DataGeneral

Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Customizing the DG/UX™ System

093–701101–04

AViiON®

PRODUCT LINE

Customizing the DG/UX™ System

093–701101–04

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. 04, January 1994

Ordering No. 093–701101

Notice

Customizing the DG/UX™ System
093–701101–04

Revision History:	Effective with:
Original Release – June 1991	DG/UX Release 5.4
First Revision – February 1992	DG/UX Update 5.4.1
Second Revision – August 1992	DG/UX Release 5.4.2
Third Revision – February 1993	DG/UX 5.4 Release 2.01
Fourth Revision – January 1994	DG/UX 5.4 Release 3.00

A vertical bar in the margin of a page indicates substantive technical change from the previous revision.

About this manual

This manual explains some commonly performed system administration tasks. Using Table 1, next, or the index, you can identify tasks that apply to you. For example, after you install the DG/UX system (manual *Installing the DG / UX™ System*), this manual will help you create virtual disks other than your system disk, add user accounts and software packages, add OS clients, and build and boot DG/UX kernels.

IMPORTANT: Information on configuring printers has been moved from this manual to *Installing and Configuring Printers on the DG / UX™ System,* Ordering Number 093-701132.

Information on configuring modems has been moved from this manual to *Managing Modems and UUCP on the DG / UX™ System*, Ordering Number 069-000698.

The information in this manual is presented in cookbook form to help you perform tasks quickly and efficiently. For a conceptual description of system administration duties, see *Managing the DG / UX™ System.*

Although tasks are explained thoroughly, this manual assumes you have experience as a system administrator of an operating system. While detailed knowledge of UNIX® is not required, it is helpful to know

- The general file system layout of the UNIX operating system
- How to use UNIX commands
- How to use a shell and work within the UNIX directory structure

For background information on these topics, refer to *Using the DG / UX™ System.*

Finding descriptions of common tasks

Table 1 lists common customizing tasks and explains where you can find details on them.

Table 1 Finding Information on Starting and Running DG/UX

Task	Where to Find Details
Abnormal shutdown, handling	*Managing the DG/UX™ System*
Additional packages, installing	*Customizing the DG/UX™ System*, Chapter 6
Applications, user, executing	*Managing the DG/UX™ System*
Asynchronous lines, using	*Customizing the DG/UX™ System*, Chapter 5
Backing up files and disks	*Managing the DG/UX™ System*
Booting	See Startup
Clients, OS, planning and supporting	*Customizing the DG/UX™ System*, Chapters 2, 6, 7; *Managing the DG/UX™ System*
Converting logical disks to virtual disks	*Customizing the DG/UX™ System*, Chapter 3
Defining terms	*Customizing the DG/UX™ System*, Chapter 1
Devices, naming see also Disks, Tapes, Terminals, Asynchronous lines	*Customizing the DG/UX™ System*, Appendix B; see also Disks, Tapes, Terminals, Asynchronous lines in this table
Disk-array storage systems	
CLARiiON storage system	014-series *Disk-Array* system manual supplied with the storage system; *Managing the DG/UX™ System*
H.A.D.A. type 30-disk	*Installing and Configuring the High-Availability Disk-Array Subsystem*
Disks, virtual disks, logical disks	
adding drives	*Customizing the DG/UX™ System*, Chapters 9 and 10
checking (fsck)	*Managing the DG/UX™ System*
file systems	*Customizing the DG/UX™ System*, Chapter 3
virtual and logical disks	*Customizing the DG/UX™ System*, Chapters 1 and 2
system disk	*Installing the DG/UX™ System*
virtual disk	*Managing the the DG/UX™ System*; *Customizing the DG/UX™ System*, Chapters 1, 2, and 3
Errors, recovering from	*Managing the DG/UX™ System*
Failover, disk	014-series Disk-array storage system manual; *Managing the DG/UX™ System; Achieving High Availability with the DG/UX™ System.*
File systems	*Customizing the DG/UX™ System*, Chapter 3
High availability features	*The CLARiiON 2000 Series Storage System with the DG/UX™ Operating System*; *Achieving High Availability DG/UX™ System*; *Managing the DG/UX™ System*
Installing software	
DG/UX	*Installing the DG/UX™ System; Managing the DG/UX™ System*
Additional packages	*Customizing the DG/UX™ System*, Chapter 6
Kernel, creating	*Customizing the DG/UX™ System*, Chapter 10; *Managing the DG/UX™ System*
Log file, system	*Managing the DG/UX™ System*
Logical disks	See virtual disks.

Continued

Task	Where to Find Details
Login	*Customizing the DG/UX™ System*, Chapter 11; *Using the DG/UX System*
Management decisions	*Managing the DG/UX™ System*; *Customizing the DG/UX™ System*, Chapter 2
Memory management	*Analyzing DG/UX™ System Performance*
Modems	*Managing Modems and UUCP on the DG/UX™ System*
Networks	*Managing TCP/IP on the DG/UX™ System*
Performance	*Analyzing DG/UX™ System Performance*; *Customizing the DG/UX™ System*, Chapter 2; *Managing the DG/UX™ System*
Planning xxx	See the topic xxx.
Printers	*Installing and Configuring Printers on the DG/UX™ System*
Releases of DG/UX primary secondary	 *Installing the DG/UX™ System*; *Managing the DG/UX™ System* *Customizing the DG/UX™ System*, Chapter 2, 3, 7, 8
SCM system	*014–series SCM hardware manual*
Security features	*Managing the DG/UX™ System*; see also “Related documents” later in this preface.
Server, OS	*Customizing the DG/UX™ System*, Chapter 6; *Managing the DG/UX™ System*
Shell commands (operator)	*Using the DG/UX System*; *Customizing the DG/UX™ System*, Appendix A
Shutdown	*Customizing the DG/UX™ System*, Chapter 11
Startup (booting)	*Customizing the DG/UX™ System,* Chapter 11; *Installing the DG/UX™ System*; *Managing the DG/UX™ System*
Tape drives, adding	*Customizing the DG/UX™ System*, Chapters 9 and 10
Terminals	*Customizing the DG/UX™ System*, Chapter 5, Appendixes C and E
Terminology	*Customizing the DG/UX™ System*, Chapter 1
User accounts login	 *Customizing the DG/UX™ System*, Chapter 4; *Managing the DG/UX™ System* *Customizing the DG/UX™ System*, Chapter 11
Virtual disks	*Customizing the DG/UX™ System*, Chapters 2, 3, and 9*;* *Managing the DG/UX™ System*
X Terminals	*Customizing the DG/UX™ System*, Chapter 6*;* *Managing the DG/UX™ System*
Worksheets	*Installing the DG/UX™ System; Customizing the DG/UX™ System*, Chapters 2, 5, 6, Appendix D; 014-series disk-array system manual supplied with storage system

How this manual is organized

This section lists the parts of this manual and their content.

Chapter 1 **Preparing to customize the DG/UX system**
Defines terms, summarizes differences between this DG/UX release and the previous one, outlines the role of the system administrator, and introduces the System Administration utility (**sysadm**) used for customizing.

Chapter 2 **Planning your disk storage**
Helps you understand your physical disk resources and plan your virtual disks and file systems for DG/UX software and your applications.

Chapter 3 **Creating virtual disks and file systems**
Shows how to create the virtual disks and file systems you planned in Chapter 2.

Chapter 4 **Adding user accounts**
Provides instructions for creating accounts (home directories and login privileges) for each user.

Chapter 5 **Adding terminals**
Explains setting up terminals to operate in the DG/UX environment.

Chapter 6 **Loading and setting up additional packages**
Outlines the procedures of loading and setting up software packages other than DG/UX that you obtained from Data General or a third-party vendor.

Chapter 7 **Adding OS clients and X terminals**
Helps you link an OS server to OS clients (systems that rely on the server for operating system services and disk storage).

Chapter 8 **Adding secondary operating system releases**
Shows how to set up a second, different release of the DG/UX or non-Data General UNIX® operating system.

Chapter 9 **Adding mass storage devices**
Provides instructions for adding physical devices, such as Winchester, CD-ROM, magneto-optical, diskette, and tape drives.

Chapter 10 **Building kernels**
Explains when and how to build a new kernel (the basis of the DG/UX operating system) to reflect your hardware and software configuration.

Chapter 11 **Booting and logging in to the DG/UX system**
Provides different methods for booting the system following a system panic or failure. Also describes how to log in and shut down.

Appendix A **Basic DG/UX system concepts and commands**
For system administrators new to a UNIX system, provides an overview of the shell environment, directory structure, viewing and editing files, and manual pages.

Appendix B **Naming DG/UX I/O devices**
Explains the DG/UX device naming format used to identify standard and nonstandard devices.

Appendix C **Using Data General terminals on the DG/UX system**
Presents guidelines for making your Data General terminals operational in a DG/UX environment.

Appendix D **Worksheets**
Contains extra copies of the worksheets you completed in previous chapters.

Appendix E **Localizing your system**
Describes how to tailor your DG/UX system to a specific locale with a specific character set.

Related documents

Refer to the following documents for details on features presented in this manual. You can order any of these manuals from Data General by mail or telephone, as shown on the TIPS Order Form in the back of the manual.

- *Using AViiON® Diagnostics and the AV/Alert™ Diagnostic Support*, Ordering Number: 014-002183

  Explains how to start and run AViiON diagnostics to detect and test computer peripherals, and explains using the AV/Alert remote diagnostic features.

- *Learning the UNIX® Operating System*, Ordering Number: 069-701042

  Helps beginners learn UNIX fundamentals using step-by-step tutorials. Published by O'Reilly and Associates.

- *Using the DG/UX™ System,* Ordering Number: 069-701035

  Describes the DG/UX system and its major features, including the C and Bourne shells, common user commands, file system, and communications facilities such as **mailx**.

- *Using the DG / UX™ Editors*, Ordering Number: 069-701036

  Describes the text editors **vi** and **ed**, the batch editor **sed**, and the command line editor **editread**.

