–2 SCSI adapter/controller device names

Adapter/Controller Number (Short Form)	Adapter/Controller Address (Long Form)
Ciprico VME SCSI Adapter (cisc)	
cisc(0)	cisc@28(FFFFF300)
cisc(1)	cisc@29(FFFFF500)
cisc(2)	cisc@2A(FFFFF700)
cisc(3)	cisc@2B(FFFFF900)
cisc(4)	cisc@2C(FFFFED00)
cisc(5)	cisc@2D(FFFFD700)
cisc(6)	cisc@2E(FFFFD900)
cisc(7)	cisc@2F(FFFFDB00)
cisc(8)	cisc@20(FFFFDD00)
cisc(9)	cisc@21(FFFFDF00)
cisc(A)	cisc@22(FFFFE100)
cisc(B)	cisc@23(FFFFE300)
cisc(C)	cisc@24(FFFFE500)
cisc(D)	cisc@25(FFFFE700)
cisc(E)	cisc@26(FFFFE900)
cisc(F)	cisc@27(FFFFEB00)
Data General VME SCSI-2 Adapter (dgsc)	
dgsc(0)	dgsc@30(FFFFC000,7)
dgsc(1)	dgsc@31(FFFFC080,7)
dgsc(2)	dgsc@32(FFFFC100,7)
dgsc(3)	dgsc@33(FFFFC180,7)
dgsc(4)	dgsc@34(FFFFC200,7)
dgsc(5)	dgsc@35(FFFFC280,7)
dgsc(6)	dgsc@36(FFFFC300,7)
dgsc(7)	dgsc@37(FFFFC380,7)
dgsc(8)	dgsc@38(FFFFC400,7)
dgsc(9)	dgsc@39(FFFFC480,7)
dgsc(A)	dgsc@3A(FFFFC500,7)
dgsc(B)	dgsc@3B(FFFFC580,7)
dgsc(C)	dgsc@3C(FFFFC600,7)
dgsc(D)	dgsc@3D(FFFFC680,7)
dgsc(E)	dgsc@3E(FFFFC700,7)
dgsc(F)	dgsc@3F(FFFFC780,7)

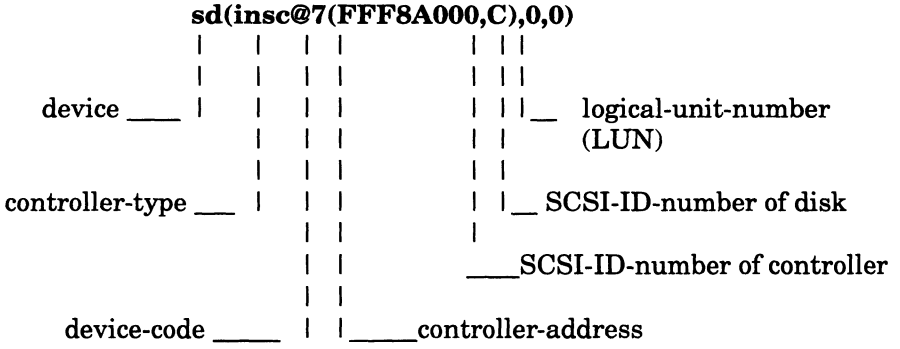
continued

Table A–2 SCSI controller/adapter device names

H.A.D.A Disk–Array I/O Processor (hada)	
hada(0)	hada@70(FFFF1000)
hada(1)	hada@71(FFFF1400)
hada(2)	hada@72(FFFF1800)
hada(3)	hada@73(FFFF1C00)
Integrated SCSI Adapter (insc)	
insc(0)	insc@7(FFFA000)
NCR SCSI Adapter (ncsc)	
ncsc(0)*	ncsc@7(FFFA0000)
ncsc(1)*	ncsc@A(FFFA2000)
ncsc(2)	ncsc@C(FFFA4000)
ncsc(3)	ncsc@E(FFFA3000)
ncsc(4)	ncsc@10(FFFA5000)
ncsc(5)	ncsc@8(FF7A0000)
ncsc(6)	ncsc@12(FF7A2000)
ncsc(7)	ncsc@14(FF7A4000)
ncsc(8)	ncsc@16(FF7A3000)
ncsc(9)	ncsc@18(FF7A5000)
* AViiON 500, 530, and 4600 computers use the following defaults:	
ncsc(0)	ncsc@13(FFFB0000)
ncsc(1)	ncsc@13(FFFB0080)

Examples 1 and 2 are short and long formats for the same device.

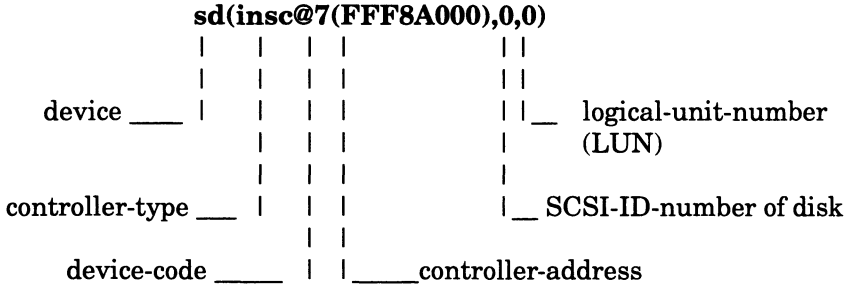
Example 1: SCSI disk on insc controller (short form)



The long name **sd(insc@7(FFF8A000,C),0,0)** identifies a disk having the SCSI ID of 0 on the integrated SCSI controller whose device code is 7, controller address is FFF8A000 and controller

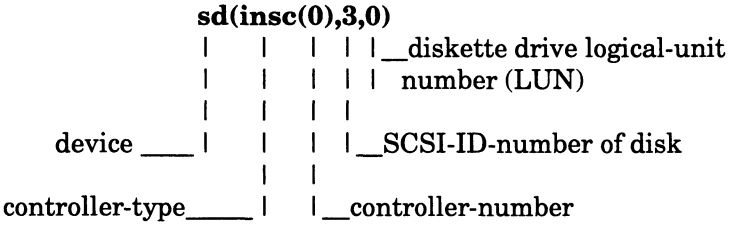
SCSI-ID is C. Since the disk is at LUN 0, it needn't be further qualified by a logical unit number, although 0 is shown for clarity. The short name for this device ican be expressed as **sd(insc( ))**, **sd(),0)** or **sd(0),0)**.