- *The Kornshell Command and Programming Language*, Ordering Number: 093-701105

  Provides a tutorial to the Kornshell command language and interface, and explains how to create scripts.

- *Installing the DG / UX™ System*, Ordering Number: 093-701087

  Describes how to install the DG/UX operating system on AViiON hardware.

- *Managing the DG / UX™ System*, Ordering Number: 093-701088

  Discusses the concepts and tasks related to DG/UX system management, providing general orientation to the administrator's job as well as instructions for managing disk resources, user profiles, files systems, tape drives, and other features of the system.

- *Installing and Configuring Printers on the DG / UX™ System,* Ordering Number: 093-701132

  Explains how to install, configure and manage printers.

- *Managing Modems and UUCP on the DG / UX™ System,* Ordering Number: 069-000698

  Describes setting up a modem that is Hayes SmartModem™ compatible.

- *Achieving High Availability on AViiON® Systems,* Ordering Number: 093–701133

  Describes the hardware and software components that go into highly available AViiON systems, and includes examples of how to set up and run these high-availability systems.

- *Analyzing DG / UX™ System Performance,* Ordering Number: 093–701129

  Tells how to analyze DG/UX system performance and fine-tune a system. Explains how the DG/UX system uses CPUs, virtual memory, file systems, and I/O devices.

- *X Window System™ User's Guide, OSF / Motif™ Edition,* Ordering Number: 069-100229

  Describes window concepts, the application programs (clients) commonly distributed with the X Window system, and how you can expect programs to operate with the OSF/Motif interface.

- *Managing ONC™ / NFS® and Its Facilities on the DG / UX™ System,* Ordering Number: 093-701049

  Explains how to manage and use the DG/UX ONC™/NFS® product. Contains information on the Network File System (NFS), the Network Information Service (NIS), Remote Procedure Calls (RPC), and External Data Representation (XDR).

- *Managing TCP / IP on the DG / UX™ System*, Ordering Number: 093-701051

  Explains how to prepare for the installation of Data General's TCP/IP (DG/UX) package on AViiON computer systems. Tells how to tailor the software for your site, and use **sysadm** to manage the package and troubleshoot system problems.

- *The CLARiiON™ 2000 Series Disk-Array Storage System with the DG / UX™ Operating System,* Ordering Number: 014-002168

  Explains the different storage system configurations, disk configurations, and routine operation of Data General's compact, high capacity, high availability disk-array storage system with the DG/UX system. This manual is packaged with the disk array hardware and licensed internal code.

- *Legato NetWorker User's Guide,* Ordering Number: 069-100496

  Explains how to use the NetWorker file backup software from a Networker client.

- *Legato NetWorker Administrator's Guide,* Ordering Number: 069-100495

  Explains the setup and maintenance procedures for the NetWorker backup software, including the NetWorker server, its clients, and backup devices.

- *Programmer's Guide: STREAMS*, Ordering Number: 093-701106

  Describes the STREAMS interface facility and how to use it. The STREAMS facility provides special queuing, messaging and buffering functions that simplify addition and deletion of modules of code. For information on how STREAMS works in the DG/UX system and descriptions of important kernel-level utility routines, see *Programming in the DG / UX™ Kernel Environment* (093-701083).

Trusted systems manuals

If you have a trusted version of the DG/UX system, you also have the following manuals.

- *Security Features User's Guide for the C2 Trusted DG / UX System*, Ordering Number: 093-701108 or,
  Security Features User's Guide for the B1 Trusted DG / UX System, Ordering Number: 093-701112

  For users, describes using the security features specific to C2 and B1 trusted systems, respectively.

- *Trusted Facility Manual for the C2 Trusted DG / UX System*, Ordering Number: 093-701110 or,
  Trusted Facility Manual for the B1 Trusted DG / UX System, Ordering Number: 093-7011154

  For administrators, describes using the security features specific to C2 and B1 trusted systems, respectively.

- *Audit System Administrator's Guide for the C2 Trusted DG / UX System*, Ordering Number: 093-701111 or,
  Audit System Administrator's Guide for the B1 Trusted DG / UX System, Ordering Number: 093-701115

  Describes using the audit features in C2 and B1 trusted systems, respectively.

Reader, please note

This manuals uses certain typefaces and symbols as follows.

Typeface/Symbol	Means
boldface	In command lines and command format lines: Indicates text and punctuation that you type verbatim from your keyboard.
`typewriter`	Represents a system response on your screen. Command format lines also use this font.
italic	In command format lines: Represents variables for which you supply values; for example, the names of your directories and files, your username and password, and possible arguments to commands.
[*optional*]	In command format lines: These brackets surround an optional argument. Don't type the brackets; they only set off what is optional. The brackets are in regular type and have a different meaning from the boldface brackets shown next.
[ ]	In command format lines: Indicates literal brackets that you should type. These brackets are in boldface type and should not be confused with the regular type brackets shown above.
...	The ellipsis means you can repeat the preceding argument as many times as desired.
$, % and #	The $ is the Bourne and Korn shell prompt; the % is the C shell prompt; and # is the superuser prompt for all three shells.
↵	Represents the New Line key. If your terminal keyboard has no New Line key, press the Enter or Return key.
< >	Angle brackets distinguish a command sequence or keystroke (such as **<Ctrl-D>**, **<Esc>**, and **<3dw>**) from surrounding text.
`xxx-> yyy-> zzz`	This indicates a series of menu selections. For example, `Device -> Disk -> Physical` means select Device, Disk, and then Physical.

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 1 – Preparing to customize the DG/UX system

Chapter 2 – Planning your disk storage

Chapter 3 – Creating virtual disks and file systems

Chapter 4 – Adding user accounts

Chapter 5 – Adding terminals

Chapter 6 – Loading and setting up additional packages

Chapter 7 – Adding OS clients and X terminal clients

Chapter 8 – Adding a secondary operating system release

Chapter 9 – Adding mass-storage devices

Chapter 10 – Building a DG/UX kernel

Chapter 11 – Booting and logging in the DG/UX system

Appendix A – Basic DG/UX system concepts and commands

Appendix B – DG/UX I/O device names

Appendix C – Using Data General terminals on the DG/UX system

Appendix E – Localizing your DG/UX system

Tables

Table

Figures

Figure

Figure

1 Preparing to customize the DG/UX system

This chapter defines some terms, summarizes the differences between this DG/UX release and the previous one, defines some system administrator tasks, and explains running the **sysadm** utility. Major sections proceed as follows.

- Terms and concepts
- Changes with DG/UX 5.4 Release 3.00
- The role of system administrator
- Using the **sysadm** utility
- Where to go next

DG/UX 5.4 Release 3.00 introduces the new virtual disk management (VDM) technology, which lets system administrators to manipulate disk storage space with the data on-line and in use. 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 moving partitions from one physical disk to another and creating or removing software mirrors. 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.

If you already have DG/UX 5.4 Release 3.00 up and running and you are comfortable with your knowledge of its terms and features, skip this chapter and go directly to the chapter you need. The Preface, “About this manual,” includes a task-oriented table that shows where you can find information on common DG/UX administrative tasks. The Preface also lists the chapters and appendixes and their content.

For background information on basic DG/UX system concepts and commands, see the manuals named in the Preface or Appendix A.

Terms and concepts

To understand this manual and customize your DG/UX system effectively, you need to understand the following terms, many of which are new with DG/UX 5.4 Release 3.00.

aggregation — Combination of virtual disks of any type to form a higher level virtual disk whose size is the sum of its constituent disks. An aggregation is the OSM equivalent of the LDM multipiece logical disk.

compatibility mode

The way a DG/UX system running virtual disks lets you access a logical disk. When you upgrade from a release earlier than DG/UX 5.4 Release 3.00, **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 lets you read from and write to the logical disk (if it is writable) without applying any new DG/UX 5.4 Release 3.00 features to it.

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 whose capacity is measured in megabytes (Mbytes) or gigabytes (Gbytes), and is identified by a model number and a DG/UX device name. Disk drive types include 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 hard disk and CD-ROM drives are bootable.

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.

file system

A directory structure located on a virtual disk. The file system contains information that the operating system requires to keep track of files and directories on the disk. Using **sysadm**, you create a file system on a virtual disk; then you add the file system, which makes it available to the DG/UX system and appends its name to the

/etc/fstab file, allowing it to be mounted automatically at future system startups. See also mount point.

LDM (logical disk management)

The predecessor to OSM (on-line storage management). LDM uses logical disks instead of OSM's virtual disks.

logical disk

A span of disk space that you software format via **sysadm** so the DG/UX system can use it: it is analogous to the DG/UX 5.4 Release 3.00 virtual disk.

You can use a logical disk with a DG/UX system that uses virtual disks, but only in limited ways (compatibility mode).

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, in the disk-array storage system disk drive device name **sd(dgsc(0),0,3)**, the 3 is the LUN. In this manual, the term disk drive means the same thing as physical disk and logical unit.

mirroring

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. The 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; doing so is 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. Binding is described in the disk-array storage system manual named in the Preface, "About this manual."

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**.

OSM (on-line storage management)

A software technology that lets you access and modify software virtual disks on physical disks that are on line and in use. OSM is not available with DG/UX releases before 5.4 Release 3.00.

partition

The DG/UX 5.4 Release 3.00 analog to logical disk piece: a contiguous span of disk space that you allocate on a virtual disk using **sysadm**. A disk can have one or more partitions.

physical disk

The magnetic media in a disk drive, or for a disk-array storage system, one or more disk drives bound together. Using **sysadm**, you format a physical disk and create one or more virtual disks (DG/UX 5.4 Release 3.00 and after) or logical disks (before DG/UX 5.4 Release 3.00) on it.

primary release

The release of the DG/UX system that runs routinely. On a server system, you can also install one or more secondary releases for client systems that need a different release.