Example 2: SCSI disk on insc controller (long form)



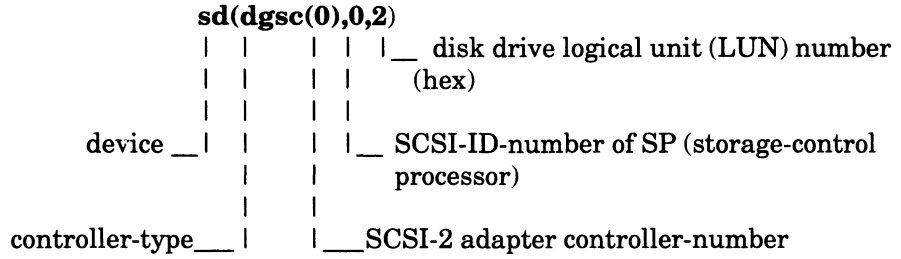
The controller address **sd(insc@7(FFF8A000),0,0)** identifies a disk having the SCSI ID of 0 on the integrated SCSI controller whose device code is 7 and controller address is FFF8A000. Since the disk is at LUN 0, it needn't be further qualified by a logical unit number, although we show the 0 for clarity. This device could also be specified as **sd(insc( ))**.

Example 3: SCSI diskette on insc controller (short form)



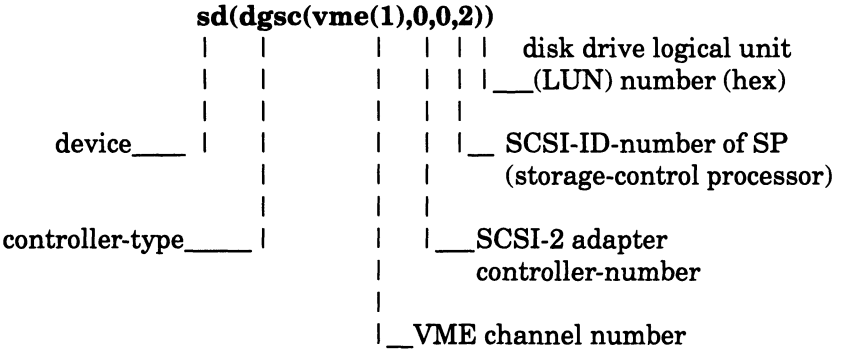
This device name identifies the first diskette drive (LUN 0) attached to a SCSI disk controller (at SCSI ID 3) on the integrated SCSI controller **(insc(0))**.

Example 4: SCSI disk in CLARiiON disk-array storage system on dgsc controller (short form)



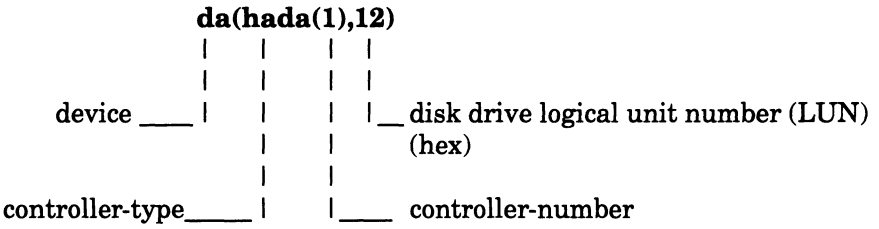
This device name identifies a disk drive in a CLARiiON disk array storage system. The SCSI-2 adapter controller is the first one in the host (number 0), the SP is SP A, the first one in the storage system (SCSI-ID 0), and the disk drive is logical unit (LUN) 2.

Example 5: SCSI disk in CLARiiON disk-array storage system on dgsc controller on second vme channel (short form)



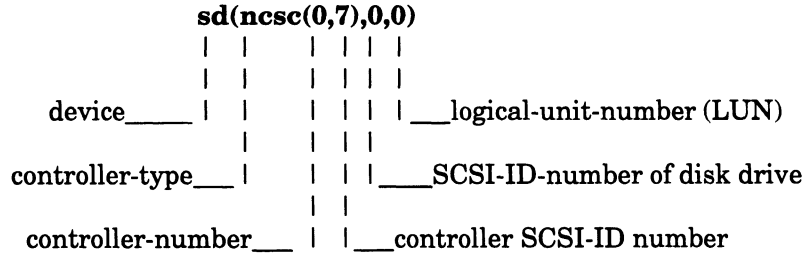
This device name identifies a disk drive in a CLARiiON disk array storage system on the second VME channel. The SCSI-2 adapter controller is the first one in the host's second VME channel **(dgsc(vme(1),0)**, the SP is SP A, the first one in the storage system (SCSI-ID 0), and the disk drive is logical unit (LUN) number 2.

Example 6: SCSI disk in hada-type disk-array subsystem on hada controller (short form)



This device name identifies a H.A.D.A. (30-module) disk array, disk unit 12, controlled by the second (1) I/O processor.

Example 7: SCSI disk on ncsc controller (short form)



This device name identifies a disk drive having the SCSI ID 0 on an **ncsc** controller whose SCSI-ID is set to 7. Since the disk is a LUN 0, it needn't be further qualified by a logical unit number, although we do show the 0 for clarity. You could specify this device as **sd(ncsc(),0), sd(ncsc())**, **sd(ncsc(0)**, or **sd(ncsc(0,7),0)**. The first controller number, the first SCSI ID number, and the logical unit number are 0. The SCSI ID of the **ncsc** controller is 7 by default.

More examples:

sd(insc(),*) All SCSI disks on the integrated SCSI controller. The asterisk is a DG/UX device-naming metacharacter that matches any character. It can be used only in the system file, not in a device name for a system (boot) disk or tape device.

st(cisc(1),4) Tape drive having the SCSI ID 4 on the second (1) Ciprico SCSI controller.

st(hada(1),4) SCSI tape device having the SCSI ID 4 on the second (1) **hada** 30 disk-module disk-array I/O processor.

ncsc(0) Integrated NCR SCSI controller.

For more explicit and detailed examples with the disk-array storage system, see the 014-series manual shipped with the storage system.