RAID

Redundant array of inexpensive disks. A technology that groups individual disks into one drive 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.

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.

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.

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. Binding is described in the disk-array storage system manual named in the Preface, "About this manual."

secondary release — See primary release.

sysadm utility — A utility program supplied with the DG/UX operating system. **Sysadm** runs as either a menu-driven or window-based program, as explained later in this chapter. There is a version you can run from DG/UX and a stand-alone version (path **/usr/stand/sysadm**). **Sysadm** lets you perform system administration tasks such as software formatting disks, creating virtual disks, creating file systems, and building operating system kernels.

virtual disk — A span of disk space that you software format using **sysadm** so the DG/UX system can use it. (This is the DG/UX 5.4 Release 3.00 analog to the logical disk.) A virtual disk can occupy part of or all of one or more physical disks. A virtual disk can be of one of the following types: aggregation, cache, mirror, partition. You can create a file system on each virtual disk. You can name a virtual disk using DG/UX file naming conventions.

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.

Changes with DG/UX 5.4 Release 3.00

This section summarizes the major user visible changes between DG/UX 5.4 Releases 2.10/2.01 and 3.00. (2.10 and 2.01 work virtually the same way; they differ in that 2.10 is designed for specific types of AViiON computers.) This section may interest you if you are running an earlier release and contemplate installing the later one.

1. The *virtual disk*, in name, concept, and disk format, replaces the *logical disk*. Virtual disks use a different disk format from logical disks. Essentially they are a more flexible type of logical disk. You can change the size and configuration of a virtual disk while the disk is mounted and in use, which you cannot do with a logical disk.

   For most disks, upgrading from the logical disk format to the virtual disk format is easy. When you upgrade from a release earlier than DG/UX 5.4 Release 3.00, **sysadm** offers to convert all your physical disks to virtual disk format. If you tell **sysadm** not to convert a disk, the DG/UX system will use it in compatibility mode; this lets you read from and write to the physical disk (if it is writable) without using any new DG/UX 5.4 Release 3.00 features with it.

IMPORTANT: When you install DG/UX 5.4 Release 3.00, you convert your system disk to virtual disk (VDM) format. After a physical disk is converted to virtual disk format, you can revert to logical disk format easily provided you have not created any special virtual disk structures on the physical disk. To convert a physical disk that has virtual disk structures or that was created as a virtual disk, you must dump its files (if any), reformat in the logical disk format, and then reload the files.

2. DG/UX 5.4 Release 3.00 includes a new stand-alone **sysadm** utility in addition to the familiar stand-among **sysadm**. The stand-alone **sysadm** pathname is **/usr/stand/sysadm**; there are shell commands you can issue directly to this new **sysadm** for disk repair, among other tasks. The two **sysadm**s include all functions of the **diskman** utility; Data General no longer ships **diskman** with the DG/UX software.

3. DG/UX 5.4 Release 3.00 now supports systems that have two VME buses (channels). To distinguish the parent channel VME channel of a VME controller from other channels, DG/UX allows a channel name **(vme(0)** or **vme(1)**) as the optional first parameter in the device name. For example, **syac(vme(1),4)** refers to the fifth standard **syac** device attached to the second VME channel controller. All the following DG/UX VME devices allow (but do not require) the parent VME to be specified as their first parameter.

cied **cimd** **cien** **cisc** **dgsc**
hada **hken** **nvrd** **pefn** **ssid**
syac **vitr** **vsxb**

For compatibility with systems that do not use multiple VME channels, we made the **vme()** device name and trailing comma optional; if you omit it, the system will assume "**vme(0),**". Therefore, the device name **sd(dgsc(vme(0),2),3,1)** is functionally the same as **sd(dgsc(2),3,1)**. The DG/UX device sizer **probedev** automatically generates entries in the system configuration file for VME channels.

For the most recent information on differences between releases, see the product release notice (085-series) supplied with the software.

The role of system administrator

The system administrator is responsible for the normal operation of a DG/UX system configuration. Typical duties include

- Monitoring a successful system boot (startup)
- Customizing the system for new hardware and software
- Backing up the system
- Tuning the system for improved efficiency
- Verifying file security
- Checking file system size

Customizing tasks occur over the lifetime of a system, from installation to any time you redefine some aspect of the configuration. For a system to reach full productivity, you may add virtual disks and file systems, user accounts, operating system clients (OS clients), I/O devices, additional software packages, and build new kernels. You gain access to the System Administration (**sysadm**) utility by logging in as **sysadm**, and then running **sysadm**. Using **sysadm**, you customize the DG/UX system environment to suit your particular needs. For more details on an administrator's duties, see *Managing the DG / UX™ System.*

If your configuration includes OS clients, the OS server administrator executes **sysadm** to build OS client first-time kernels, after which the administrator of each OS client must boot the kernel at the computer used as an OS client. OS client administrators can then use **sysadm** for continued customization.

If your configuration requires a network connection, NIS file service, or X Window System™ service, you will have to contact administrators of those facilities to complete customization.

The DG/UX system restricts certain administrative commands and operations to the superuser. As superuser, you can also override and change any file permissions on the system. The DG/UX system has two superuser logins by default: **root** and **sysadm**.

You should use the **sysadm** rather than the **root** profile when performing administrative tasks because, unlike the **root** profile, the **sysadm** profile has the **/admin** directory as its home directory. The **admin** directory is a safe place for you to create files as superuser. The root (/) directory, which is the home directory of the **root** profile, is not a good place to create such files.

To become superuser, you may either log in as the superuser or, if already logged in as a normal user, you may use the **su** command. Either way, you must know the password of one of the superuser profiles. To become **sysadm** after login, enter

```
% su sysadm ↲
```

The **su** command prompts for the **sysadm** password. If you have not assigned passwords to the **sysadm** and **root** profiles, do so now otherwise, any user may log in as the superuser. You can assign passwords to these profiles with the **passwd** command; for example,

```
# passwd root ↲
  or
# passwd sysadm ↲
```

Using sysadm

You will use the **sysadm** utility, which offers a menu-based interface, to administer your system. Three interfaces are available:

- Graphical
- ASCII terminal menu
- Shell command line

The graphical interface is available only if the X Window System package is installed and you are using a graphical computer system. Users of graphical computer systems and ASCII terminals can also access both the ASCII terminal menu interface and the shell command line. You may choose to use the shell if you are already familiar with **sysadm** and wish to bypass it.

Using the graphical interface

Regardless of the interface you're using, you can simply type **sysadm** at the superuser prompt (#) and the appropriate interface is invoked automatically. Or you can type **xsysadm** for the graphical interface. Figure 1–1 shows the top level **sysadm** menu as it appears in the graphical interface.

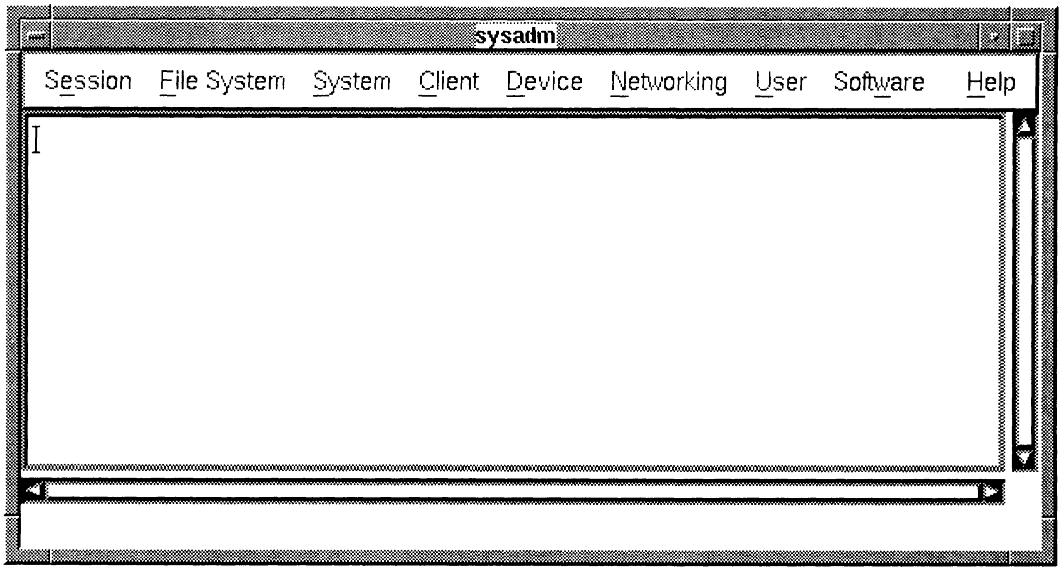

sysadm
Session File System System Client Device Networking User Software Help

Figure 1–1 Sysadm Main Menu in the Graphical Interface

To navigate the graphical interface using the mouse, move the pointer to the desired menu and select the menu by clicking the mouse's select button (typically the left button). Continue this way, selecting the desired menus until you reach an operation you want to perform. With the mouse, select the operation to start it. Optionally, you may drag the mouse across menus: press and hold the select button while moving the mouse across the desired selections. Releasing the select button when an operation is highlighted begins the operation.

After you select your option, **sysadm** presents a *form* when it needs more information to complete an operation. A form contains prompts that you may answer in any order. Some prompts appear next to a small square, a button that you click to alternate between "yes" (on and shaded dark) and "no" (off and shaded light). Some prompts require text input. To type text, first click on the desired box or press the Tab key to cycle through the boxes until you get the desired one. When the box is highlighted, you can type text. Do not press Enter to advance to the next box; instead, move the cursor to the desired box and click or press the Tab key. Pressing Enter has the same effect as using the **Next** or **OK** button at the bottom of the form to conclude the entire form. Pressing Enter prematurely executes the entire form before you have completed your text entry.