SMD and ESDI disk drive names

The format for the SMD and ESDI disk drives follows:

device [**@***device-code*] ([*vme-controller*] [*controller-address* [, *unit-number*]])

where:

device is a mnemonic that specifies the controller device to which SMD and ESDI disk devices are attached. The valid values are

cied Ciprico VME ESDI controller
cimd Ciprico VME SMD controller
cird Ciprico VME ESDI or SMD controller

IMPORTANT: The **cird** device name for the Ciprico controller is not recognized at the SCM prompt.

device-code VME interrupt vector number.

vme-controller specifies the VME controller channel, for example, **(vme(1)**; needed only if the computer

has more than one VME channel and this device is on a channel other than the first one. If you omit this, **vme(0),** is implied. For example, the device name **cied(vme(0),1,0)** is functionally the same as **cied(1,0)**.

controller-address specifies the controller's VME A16 address.

unit-number identifies the disk drive on the controller.

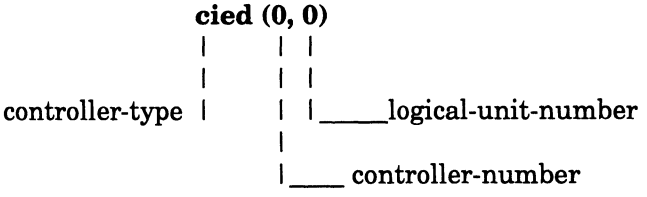
Table A–3 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form). Controllers with values outside this range are nonstandard.

Table A–3 SMD and ESDI disk drive controller names

Controller Number (Short Form)	Controller Address (Long Form)
Ciprico ESDI Controller (cied)	
cied(0)	cied@18(FFFFEF00)
cied(1)	cied@19(FFFFF100)
cied(2)	cied@1A(FFFFFB00)
cied(3)	cied@1B(FFFFFD00)
Ciprico SMD Controller (cimd)	
cimd(0)	cimd@18(FFFFEF00)
cimd(1)	cimd@19(FFFFF100)
cimd(2)	cimd@1A(FFFFFB00)
cimd(3)	cimd@1B(FFFFFD00)

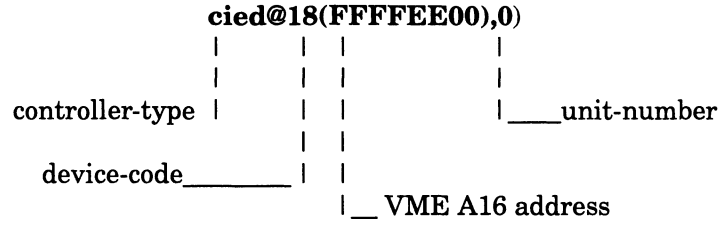
Examples 1 and 2 are short and long forms of the same disk drive.

Example 1: ESDI disk drive on cied controller (short form)



This device name identifies the first (0) disk drive attached to the first (0) Ciprico ESDI controller.

Example 2: ESDI disk drive on cied controller (long form)



This device name identifies the first (0) disk drive attached to the ESDI controller whose device code is 18 and whose VME A16 address is FFFFEE00.

Other examples

cied(0,2)	Third (2) disk drive attached to the first (0) Ciprico Rimfire controller.
cimd()	First disk drive attached to the first Ciprico Rimfire controller.
cird(3)	First SMD or ESDI disk drive attached to to the fourth (3) Ciprico Rimfire controller.

Non-volatile error correcting memory (NVRAM) device names

The format of a non-volatile error correcting memory device name for use as a non-volatile memory disk follows:

device [@*device-code*] ([*vme-controller*] *cr-address, memory–address, size*)

where:

device	is the mnemonic that specifies the non-volatile memory disk device. The only valid value is **nvrd**.
device–code	represents the interrupt vector.
vme-controller	specifies the VME controller channel, for example, **(vme(1))**; needed only if the computer has more than one VME channel and this device is on a channel other than the first one.
cr–address	is the VME control register address.
memory–address	is the VME memory address.
size	is the size of the memory board in megabytes. For example, a two-megabyte board will have 2 as its size.

Table A–4 lists the valid device short forms and long forms.

Table A–4 NVRAM device names

Controller Number (Short Form)	Controller Address (Long Form)
nvrd(0)	nvrd@80(FFFFFF00,E5000000,2)
nvrd(1)	nvrd@82(FFFFFF20,E5200000,2)
nvrd(2)	nvrd@84(FFFFFF40,E5400000,2)
nvrd(3)	nvrd@86(FFFFFF60,E5600000,2)
nvrd(4)	nvrd@88(FFFFFF80,E5800000,2)
nvrd(5)	nvrd@8A(FFFFFFA0,E5A00000,2)
nvrd(6)	nvrd@8C(FFFFFFC0,E5C00000,2)
nvrd(7)	nvrd@8E(FFFFFFE0,E5E00000,2)

Note that the device codes and control register addresses grow by 2 and 32 respectively. This provides support for mirrored non-volatile memory boards. If two boards are mirrored, only the device given in the preceding table needs to be configured; however, the mirrored board needs to have its DIP switches for its *device code*, *control register address*, and *memory address* set to values corresponding to the configured board. Determine these values using the following rules:

- The *device code* of the mirrored board is 1 greater than the *device code* on the configured board.
- The *control register address* is set to 16 greater than the *control register address* on the configured board.
- The *memory address* on the mirrored board must be set to the identical address on the configured board.

Examples 1 and 2 are the short and long form of the same device. Example 3 applies the preceding rules for mirroring a NVRAM board.

Example 1: Non–volatile memory device (short form)

```
                    nvrd(0)
                      |   |
device–name_          |   |___unit–number
```

Example 2: Non–volatile memory device (long form)

```
                    nvrd@80(FFFFFF00,E5000000,2)
                      |     |     |          |          |___Size in megabytes
device–name__         |     |     |          |
                      |     |     |          |___VME memory address
                      |     |     |___VME control register address
                      |     |___device code
```

Example 3: Values for mirror image of NVRAM board

To mirror **nvrd**(0), which was illustrated in the preceding two examples, you would set the DIP switches on the mirrored board accordingly:

Device Code	**81**
Control Register Address	**FFFFFF10**
Memory Address	**E5000000**

End of Appendix

B Virtual disk management (VDM)

DG/UX 5.4R3.00 introduced virtual disk management (VDM) technology, which enables system administrators to manipulate storage space with the data online and in use. This on-line convenience, in turn, enables you to dynamically create and rearrange disk resources for improved system performance and high availability of data.