Some prompts let you select one or more values from a list. Click on the desired entry. If the list is too long to display completely on the screen, the operation displays the first part of the list in a box that has a scroll bar along the right side. To scroll through the list, use your mouse to drag the scroll bar up and down. When you see the value that you want in the list, click on it. For some prompts, you enter a numerical value by typing a response in a box or by using your mouse to drag an indicator along a horizontal scale.

After answering the form's prompts, proceed by pressing Enter or by selecting **OK** or **Next**. If you choose not to perform the operation, select **Cancel**. Select **Reset** to restore the default responses.

Sysadm provides on-line help in several ways. Clicking at the main menu, you see help on four subjects: **sysadm**, the interface, the **sysadm** version, and other help options.

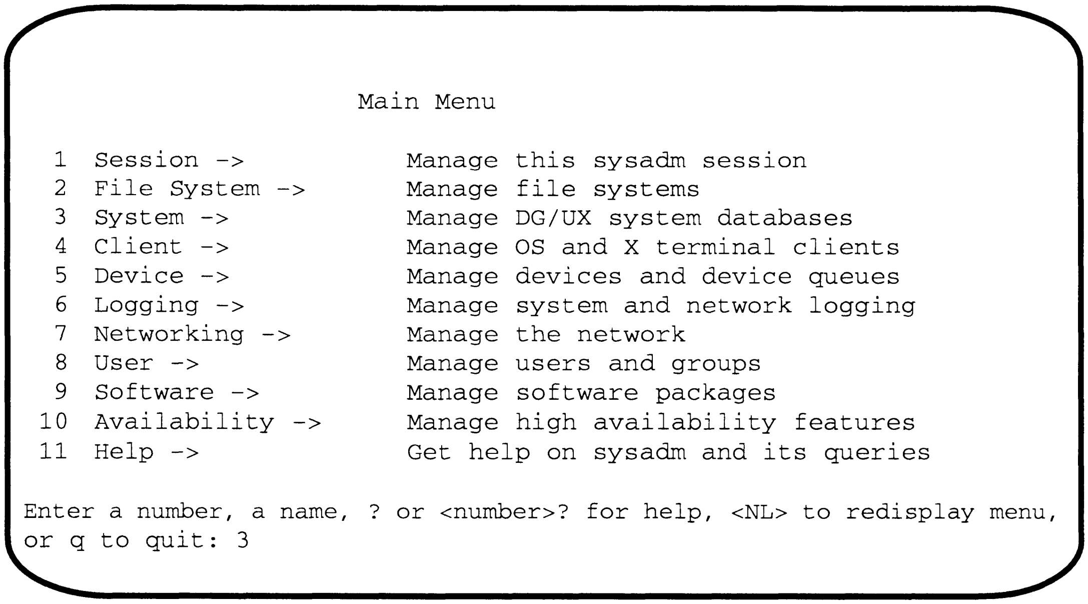
In addition, each **sysadm** menu and operation has a corresponding help message. Click on the desired menu choice or operation (a dark border surrounds it) and then press F1 (function key 1) to view the help message. The help message itself is easily recognized by the appearance of an "i" (for "information") to the left of the message.

To get help about an operation already in progress, click on **Help** at the bottom right of its form.

To get help about a specific prompt within a form, click on the prompt or press the Tab key to cycle through the prompts until you get the desired one. Then press F1 for help.

Using the ASCII terminal interface

Type **sysadm** at the superuser prompt (#) to invoke the appropriate interface. Or enter **asysadm** for the ASCII terminal interface. Figure 1–2 shows **sysadm** Main Menu as it appears in the ASCII interface.



```
                       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: 3
```

Figure 1–2 Sysadm Main Menu in the ASCII Interface

To navigate the ASCII terminal interface, type the desired menu selection number, and then press the Enter key. (Always end your entry by pressing Enter.) You may select your choice by typing its name (or as many letters as are necessary to make your selection unique). To return to the preceding menu, press the caret key (**^**). You may exit **sysadm** by pressing the **q** key (you are asked to confirm the request).

Table 1–1 presents a summary of the methods for making ASCII **sysadm** menu choices.

Table 1–1 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.

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. Examples of destructive actions include deleting an OS client or a release area.

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.

Using the shell command line to bypass menus

You can bypass the menu interfaces altogether by invoking **sysadm** directly from the command line, supplying the menu selections in this form:

sysadm -m *menu-name(s)*

where **-m** selects the menu name.

You specify menu names using colon-separated lists of strings that are not case-sensitive (see Table 1–1 for definitions). For example,

```
# sysadm -m file:local ↵
# sysadm -m f:l ↵
```

Both commands perform the same operation: they start **sysadm** at the Local Filesystem menu. The second example shows the operation names abbreviated to the shortest unique strings.

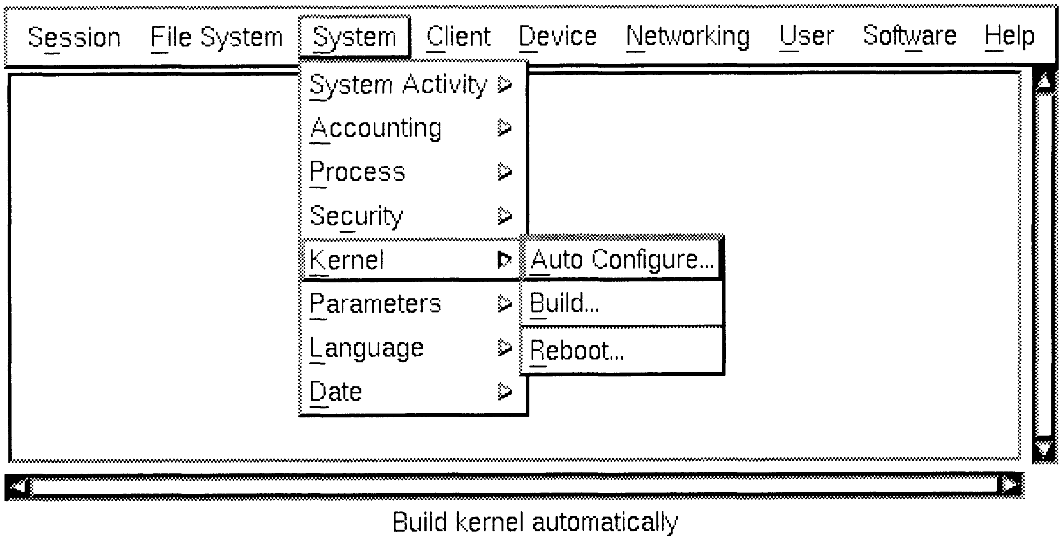
Conventions used in sysadm menus

This manual uses a generic pathway you follow through the **sysadm** menus, regardless of the interface you use. For example, to build an automatically configured kernel, start at the **sysadm** Main Menu and follow this path:

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

The names `System` and `Kernel` appear on the menu. The arrow (->) indicates the next menu choice.

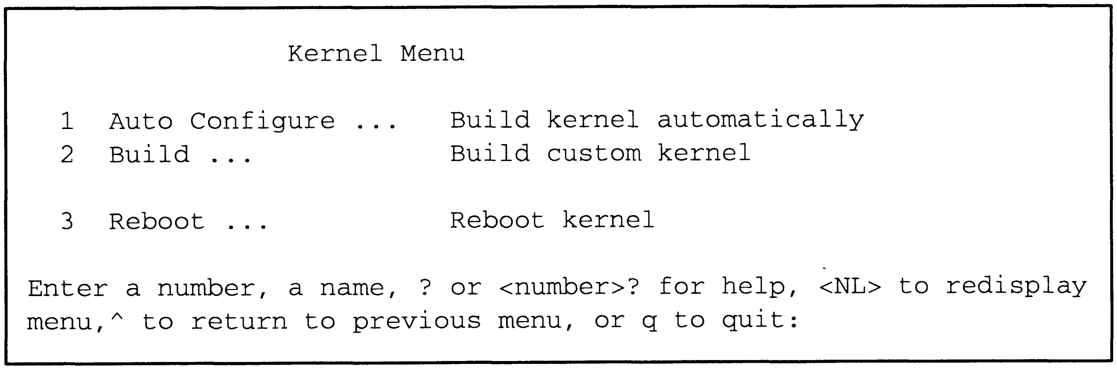
Figure 1–3 shows the Kernel Menu in the graphical interface.



Build kernel automatically

Figure 1–3 Graphical Kernel Menu

Figure 1–4 shows the Kernel Menu as it appears in the ASCII terminal menu interface.



```
                    Kernel Menu

   1  Auto Configure ...     Build kernel automatically
   2  Build ...              Build custom kernel

   3  Reboot ...             Reboot kernel

Enter a number, a name, ? or <number>? for help, <NL> to redisplay
menu,^ to return to previous menu, or q to quit:
```

Figure 1–4 ASCII Interface Kernel Menu

Using the graphical interface to select the Auto Configure operation from the Kernel Menu, click on these choices:

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

Using the ASCII terminal menu interface to select the Auto Configure operation from the Kernel Menu, supply an entry to the following prompt:

```
Enter a number, a name, ? or <number>? for help, or
q to quit: System:Kernel:Auto ↵
```

Alternatively, you could choose to enter the desired string or number at each menu level.

```
# sysadm -m System:Kernel:Auto ↵
```

For each of the three menus, at the conclusion of the menu traversal, you answer prompts to perform the desired operation.