The most important new feature that VDM offers is high availability. You can do almost all disk management operations on-line, providing users and applications uninterrupted access to data. Such operations include expanding file systems on-line, moving partitions from one physical disk to another on-line, and creating or removing software mirrors on-line. You do not have to shut down applications to perform these tasks, nor do you need to schedule them during times when there are no users on the system. VDM enables you to manage disks at your convenience — even in the middle of the work day.

Both the virtual disk management technology and its predecessor, logical disk management (LDM), enable you to subdivide physical disks so that you can store many types of information on different parts of a single physical disk.

With disk managers, you can subdivide the physical disk into discrete amounts of space, each of which can be used to store a different file system or database. Furthermore, both VDM and LDM allow you to combine amounts of storage from multiple physical disks to form storage areas that exceed the size of a single physical disk. They also enable you to replicate data through software mirroring and to set up fast caches.

VDM and LDM provide many of the same functions, but their implementations are significantly different. VDM can be thought of as the next generation of LDM, offering an expanded set of disk management options for you to use when organizing your data resources.

To create and maintain virtual disks, use the **sysadm** menu `Device -> Disk -> Virtual`. See the section on virtual disks in this manual for more information.

IMPORTANT: Be aware that DG/UX 5.4R3.10 supports VDM only, replacing entirely its predecessor, LDM. The current arrangement of **sysadm** menus and options is significantly different from releases prior to DG/UX 5.4R3.00.

If you already have upgraded to DG/UX 5.4R3.00 or R3.10, you may continue to use logical disks in a virtual disk environment in compatibility mode, which allows reading and writing to logical disks but does not support operations that re-create the disk's metadata, such as expanding, shrinking, moving, or mirroring.

If you currently are operating in compatibility mode and wish to convert fully from logical disk to virtual disk format, you may do so with the **sysadm** operation `Device -> Disk -> Physical -> Convert`. Refer to Chapter 10 for details.

End of Appendix

C Completing the disk planning worksheets

The appendix contains copies of worksheets that you completed in the preceding chapters. You may want to consolidate the completed worksheets here and/or use the copies in this appendix to supplement those in previous chapters. Worksheets are provided for:

- Virtual disks
- Mirror and cache
- Local file systems
- Remote file systems

Virtual disk worksheets

Figures C–1 and C–2 show two virtual disk planning worksheets.

Virtual Disk Layout Worksheet (Page 1 of)

Virtual Disk Name	Mount Point Directory	Striped ?	Size in blocks	Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes	
				Piece	Size in Blocks	Piece	Size in Blocks
Total Used							
Total Capacity							
Free Space							

Figure C–1 Virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

Virtual Disk Layout Worksheet (Page of)

Virtual Disk Name	Mount Point Directory	Striped ?	Size in blocks	Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes	
				Piece	Size in Blocks	Piece	Size in Blocks
Total Used							
Total Capacity							
Free Space							

Figure C–2 Virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

Mirror and cache worksheet

Figures C–3 and C–4 show two mirror and cache virtual disk planning worksheets.

Mirror and Cache Layout Worksheet (Page 1 of)

Mirrored Virtual Disk Name	Mount Point Directory	Mirrors Only: Striped Image being Mirrored ?	Size in blocks	Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes	
				Piece/ Image	Size in Blocks	Piece/ Image	Size in Blocks	Piece/ Image	Size in Blocks
Total Used									
Total Capacity									
Free Space									

Figure C–3 Mirrored and cached virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

Mirror and Cache Layout Worksheet (Page of)

Mirrored Virtual Disk Name	Mount Point Directory	Mirrors Only: Striped Image being Mirrored ?	Size in blocks	Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes		Drive Name _____ _____ Mbytes	
				Piece/ Image	Size in Blocks	Piece/ Image	Size in Blocks	Piec / Image	Size in Blocks
Total Used									
Total Capacity									
Free Space									

Figure C–4 Mirrored and cached virtual disk planning worksheet

1 Mbyte = 2,048 blocks.

Local file systems

Figure C–5 shows a worksheet for the file systems and mount points such as for a CD-ROM or DOS diskette.

Local File System Layout Worksheet

Device Type	Device Name	Local Mount Point

Figure C–5 Local file system planning worksheet

Remote file systems

Figures C–6 and C–7 shows a worksheet for remote file systems and mount points.

Remote File System Layout Worksheet (Page 1 of)

Remote Hostname	Remote Mount Point	Local Mount Point

Figure C–6 Remote file system planning worksheet

Remote File System Layout Worksheet (Page of)

Remote Hostname	Remote Mount Point	Local Mount Point

Figure C–7 Remote file system planning worksheet

End of Appendix

D Troubleshooting

This appendix covers these classes of errors:

- Soft SCSI-2 tape drive errors
- NVRAM board errors

Soft SCSI-2 tape drive errors

Soft tape errors are reported on the system console for SCSI-2 tape drives. These error reports are deferred, meaning the system does not report them as they occur. Instead, the system compares the number of bytes transferred and the number of errors that occurred, and reports according to the ratio of errors to bytes. If there are a large number of errors, the system reports an acceptance level of marginal. If there are a very large number of errors, the system reports an acceptance level of bad.

The DG/UX system logs these error messages, which by default goes to the system console and to **/usr/adm/messages**.

An example of a marginal message follows:

```
Feb  1 17:00:03 blatz dg/ux: Tape device at
st(ncsc@7(FFFB0000,7),3,0) encountered a high number of
correctable (soft) errors.
Feb  1 17:00:03 blatz dg/ux: Please observe the suggested
maintenance schedule for the drive.
```

Suggested maintenance includes cleaning the drive heads.

An example of a bad message follows:

```
Feb  2 17:20:03 blatz dg/ux: Tape device at
st(ncsc@7(FFFB0000,7),2,0) encountered an unacceptably high
number of correctable (soft) errors.
Feb  2 17:20:03 blatz dg/ux: Please clean the tape drive and use
a known good tape.
Feb  2 17:20:03 blatz dg/ux: If you receive this message
frequently, contact your DG service representative.
```

NVRAM errors

NVRAM board errors are reported on the system console for DG/UX system with caching configurations.

Non Volatile RAM Module nvrd@*long-device-name*:
battery 3 failed.

The battery is near expiration. Refer to the NVRAM board hardware documentation for information on installing a new battery.

For all other error messages, refer to the NVRAM hardware manual for recovery actions.

Non Volatile RAM Module nvrd@*long-device-name*:
battery 2 not enabled. Check jumpers and switches.

Non Volatile RAM Module nvrd@*long-device-name*:
encountered a read failure at *address*.