Returning to the shell

During a **sysadm** session, you can easily return 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 ↵
```

Where to go next

Generally, we suggest you continue to Chapter 2, “Planning your disk storage.” This chapter explains gathering information that will be useful as you customize your system.

If you already have DG/UX up and running and know what you want to do, skip directly to the chapter you need.

The Preface, “About this manual,” includes a task-oriented table that shows where you can find information on common DG/UX administrative tasks.

End of Chapter

2 Planning your disk storage

This chapter helps you plan your disk storage. Proper planning can prevent a lot of time-consuming rebuilding.

Major sections proceed as follows.

- Assessing your storage needs: software packages, work areas, and data
- About the planning worksheets
- About virtual disks
- Planning virtual disks and file systems
- Assessing disk capacity
- Mapping virtual disks to disk drives
- Deciding where to mount file systems
- Completing the virtual disk planning worksheets
- Where to go next

Assessing your storage needs

Ask yourself these questions:

- What are my software and work area size requirements?
- Do they require virtual disks and corresponding file systems?
- Assuming virtual disks and file systems are needed, how much space does each software package or work area require?
- What do I name the virtual disks?
- What disk drives do I have and how much space do they provide?
- How do I map virtual disks to disk drives? That is, do I want to create contiguous virtual disks (partitions), aggregations of virtual disks, software mirror virtual disks, software striped virtual disks, cache virtual disks, or combinations of the above?
- Where should I mount the file systems in the DG/UX directory structure?

About the planning worksheets

The planning worksheets contain tables that you complete to identify the names of virtual disks, sizes, possible types, and mount points in the DG/UX directory structure. Time you spend here will speed up the customizing process. The planning worksheets are shown in Figures 2–5 through 2–9 (near the end of this chapter).

If you know how to customize the DG/UX system or another UNIX operating system, you may prefer to skip directly to the worksheets in Figures 2–5 through 2–9.

About virtual disks

This section explains when to create virtual disks, file system overhead, virtual disk names, and viewing virtual disk layout.

When to create virtual disks

The most common type of disk drive is a hard disk on which you create one or more virtual disks to contain file systems. If you have non-volatile RAM (NVRAM) memory that you want to use as a software cache, you must create a virtual disk on it. However, you do not need to create a virtual disk for other devices. Some devices on which you would not create virtual disks are

- a CD-ROM/magneto-optical disk
- a diskette
- a memory file system (a diskless file system reserved for file storage in the computer's main memory)

You can add file systems for these devices as explained in Chapter 3.

File system overhead

To accommodate a file system that is readable and writable, if you want a comfortable cushion for the superuser to use for reading and writing, add 10 percent to the virtual disk's size as overhead. If you don't want a cushion for the superuser, add nothing for 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. You might add a 10 percent cushion to derive its 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 (like those for swap areas and some database management systems) do not require a file system, in which case there is no overhead for the superuser. Read your software application's release notice for details.

Naming virtual disks

Virtual disk names are filenames of as many as 31 characters, including alphabetic characters, numbers, period (.), hyphen (-), and underscore (_).

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. (The mount point identifies a location in the DG/UX file system for placing the virtual disk's file system.) For example, the virtual disk name **usr_opt_X11** implies the mount point for its contents, **/usr/opt/X11**.

Viewing a disk drive's layout

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

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

and then 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 2–1 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 2–1 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 disk blocks, and total number of free disk blocks. A disk block is 512 bytes.

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; and
- size in disk blocks.

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

Planning virtual disks and file systems

If you have just installed the DG/UX system, your next task is to decide what virtual disks to create. If your system will have OS clients, you must create virtual disks for them and decide whether to install secondary releases. Select the applicable virtual disk types from the following list.

- OS client space (if you have OS clients)
- Multiple DG/UX release areas (if you have OS clients)
- Local directories

  — User home directories

  — Software packages

  — Work directories

  — Tools directories

  — Temporary space

  — Extra swap space

The following sections help you determine how to use virtual disks.

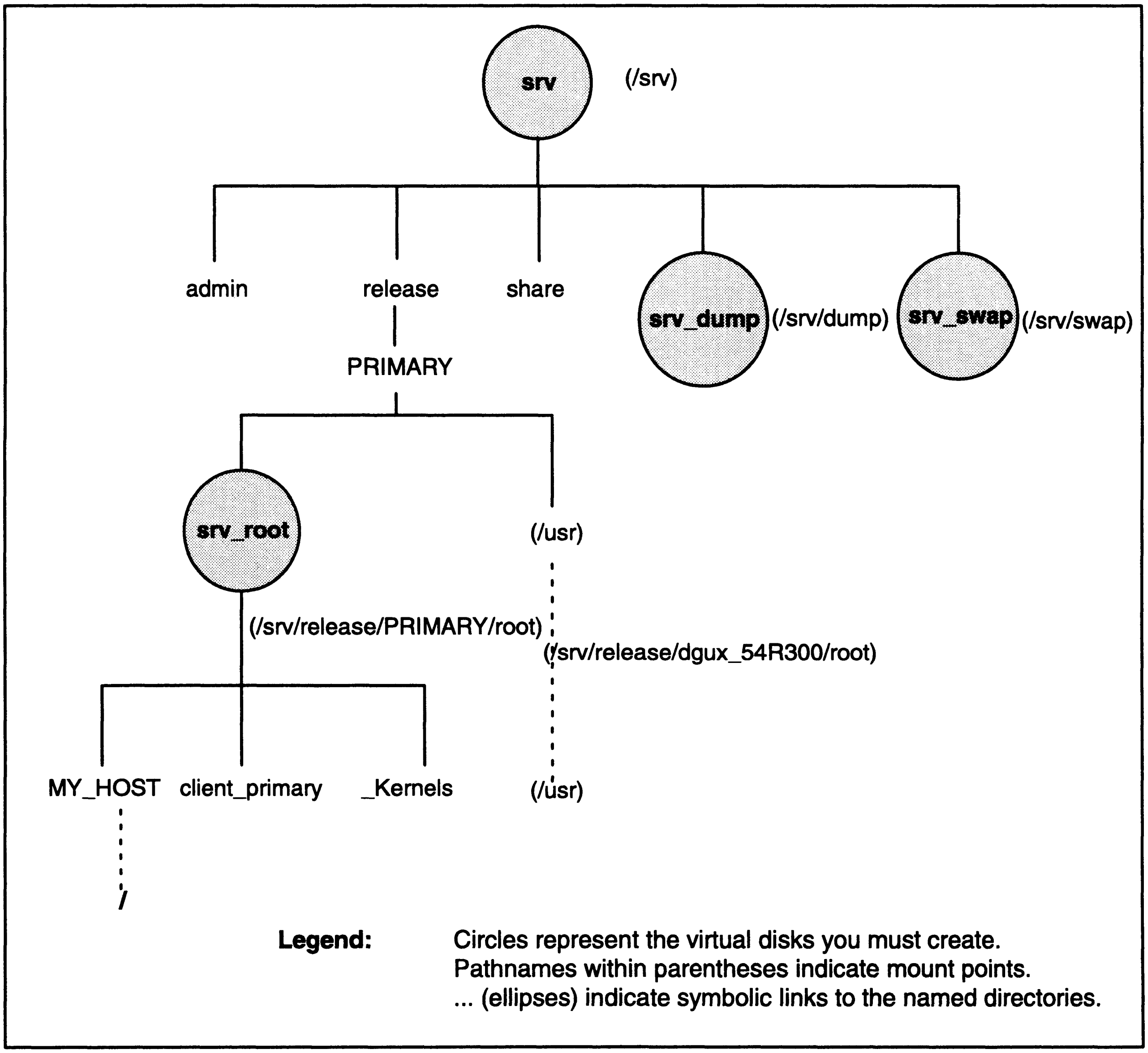
IMPORTANT: For any physical disks in a disk-array storage system that you intend to use for failover (defined in Chapter 1, section "Terms and Concepts"), see the 014-series manual supplied with the storage system or see *Managing the DG / UX™ System.*

OS client space

If your configuration includes diskless OS clients, you must create the following virtual disks on the OS server:

- OS client directory (**srv**)
- OS client root space (**srv_root**)
- OS client swap space (**srv_swap**)
- OS client dump space (**srv_dump**)

Figure 2–2 shows the DG/UX primary release area, **/srv/release/PRIMARY**. In the figure, shaded circles represent the virtual disks you must create mentioned above.



Legend: Circles represent the virtual disks you must create.
Pathnames within parentheses indicate mount points.
... (ellipses) indicate symbolic links to the named directories.

Figure 2–2 /srv File System

OS client directory (srv)

The virtual disk for the OS clients' directory holds OS client file systems, such as root directories and swap space. A typical name for this virtual disk is **srv**. It is typically mounted at /**srv**.

The **srv** virtual disk should be large enough to accommodate the **sysadm** database. You should create a single virtual disk named **srv** with a size of 5,000 blocks. You do not need to add additional space for this virtual disk.

OS client root space (srv_root)

The OS client root space is a single virtual disk that contains all the root directories for all clients. The pathname for this virtual disk is **/srv/release/PRIMARY/root**. The root directory contains subdirectories that correspond to each OS client.

The size of the DG/UX system's default root file system depends on whether OS clients have individual kernels or share a common kernel with other OS clients.

Root space for OS clients with individual kernels

For the DG/UX system's default root file system, each OS client with a bootable kernel needs about 40,000 blocks. To calculate the size of the virtual disk, multiply the number of OS clients by 40,000. For example, five OS clients need 200,000 blocks. You do not need to add additional space.

Root space for OS clients sharing a common kernel

When you build a kernel for an OS client, **sysadm** lets you link all OS clients to the same kernel image, thus saving disk space. You save approximately 6,000 blocks, the size of a kernel, for the second and each subsequent OS client that shares a kernel. But sharing kernels in this way can result in weakened security because any superuser can access and change the kernel image. If you decide to make such links anyway, remember that for OS clients to share a kernel, their root directories (and the directory containing the kernel) must all be on the same virtual disk. Thus, you should not distribute OS client root directories over different virtual disks.

For each client linked to a common kernel, use the following formula to calculate OS client root size.

(*number-of-OS-clients* *
(*recommended-OS-client-root-size* – *kernel-size*)) + *kernel-size*

For example, for five clients and a kernel size of 6,000 blocks, the numbers would be

```
(5 * (40,000 - 6,000)) + 6,000 =
(5 * 34,000) + 6,000 = 176,000
```

Linking the five OS clients to a common kernel saves 24,000 blocks.

Do not add any additional space as overhead.

OS client swap space (srv_swap)

Swap space is the temporary storage location of an active page from a process on a virtual disk. A page is stored (or paged) in swap space when there are more active processes than can simultaneously fit into the computer's main memory. When memory resources become available, the temporarily suspended page is sent back into main memory for execution.

The amount of swap space that you need depends on the amount of physical memory in your computer, the nature and number of the applications you run, and the number of users on the operating system. If your programs allocate large portions of memory, you may need more swap area. Insufficient swap area can result in the termination of running processes and error messages such as `From System: out of paging area space` at the system console. If you encounter such an error, you need to create more swap space or reduce your system load. You can also create multiple swap virtual disks to supplement existing swap virtual space.

Like the OS server, OS clients (including OS clients of a secondary release) require swap space. Unlike the OS server, an OS client's swap space is provided by an actual file on the OS server. These files are typically mounted at **/srv/swap**, whose virtual disk usually is called **srv_swap**.

Use the following formula and example to compute the initial swap space for OS clients. The number of *blocks-per-OS-client* is either 50,000 (24 Mbytes) or 1.5 times physical memory, whichever is larger.

(*number-of-OS-clients* * *blocks-per-OS-client)*

For example, for four clients, the numbers would be

```
4 * 50,000 = 200,000 blocks
```

The example above is a rough estimate. If your swap space is not sufficient once your system is running, use the free swap value reported by the **sar** command to determine the available swap space. (Divide the amount of free swap space by 2048 to convert blocks to megabytes.) You should maintain the amount of free swap space at 15% to 30% of the total physical memory plus swap area space on the system. Numbers for a sample system follow.

```
Physical memory:   256 Mbytes
Swap space:       +384 Mbytes
                   -------
Total:             640 Mbytes

15% of 640 Mbytes =  96 Mbytes, 196608 blocks
30% of 640 Mbytes = 192 Mbytes, 393216 blocks
```

For the sample system above, if free swap space is typically under 200,000 blocks or sometimes under 100,000 blocks, you should increase the amount of swap space. Conversely, if free swap space is typically over 400,000 blocks and rarely below 300,000 blocks, you can decrease the amount of swap space. Systems with surges in swap usage (variance over time greater than 30%) need a larger reserve than 15% to 30%. Systems with static swap usage (variance of less than 10%) need less reserve.

On the DG/UX system, you are not required to have as much swap space as physical memory. As long as the free swap space values stay within the above recommendations, you can reduce swap space to a fraction of physical memory.

OS client dump space (srv_dump)

You must allocate space for OS client system dumps. When a system panics or hangs, you write the contents of the system's memory to a file or tape so that Data General engineers can diagnose the problem.

On a system with a tape drive, you configure your kernel so that it dumps to tape or a local virtual disk. On an OS client without a tape drive, you configure the system so it dumps over the network to a file on a disk drive attached to the OS server. If you don't create a **srv_dump** virtual disk, ensure that **/srv** is sufficiently large to accommodate a dump.

The maximum amount of space you need for **/srv/dump** is equal to the total size of physical memory on all of your OS client systems (including OS clients of secondary releases). In practice, however, you need less space because all OS clients do not need to make system dumps simultaneously. Furthermore, you do not keep system dump files on line for very long. Space for one or two system dumps is probably sufficient.

If each OS client has 16 Mbytes (16,777,216 bytes or 32,768 blocks) of physical memory, approximately 33,000 blocks (plus overhead) is sufficient to hold one system dump. The formula for calculating client dump space follows.

OS-client-physical-memory (in blocks) + *overhead*

For example, for 16 Mbytes (33,000 blocks):