Non Volatile RAM Module nvrd@*long-device-name*:
corrected a single bit error.

Non Volatile RAM Module
nvrd@*long-device-name*:detected an uncorrectable
multi-bit error.

End of Appendix

E Introduction to interfaces

You may use these interfaces to perform operations on mass-storage devices:

- **sysadm** (system administration utilities)
- administrative shell commands

Sysadm

You can use the **sysadm** menu-driven utility to manage your DG/UX system. Of the nine major menus, you will use two to manage devices: Device and File System. **Sysadm** is offered in two forms: stand-alone and stand-among.

When to use stand-alone sysadm

You use stand-alone **sysadm** (located at **/usr/stand/sysadm**) while the DG/UX system is not running. You boot this version to perform operations that you cannot perform while running the installed version of the DG/UX software.

If, for example, the / (root) or **/usr** file systems become damaged and you cannot boot the system, use stand-alone **sysadm** to repair them and then retry booting the system. Refer to Chapter 12 for information on repairing damaged DG/UX file systems. You must take the system down, and then boot stand-alone **sysadm** from the SCM. An example follows:

```
SCM> b  sd(cisc(),0)usr:/stand/sysadm ↵
```

You can boot stand-alone **sysadm** from disk only if the virtual disks containing the **/usr** file system are located on a single physical disk. If **/usr** spans multiple physical disks, you cannot boot from **/usr**.

You may also use stand-alone **sysadm** to copy the physical disk that contains the / file system to another physical disk.

Figure E–1 shows the stand-alone **sysadm** main menu.

```
                Standalone Sysadm Main Menu

  1  Physical Disk ->          Manage physical disks
  2  Virtual Disk ->           Manage virtual disks
  2  File System ->            Manage file systems
  4  Install Software ->       Install system software


Enter a number, a name, ? or <number>? for help, <NL> to redisplay
menu,or q to quit: [Install Software]:
```

Figure E–1 Stand–alone sysadm main menu

Shell commands supported by stand-alone sysadm

Table E–1 lists the shell commands in **/sbin** that stand-alone **sysadm** supports:

Table E–1 Shell commands in /sbin

fsck	init	reboot	sh
halt	mount	unmount	su

Table E–2 lists the shell commands in **/usr/sbin** that stand-alone **sysadm** supports:

Table E–2 Shell commands in /usr/sbin

devnm	gridman	swapon
dg_sysctl	mkfs	syslogd
exportfs	probedev	xdrtoc

Table E–3 lists the shell commands in **/usr/bin** that stand-alone **sysadm** supports:

Table E–3 Shell commands in /usr/bin

admdefault	cut	idi_confirm	rmdir
admdevice	date	idi_doop	sde_target
admfilesystem	dc	idi_echo	sed
admkernel	dd	idi_error	sort
admpackage	df	idi_log	stty
admpdisk	diff	idi_warning	sync
admrelease	dirname	ifconfig	tail
admservice	du	kill	tar
admtape	ed	ln	tee
admvdisk	egrep	logger	touch
awk	expr	ls	tput
basename	false	mkdir	tr
cat	find	mt	true
chgrp	grep	mv	tty
chmod	gunzip	netinit	uncompress
chown	gzip	newaliases	uniq
comm	head	ping	who
compress	hostname	pmtd	xargs
cp	id	printf	
cpio	idc	pwd	

When to use stand-among sysadm

By default, use this version, **/usr/sbin/sysadm**, while the DG/UX system is booted and running at init level 1 or higher. If you are not logged in as superuser, some **sysadm** operations will be restricted from use. Restricted operations are shaded in the graphical version and surrounded by brackets in the ASCII version.

To begin DG/UX system administration, become superuser and start **sysadm** as follows:

```
# sysadm ↵
```

If you are using a graphical workstation or an X terminal, the graphical from of stand-among **sysadm** is invoked automatically as shown in Figure E–2:

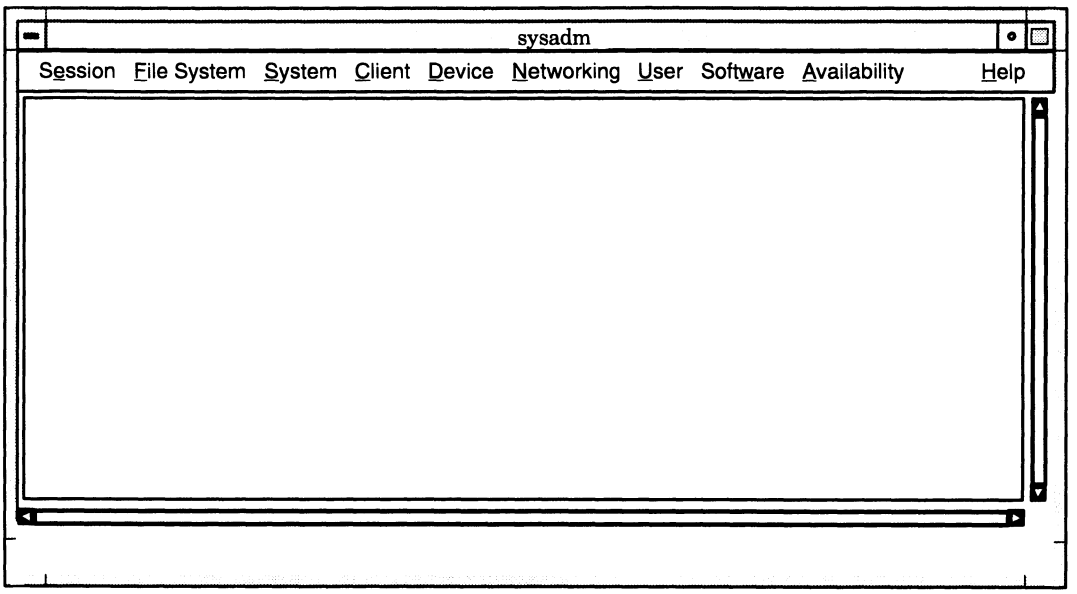


Figure E–2 The graphical version of the sysadm main menu

If you are working from an ASCII terminal (or if you enter **asysadm** instead of **sysadm**), an ASCII version of **sysadm** appears as shown in Figure E–3:

```
                Main Menu

  1  Session ->          Manage this sysadm session
  2  File System ->      Manage file systems
  3  System ->           Manage DG/UX system databases
  4  Client ->           Manage OS and X terminal clients
  5  Device ->           Manage devices and device queues
  6  Logging ->          Manage system and network logging
  7  Networking ->       Manage the network
  8  User ->             Manage users and groups
  9  Software ->         Manage software packages
 10  Availability ->     Manage high availability features
 11  Help ->             Get help on sysadm and its queries

Enter a number, a name, ? or <number>? for help, <NL> to redisplay
menu,or q to quit:
```

Figure E–3 The ASCII version of the sysadm main menu

The graphical and ASCII interfaces are functionally equivalent. Illustrations in this manual are taken from the ASCII version of **sysadm**.