```
33,000 + 10% = 36,300 blocks
```

Multiple DG/UX release areas

Read this section only if you have OS clients in your configuration that want to run a UNIX system other than DG/UX or want to run a DG/UX release other than 5.4 Release 3.00. If all your clients will be happy to run DG/UX 5.4 Release 3.00, or if you don't have any clients, skip this section.

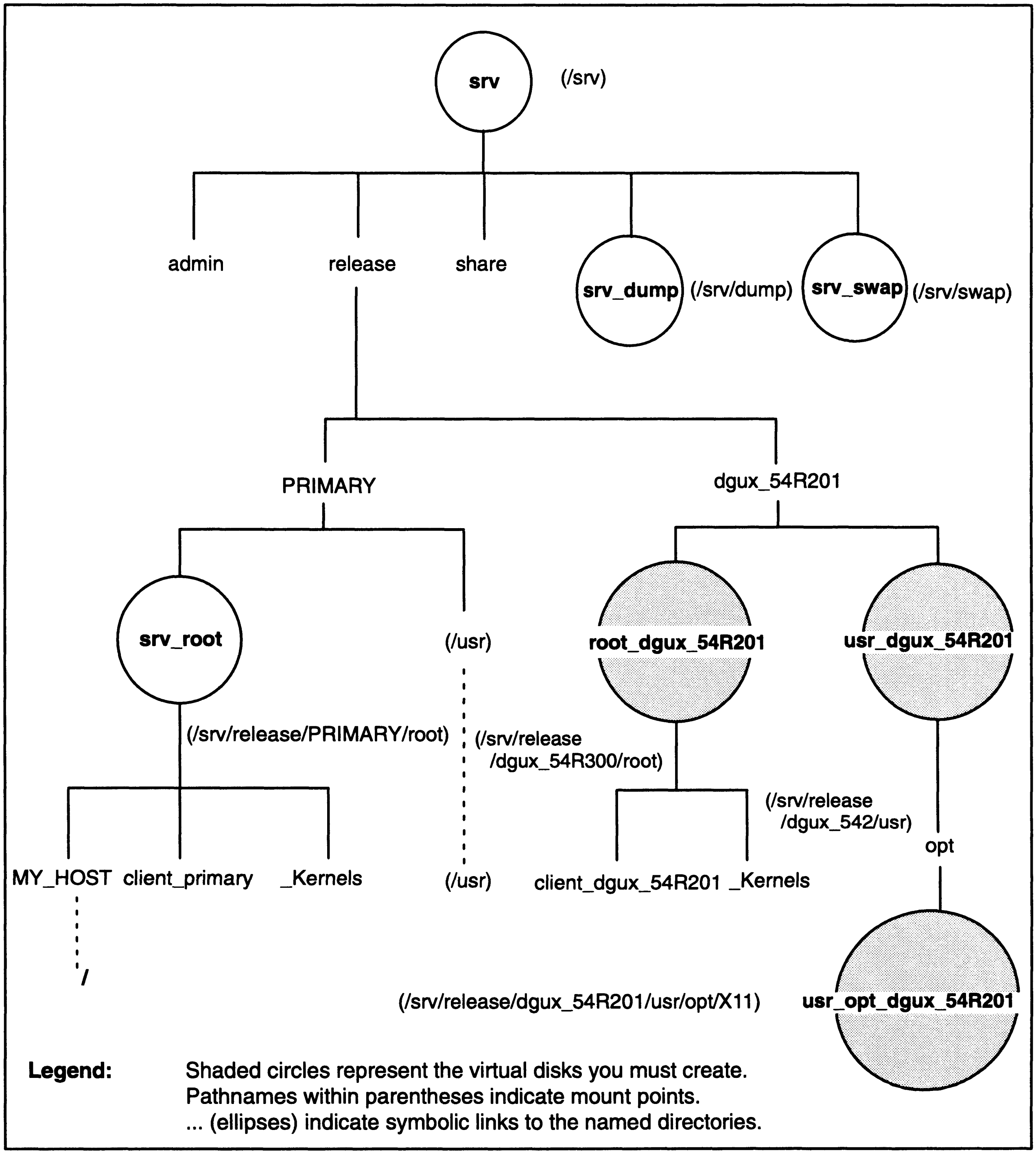
Generally, OS clients run the same operating system (here, assumed to be DG/UX 5.4 Release 3.00) as the server. That operating system is available to OS clients from directory **/srv/release/PRIMARY**. But DG/UX lets you install other releases of an operating system, such as an earlier versions of the DG/UX system or other operating systems such as SunOS, for OS clients. You install each other release in a directory structure called a secondary release area. Each secondary release area directory pathname has the form **/srv/release/***release-name*, where *release-name* is the name for the OS release; for example, **/srv/release/dgux_54R201** and **srv/release/SunOS**. The number of releases an OS server can support is restricted only by available disk drive resources and desired system performance.

The number of virtual disks you must create for a secondary release area depends on the OS release you are installing. To create a secondary OS release area for an operating system other than the DG/UX system, consult the documentation and release notice that accompanies that operating system. To create a secondary release area for DG/UX, you will create the following virtual disks:

- OS client root in a secondary release area (for 5.4 Release 2.01, virtual disk name **root_dgux_54R201**).
- If the OS client that will use the secondary release needs a usr disk, **usr** space in a secondary release area (for 5.4 Release 2.01, virtual disk name **usr_dgux_54R201**).
- If the OS client that will use the secondary release needs X11, X11 space in a secondary release area (for 5.4 Release 2.01, virtual disk **usr_opt_X11_dgux_54R201**).

Notice that you do not have to create a dump area or a swap space. Those resources are available from the **/srv** directory structure (for example, **srv_dump** and **srv_swap**).

A difference between the **/srv/release/PRIMARY** and the **/srv/release/***secondary-release* directory structures is that the latter requires a virtual disk for the **/srv/release/***secondary-release***/usr**, and **/srv/release/***secondary-release***/usr/opt/X11** file systems. Figure 2–3 shows a primary release area (**/srv/release/PRIMARY**) and a secondary release area (all files within **/srv/release/dgux_54R201**). In the figure, circles represent virtual disks and shaded circles represent the virtual disks mentioned above.



Legend: Shaded circles represent the virtual disks you must create.
Pathnames within parentheses indicate mount points.
... (ellipses) indicate symbolic links to the named directories.

Figure 2–3 Primary and Secondary Release File Structure

OS client root space (root_dgux_54201)

The OS client root space is a single virtual disk that contains all the root directories for all OS clients. You can mount this virtual disk wherever you want; an example of a directory mount point shown in Figure 2–3 is **/srv/release/dgux_54R300/root**. The root directory contains subdirectories that correspond to each OS client. Instructions are provided in Chapter 7 for adding an OS client.

For the DG/UX system's default root file system, each OS client needs the same amount of space as the OS server: 40,000 blocks. To calculate the size of the virtual disk, multiply the number of OS clients by 40,000. Do not add any additional space as overhead.

When you build a kernel for an OS client, you can link all OS clients to the same kernel image, saving disk space. Sharing kernels in this way, however, can result in weakened security, because any user can access and change the kernel image. If you decide to use a common kernel, remember that for OS clients to share the kernel, their root directories (and the directory containing the kernel) must all be on the same virtual disk. Thus, you should not distribute OS client root directories on different virtual disks.

OS client usr space (usr_dgux_54R201)

The **usr** virtual disk, whose file system mount point shown in Figure 2–3 is **/srv/release/dgux_54R201/usr**, is reserved for system-level programs, facilities, and software packages. The **/srv/release/dgux_54R201/usr** directory holds subdirectories that contain database and configuration files, administrator commands, stand-alone utilities and bootstraps, and user commands.

If you do not accept the default virtual disk size of 240,000 blocks, record the desired virtual disk size on the Virtual Disk Planning Worksheet later in Figure 2–6 or Appendix D.

X11 package space (usr_opt_X11_dgux_54R201)

The X11 virtual disk, shown as **usr_opt_X11_dgux_54R201** virtual disk in Figure 2–3 is required only if you have purchased the DG/UX X Window Package and you intend to install the DG/UX X Window System. This virtual disk, whose file system mount point is **/srv/release/dgux_54R201/usr/opt/X11**, contains X11 documentation and an X server development environment.

- If you do not accept the default virtual disk minimum size of 140,000 blocks, record the desired size on the Virtual Disk Planning Worksheet in Figure 2–6 or Appendix D.

CAUTION: Do not reduce the size of the X11 virtual disk.

Local directories

The system administrators at each site must determine how to set up home directories, work directories, and tools directories. These virtual disks vary in size according to the work requirements placed on them.

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 virtual disk 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. Chapter 4 explains adding user accounts.

Software packages (usr_opt_pkg)

You can create a separate virtual disk for each software package, or you can create a single virtual disk for all software packages. If you put all software packages on one virtual disk, it must be sufficiently large to accommodate the sum of the individual software 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 software package. Examples of software packages from Data General and other vendors are databases, spreadsheets, debugging tools, and publishing systems. The release notice that accompanies the software 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 software package whose file system will be readable and writable, if you want a 10 percent cushion for the superuser to use for reading and writing, add 10 percent of the package size (in Mbytes) for file system overhead requirements. If you don't want the 10 percent cushion, add nothing for overhead. For a read-only file system, add nothing for overhead.

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

Work directories

Work directories, like 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. But additional tools packages can be another candidate for a virtual disk. An appropriately named tools directory is easily recognizable and accessible to users on your system. While you may allow read and write access to a work directory, 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, 40,000 blocks are allocated to the **root** virtual disk for its file system. After the / file system is loaded, 12 Mbytes remain as free space that can be used 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.

Extra swap space (extra_swap)

Following procedures described in *Installing the DG/UX™ System*, you created a virtual disk for swap space of 50,000 blocks. However, additional space may be required to satisfy your applications' demand for virtual memory. You should consider adding more swap space if the message `From System: out of paging area space` appears on the system console or when the number of users or application load on the system increases. Use of swap space varies considerably from application to application. If you need to add swap space after your system is operational, see *Managing the DG/UX™ System* for details.

The operating system automatically balances paging activity among multiple swap areas in a system. For maximum performance, create equally-sized swap virtual disks on each disk drive. The maximum number of system-wide swap areas is eight.

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), **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, and 2.0 Gbytes. As examples, 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.

To determine the remaining space on a disk, use the **sysadm** sequence `Device -> Disk -> Physical -> List`, specify the disk by number (or select `All`), and then select normal display and no (no label).

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 Appendix B.

Write the DG/UX disk-drive device name and capacity on the Virtual Disk Planning Worksheets in Figure 2–6 (later).

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 virtual disk that combines a small, fast virtual disk (usually located in nonvolatile memory) with a large, slower virtual disk (usually located on a disk) to provide a large, fast storage resource.

Contiguous virtual disks

The simplest way to arrange virtual disks is to take the default starting address when you create each virtual disk until all space has been used. This way, the virtual disks will be contiguous on each drive. Or you can specify the whole physical disk address space as a virtual disk.