Selecting menu options

In the graphical interface, you select using any of the following:

`Mouse`	With the mouse pointer on your choice, click the left button.
`Keyboard`	From the Main window, type **alt**–*letter*, where *letter* is the underlined letter (**alt–d** for Device, for example). From a pull-down window, type *letter* (**p** to select Disk on the Device pull-down menu, for example).
`Arrow keys`	From the Device pull-down, with a box around Disk, press the right arrow key to see the Disk functions.

In the ASCII interface, you can select menu options by entering the number of a menu item or by typing an abbreviation of the menu item name. For example, from the Main Menu, typing "de" and pressing Enter selects Device and "Fi" followed by Enter selects File System.

You can start the ASCII interface in the window you want to use. For example, to start **asysadm**, skipping directly to the `Device -> Physical` menu, enter:

```
# sysadm -m Device:Physical ↵
```

The system invokes the appropriate interface — the ASCII version (**asysadm**) or graphical version (**xsysadm**) of the **sysadm** utility — based on your display type. If you are working in an X windowing environment, invocation of **sysadm** will produce an X version of **sysadm**; when working in an ASCII terminal environment, by default, you will get an ASCII version of **sysadm**. You may choose explicitly the desired interface, however, by invoking either **xsysadm** or **asysadm**.

In this manual, the following convention is used to denote a menu traversal: the selection of the Physical option on the Device menu, you will: select `Device -> Disk -> Physical`. Depending on the interface you're using and your starting point, you interpret the instructions in one of the following ways:

- click Device in the **sysadm** main menu, Disk in the Device pull-down menu, and Physical in the Disk pull-down menu
- enter `5` from the **asysadm** main menu, `2` from the Device menu, and 1 from the Physical menu
- start the utility with the command:

```
# asysadm -m Device:Disk:Physical ↵
```

When repeating the same **sysadm** operation from the graphical interface (for example, creating mount points for several dozen remote file systems), you can save mouse clicks by using a tear-off menu rather than the equivalent pull-down menu. There is a dashed line at the top of the Remote File System pull-down menu. This indicates that the pull-down menu is optionally a tear-off window. If you select the dashed line, the pull-down menu becomes a tear-off menu at the new location and remains in place on your screen until you close it.

With both a pull-down menu and a tear-off menu, you return to the **sysadm** main menu each time you complete an operation.

Getting context-sensitive help

For descriptions of general topics, select Help. Thus, for a description of menu option selection methods and tear-off windows discussed in the previous section, select `Help -> On Interface`.

For help messages describing particular details, use the **sysadm** context-sensitive help facility. At any **sysadm** prompt, you can either enter the requested information or request an explanation. In the graphical interface, press function key 1 (F1) to request a help message. In the ASCII interface, type a question mark (`?`) and press Enter. (In the ASCII interface, obtain a description of a menu by entering *n*`?`, where *n* is the number of the menu.)

For example, when creating a virtual disk, you see the prompt:

```
Stripe Size (in blocks): [16]
```

You can enter a value or you can first request an explanation of the prompt by pressing F1 if you are using the graphical interface or by typing `?` and pressing Enter if you are using the ASCII interface.

Table E–4 presents a summary of the methods for making ASCII **sysadm** menu choices.

Table E–4 Making ASCII sysadm menu choices

User Input	Description
number	Choose menu item by entering *number.*
name	Choose menu item by entering full name of menu item, such as **Session**, or a string fragment that uniquely identifies the menu item such as **Ses** or **ses** for **Session.** The string is not case-sensitive.
names-separated-by - colons	Specify menu traversals by using multiple menu names, with the names separated by colons. Again, the case of characters is not significant. For example, **Software:Package:Install** or **So:Pack:In**.
?	Print help message, then redisplay menu prompt.
*number***?**	Print help message for a particular menu item, then redisplay menu prompt.
q	Exit from **sysadm** from any menu.
New Line/Enter key	Redisplay menu.
^ or ..	Return to the next higher menu.
:	Return to the main menu.

When an operation presents a query, a default response often appears within brackets. For example, [yes] indicates an affirmative response if you press Enter.

IMPORTANT: When only a predetermined set of responses is appropriate, use the **?** key to display all your choices. You may then select your choice by number or by a unique string.

After you select an operation and enter any information that it requires, **sysadm** either performs the action immediately or asks for confirmation before performing a potentially destructive action.

Sysadm provides on-line help in several ways. At the main menu, the Help menu offers information on: **sysadm**, the interface, the **sysadm** version, and help itself.

In addition to the Main Menu help option, each menu and operation in **sysadm** has a help message. Enter **?** to get help about the current menu, or enter a menu selection number followed by the **?** key to get information about a particular selection. You may also use **?** to get help and syntax information from any query.

Returning to the shell

During a **sysadm** session, you can return easily to the shell (Bourne, C, or Korn). How you return depends on whether you are using the graphical or the ASCII terminal **sysadm**.

From graphical **sysadm**, the shell runs as a background job and occupies a separate window. To return to the shell, simply move the mouse back to the shell prompt in the shell window. To return to **sysadm**, move the cursor back to the **sysadm** window.

From an ASCII-based **sysadm** prompt, escape to the shell by executing the appropriate shell escape command. An example of a C shell escape follows:

```
Enter a number, a name, ? or <number>? for help, or
   q to quit: !csh↵
```

Use **!sh** for a Bourne shell escape and **!ksh** for a Korn shell escape.

Return to ASCII **sysadm** by typing

```
% exit ↵
```

Using administrative shell commands

As an alternative to using the **sysadm** interface, you may prefer using administrative shell command equivalents. If you perform disk or file management operations repeatedly, you may prefer to use administrative shell commands in shell scripts for repeated execution. The primary shell command equivalents are: **admvdisk**, **admpdisk**, **admfilesystem** and **admdevice**. Refer to the respective 1M level manual page for details on each command's arguments and options.

End of Appendix

Glossary

The terms defined in this section are important to the mass-storage devices in the DG/UX environment.

/dev Administrative directory containing entries for all devices on the system.

/etc/fstab The **fstab** file is the file system table describing both local and remote file systems that are accessible on the local system and swap areas. File systems listed in **fstab** become accessible automatically at boot time. The **fstab** file contains information for commands that mount, unmount, backup, restore, and check file systems.

aggregation A type of virtual disk. An aggregation is the OSM equivalent of the LDM multi-piece logical disk. You may combine virtual disks of any type to form an aggregation — partition, other aggregations or mirrors — whose total size is the sum of the sizes of its constituents. An aggregation can comprise up to 120 partitions on one or more physical disks. You can stripe an aggregation if each child component is identically sized. *See also virtual disk.*

back-end device Used in a caching configuration in which a virtual disk containing data located on a large, slow device (typically a slow disk) is associated with a virtual disk on a small, fast device (typically a NVRAM board or a fast disk). The back-end device is the slow device in a caching configuration. *See also cache and front-end device.*

cache A type of virtual disk. Association of two or more data storage devices, typically a small, fast one with a large, slow one, so that an application uses the fast device for read and write operations while the operating system duplicates these operations on the larger device. Caching offers the benefits of speed from the fast device and capacity from the large device. *See also front-end device and back-end device.*

compatibility mode Way in which logical disks on physical disks remain in operation in a DG/UX 5.4R3.00 configuration. When upgrading from a release earlier than DG/UX 5.4R3.00, **sysadm** offers to convert all your physical disks to the new format. If you tell **sysadm** to not convert a disk, DG/UX will register it in compatibility mode. This mode permits reading from and writing to the logical disk but prevents operations that enlarge, shrink, move or mirror the logical disk's metadata. *See also logical disk.*

directory structure Arrangement of hierarchically structured file systems. The DG/UX directory structure base is the root directory, pathname /. Directories within the root include **usr**, **bin**, and **opt**, with pathnames **/usr**, **/bin**, and **/opt**.

disk drive Hardware medium used for storing data. Capacity of the drive is measured in megabytes (Mbytes) or gigabytes (Gbytes). Each drive has a model number and a DG/UX device name. Disk drive types include Winchester hard disk (usually mounted in a computer cabinet, combined storage subsystem — CSS — or disk-array storage system), magneto-optical disk, CD-ROM (compact disk-read-only memory), WORM (write-once read-many) optical disk, and diskette. Only Winchester hard disk and CD-ROM drives are bootable.

export To make a local file system available for mounting by other systems in your network. To export a local file system or mount an exported file system from a remote system, both systems must be running the ONC/NFS software.

failover The act of transfer from a failed hardware component to a working component. Failover pertains primarily to disk drives in a disk-array storage system, although it can also apply to tape drives on a shared SCSI bus. *See Achieving High Availability on the DG/UX™ System for details.*

fast recovery file system A file system mounted with the **fsck** logging feature. The system records file system modifications to a log, which **fsck** uses during recovery after a failure. Using the log, **fsck** can check and repair the file system faster than it can without the log. *See also fsck utility.*

file system Organization of data and work areas in a hierarchical tree-like structure. File system refers to both the entire tree known as the DG/UX file system and to each component of the tree which contains its own directory structure. Typically, you create a file system on each component in the hierarchical structure known as a virtual disk. The file system contains the internal data structures that the operating system requires to keep track of files and directories on the virtual disk. You create a virtual disk and an implicit file system structure using the **sysadm** operation `Device -> Virtual -> Create`. Then you create a mount point for virtual disk's file system using **sysadm**'s `File System -> Local Filesys -> Add` operation. This operation adds the file system's mount point and other important attributes to the **/etc/fstab** file, allowing it to be mounted automatically each time the system boots. *See also virtual disk, mount point, and **/etc/fstab** file.*

front-end device Used in a caching configuration in which a virtual disk located on a small, fast device (typically a NVRAM board or a fast disk) is associated with another virtual disk on a large, slow back-end device (typically a slow disk). The front-end device is the fast device in a caching configuration. *See also cache and back-end device.*

fsck utility File system checker utility that verifies the internal structure of a file system that is suspected of being corrupted. A file system can show signs of corruption while the system is booting or during normal operation. If a file system cannot be mounted during normal operation or if a file system becomes inaccessible, the file system probably is corrupt. You run **fsck** to repair the file system's inconsistencies, thereby enabling a successful file system mount. *See also fast recovery file system.*

hard mount Hard-mounted remote directories (or file systems in UNIX terminology) that cause programs to retry a mount request until the server responds. Programs that access hard-mounted files will hang as long as the server fails to respond. *See also soft mount.*

inode Data structure containing information about a file such as file type, size, date of creation, owner ID, and group ID. The number of inodes represents the total number of files that can exist on the system. The **mkfs** program, which creates a file system, accepts options that you can use to control the number of inodes in a file system. An inode is 126 bytes long, and there are 4 inodes to a disk block.

logical disk A span of disk space that you partition with **sysadm** so the DG/UX system can use it: it is analogous to the DG/UX 5.4R3.00 virtual disk. You can use a logical disk with a DG/UX system that uses virtual disks, but only in limited ways (compatibility mode). A logical disk can comprise multiple pieces spanning up to 32 noncontiguous areas on one or more physical disks. *See also virtual disk and compatibility mode.*

LDM Logical disk manager. The predecessor to OSM (on-line storage management), LDM allows access and manipulation of logical disks built on physical disks. LDM takes logical disks off line when changing disk storage options, rendering the data unusable for the duration of the operation. *See also OSM and logical disk.*

logical unit number (LUN) Logical unit number, a hexadecimal number that helps identify a physical disk. With a disk-array storage system, the LUN becomes part of the disk device name; generally, with other disks, the LUN is not important. For example, the 3 in the name for the disk-array storage system, **sd(dgsc(0),0,3)**, represents the LUN. The term disk drive means the same thing as physical disk and logical unit.

mirror A type of virtual disk. A collection of identically-sized virtual disks (in OSM) or logical disks (in LDM) that are duplicates (copies) of each other. In OSM, you can create up to three multi-image mirrors from partitions, aggregations, other mirrors, caches, or any combination of the same virtual disk types. In LDM, you can mirror only logical disks. Maintenance of one or more copies of a virtual disk to provide continuous access if the original virtual disk becomes inaccessible. Each copy is called an image. Mirroring assures continuous access to data; if an image goes bad, your system and user applications continue running on the good image without interruption.

There are two kinds of mirroring, software and hardware. With software mirroring, the operating system synchronizes the images. You can implement software mirroring with any group of disks, as described in this manual. With hardware mirroring, the disk controller hardware synchronizes the disk images. If you have a disk-array storage system, you can implement hardware mirroring by binding disk modules as a RAID-1 mirrored pair. Refer to a discussion of binding as described in the *CLARiiON™ Series 2000 Disk-Array Storage System with the DG/UX™ Operating System.*

mount To attach a file system to a directory, making it accessible to users. The system mounts directories listed in the **/etc/fstab** file at boot time. You can mount directories explicitly with the shell command **mount** or with **sysadm**'s mount operation. You mount local file systems as well as remote ones. To mount remote file systems, you need the ONC/NFS network software. *See also /etc/fstab file and mount point .*

mount point Placement of a file system within the DG/UX directory structure. Mounting a file system attaches it to a directory and makes it accessible to users. At startup, the DG/UX system automatically mounts all file systems noted in the file system table file, **/etc/fstab**. Operations such as backup, restore, check, or disk use require the name of the mount point directory. *See also /etc/fstab file and mount point.*

OSM On-line storage manager. A software technology allowing the access and manipulation of software virtual disks on physical disks while the data is on line and in use. Also, it offers a flexible hierarchical structure that allows grouping resource configurations into higher-level configurations at no performance or availability cost. *See also LDM and virtual disk.*

partition A type of virtual disk. OSM equivalent of the logical disk piece, it is the specific, contiguous space reserved on a physical disk for later data storage. You can create a partition from a physical disk or partition from an existing virtual disk. The difference between a partition and a logical disk piece is that you can mount a partition; you cannot mount a logical disk piece. *See also virtual disk, OSM, logical disk and mount point.*

pass number A number from a file system's **fstab** entry that indicates the order in which the **fsck** program checks file systems for corrupted and damaged files. In the first pass, **fsck** simultaneously checks all file systems with a pass number of 1. In the second pass, **fsck** checks file systems with a pass number of 2, and so on. *See also fsck utility.*

physical disk Hardware medium on which logical disks (or virtual disks) reside.

RAID Redundant array of inexpensive disks. A technology that groups individual disks into one unit to improve performance and/or reliability. Applies to disks in disk array storage systems, which can be combined (bound) in RAID levels 0, 1, 3, or 5. Refer to a discussion of RAID as described in the *CLARiiON™ Series 2000 Disk-Array Storage System with the DG/UX™ Operating System.*

read-write mode Access permission that allows users to write (change) files and directories in the file system as well as read them. This mode does not override the normal permissions that already apply to the files and directories.

read-only mode Access permission that allows only reading of a file system.

registration The identification of a physical disk to the DG/UX operating system with the **sysadm** utility. You can register the disk when you software format it. At startup, the operating system automatically registers each disk built into its kernel, unless the disk is already registered by another host.

soft mount Soft-mounted directories return an error code to programs after trying a mount request unsuccessfully for a system-specified time period. *See also hard mount.*

striping (disk) The arrangement of information in round-robin fashion on physical disks such that reads and writes can occur with multiple disk drives simultaneously and independently. By allowing multiple sets of read/write heads to work on the same task at once, disk striping can enhance performance. Striping requires that each virtual disk be identically sized. A striped virtual disk cannot be expanded or shrunk. Striping is available only with aggregated virtual disks. *See also virtual disk and aggregation.*

There are two kinds of disk striping: software and hardware. You can implement software striping with any group of disks; doing so is described in this manual. With a disk-array storage system, you can implement hardware disk striping by binding disk modules as a RAID group. Refer to a discussion of binding as described in the *CLARiiON™ Series 2000 Disk-Array Storage System with the DG/UX™ Operating System.*

sysadm utility A utility program supplied with the DG/UX operating system, **sysadm** runs as either an ASCII menu-driven or window-based program. There are two versions of sysadm, each available at a different level of system operation: stand-among **sysadm** at a run level of init 1 or higher and stand-alone version when the system is not yet operational. Information on using **sysadm** is in Appendix E. With **sysadm,** you perform system administration tasks such as software formatting disks, creating virtual disks, creating file systems, and building operating system kernels.

unmount To detach a file system from a directory, making it inaccessible to users. The system unmounts remote (ONC/NFS-mounted) file systems when you shut your system down to run level 2 or lower. The system unmounts local file systems (except **/usr** and /) when you shut the system down to run level S (single-user mode). *See also mount and /etc/fstab file.*

VDM Virtual disk manager. Often used interchangeably with OSM. A software technology allowing the access and manipulation of software virtual disks on physical disks while the data is on line and in use. Also, it offers a flexible hierarchical structure that allows grouping resource configurations into higher-level configurations at no performance or availability cost. *See also OSM, LDM and virtual disk.*

virtual disk OSM counterpart to the LDM logical disk, which is an amount of space that you reserve on a physical disk onto which data is stored. You can name a virtual disk using DG/UX file naming conventions. *See also aggregation, cache, mirror, partition and file system.*

volume A virtual disk's device filename in the device directory (form **/dev/dsk/***virtual-disk-name*). If the volume contains a file system, you can mount it.

End of Glossary

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a. MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   b. Send your order form with payment to:

      Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581–9973

   c. TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a. Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.

   b. Check or Money Order – Make payable to Data General Corporation. Credit Card – A minimum order of $20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
Over 200 Items	$100.00

   If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$0–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	**SHIP TO:** (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME _____	COMPANY NAME _____
ATTN: _____	ATTN: _____
ADDRESS _____	ADDRESS (NO PO BOXES) _____
CITY _____	CITY _____
STATE _____ ZIP _____	STATE _____ ZIP _____

Priority Code _____ (See label on back of catalog)

_____ _____ _____ _____ _____
Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS	ADD
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0–$149.99	0%
$150–$499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	–
SUB TOTAL	
Your local* sales tax	+
Shipping and handling – See A	+
TOTAL – See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is _____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐ Expiration Date ☐☐☐☐

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508–870–1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE

TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub–licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision–locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Managing Mass Storage Devices and DG/UX™ File Systems

093–701136–00

Cut here and insert in binder spine pocket