Multiple-piece virtual disks (aggregations)

If you will need enormous amounts of file space or have portions of disk space available on 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 a bulk virtual disk from partitions on three disk drives as follows:

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

You can specify up to 120 partitions for a virtual disk.

A software disk mirror and software striped disk are both aggregations. The mirror is an aggregation of two or more virtual disks; the striped disk is an aggregation of two or more partitions. We describe software disk mirroring and software disk striping in their own sections.

Software disk mirroring

A software disk mirror offers high availability through redundant images. If the disk that holds an image fails, DG/UX automatically sends all read and write operations to another image until the failing disk is repaired or replaced. Mirroring also increases disk I/O performance with read-intensive applications because the system can read from multiple images. You implement software disk mirroring with mirror virtual disks.

IMPORTANT: 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 disk array storage system manual supplied with the storage-system hardware.

When planning for disk mirroring, you must decide the number of images that must be available to allow access to the mirror. For example, if one image of a two–image mirror fails, do you want users to continue using the remaining image or do you want the mirror removed from service until the failing image is repaired and the images synchronized? If your priority is data integrity, you may want at least two images functioning. That way no data will be lost if the remaining image fails. However, the mirror will be out of service until the failing image is repaired.

You also need to decide how many images you want. If both data integrity and availability are priorities, you may choose to have three mirror images and require two to be functioning. For a more detailed discussion of this and other disk mirroring concepts, see *Managing the DG / UX™ System.*

When creating a software disk mirror, consider the following:

- The mirror virtual disk uses multiple virtual disks of the same size. Each of these virtual disk is an image. You must decide on the number of images to create: two or three. Three images provide great protection against data corruption and disk down time. Two images provide less protection than three, but do provide substantially more than an unmirrored virtual disk.
- Each image can reside on any disk drive attached to any disk controller. However, for maximum availability, you should arrange individual virtual disk mirror images on separate drives that are attached to different controllers.

For example, you could create two identically sized virtual disk images to form a mirror virtual disk that occupies 1,600,000 blocks, provides 800,000 blocks of storage capacity, and spans two different disk drives. Its arrangement follows:

Virtual Disk Image	Physical Device	Size (in blocks)
1 of 2	sd(ncsc(0),0,0)	800,000
2 of 2	sd(ncsc(0),1,0)	800,000
TOTAL		1,600,000

After software (or hardware) mirroring has been set up and started, the mirroring I/O occurs automatically. Chapter 3, section “Creating a virtual disk” explains how to create a software mirrored virtual disk.

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. For example, 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. You can choose software disk striping when you create a virtual disk.

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.

The requirements of a striped virtual disk are as follows.

- The virtual disk must consist of multiple same-size pieces.
- For best performance, each piece should reside on a different disk drive.
- 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 size of each piece must be an integer multiple of the stripe size; for example, if the stripe size is 16, the piece size could be 800,000 blocks.
- You cannot change the size of a striped virtual disk after creating it.
- You cannot use striping with the **root** and **usr** virtual disks.

For example, you could create three identically sized virtual disk partitions to form a 540,000-block striped virtual disk on three disk drives. Its arrangement follows:

Virtual Disk Partition	DG/UX Device Name	Size in blocks
1 of 3	sd(ncsc(0),1,0)	180,000
2 of 3	sd(ncsc(0),2,0)	180,000
3 of 3	sd(ncsc(0),3,0)	180,000
TOTAL		540,000

Software disk caching

Software disk caching lets DG/UX store data temporarily on a fast medium for later storage on a slower medium. With the fast medium, data to be read can be retrieved faster, and data to be written can be written faster (and, if the data is cached in memory, the data can wait in the cache until the system has spare cycles to write it to disk). Software disk caching can significantly improve disk performance. You can implement software disk caching with cache virtual disks.

The disk cached device consists of two devices: a front-end device and a back-end device. The front-end device is small and fast; the back-end device is large and relatively slow.

When creating a software disk cache, keep the following points in mind.

- The front-end device should be relatively fast and can be relatively small. It must be a virtual disk, which can be built on NVRAM memory (device name **nvrd()**) or on a fast hard disk. The ideal front end device is based on nonvolatile or battery backed-up RAM, which provides very high speed and stability (the battery lets the memory retain the cache information if power fails).
- The back-end device should be large and will usually be relatively slow. Like the front end, it must be the name of a virtual disk. The virtual disk must exist for you to make it a back end.

For conceptual information on disk caching, see *Managing the DG / UX™ System*. Chapter 3, section “Creating a virtual disk” explains how to create a disk cache.

Deciding where to mount file systems

After you determine the size and arrangement of virtual disks on disk drives, you need to decide where to put the virtual disks in the

DG/UX directory structure. These places are called *mount points*; they provide a directory pathname for the local file systems that are placed on the virtual disk.

There are two types of file system mounts: local and remote.

IMPORTANT: Virtual disks without file systems do not need mount points. Examples of such virtual disks include those used as local swap areas (explained in *Installing the DG / UX™ System*) and those used for special purposes by some application programs.

Local file system mounts

Figure 2–4 shows a typical DG/UX file system with virtual disks and their mount points. It shows the required virtual disks — **root**, **usr**, and **usr_opt_X11** — and four more virtual disks and mount points.

In the figure, circles represent virtual disks and the shaded circles represent the virtual disks mentioned above.

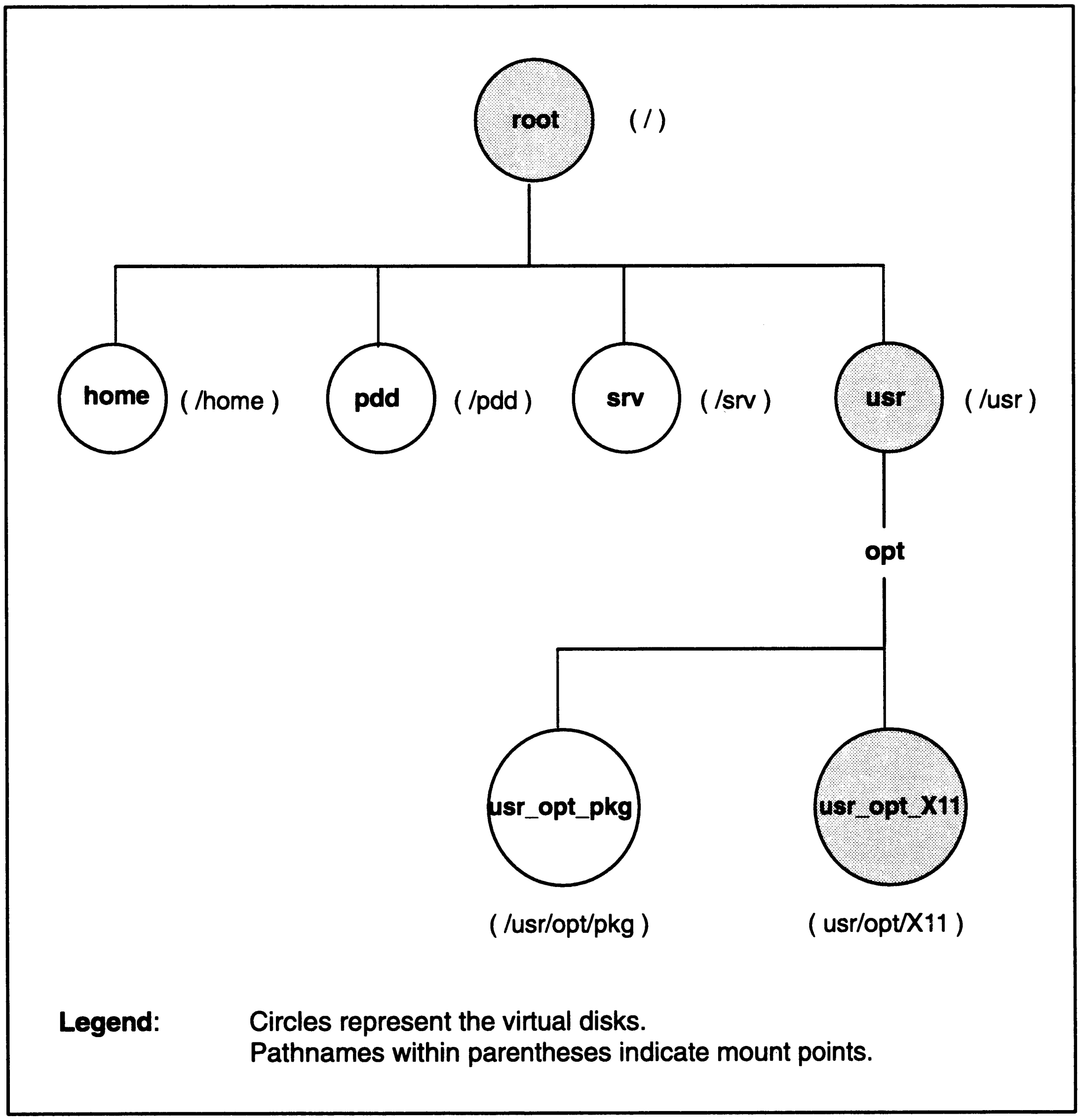


Figure 2–4 Sample Virtual Disks and Mount Points on the DG/UX File System

Figure 2–4 shows the following mount points.

Virtual Disk Name	Mount Point
home	/home
pdd	/pdd
srv	/srv
usr_opt_pkg	/usr/opt/pkg

Remote file system mounts

A remote file system is one that is located on a disk drive attached to a remote host that you can access via 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. Instructions on mounting the remote file system are given in Chapter 3. The following table shows some sample remote hosts (and their file system mount points) and the corresponding local mount points on the local host.

Remote Mount Point	Local Mount Point
cobra:/pdd/sam/image	/pdd/elixer/image
miracle:/pdd/notes	/pdd/notes
thing:/pdd/otis/papers	/pdd/proceedings/papers
spleen:/pdd/spleen/games	/pdd/spleen/games

where:
cobra is the remote hostname.
/pdd/sam/image is the remote host mount point for the file system.
/pdd/elixer/image is the local mount point for the remotely mounted file system.

Completing the virtual 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 disks, mirrors, caches, or aggregations (virtual disks with more two or more virtual disks) and how large will each virtual disk be?
- 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 mirror or 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 build the virtual disk and you can enter it then. For each such disk, wait until you build the disk (Chapter 3, section "Creating a virtual disk") to enter the size of the virtual disk in the worksheet.

Local virtual disk planning

Figure 2–5 shows a sample virtual disk planning worksheet. The sample entries appear in italic typeface. This system is an OS server with four OS clients that use a common kernel.

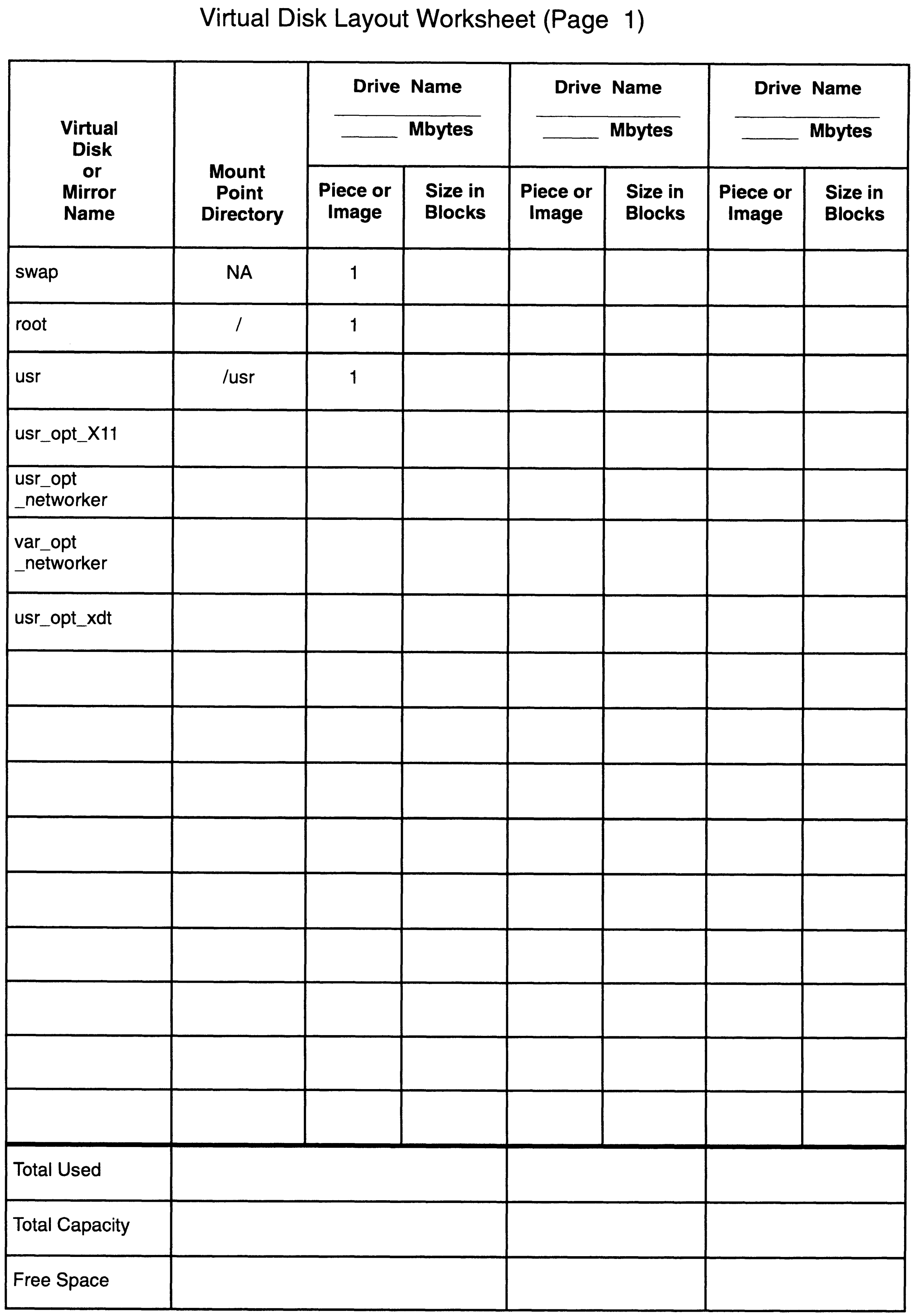
The sample system has two physical disks in a disk-array storage system. The system disk, **sd(dgsc(0),0,0)**, is a hardware mirrored physical disk with two disk modules. The other disk, **sd(dgsc(0),0,1)**, is a RAID group with five disk modules. Both disks have inherent data redundancy, therefore they are not software mirrored.

Figures 2–6 and 2–7 show blank virtual disk planning worksheets for you to complete. You may not need a second page (Figure 2–7) to hold your virtual disk information.

Sample Worksheet
Virtual Disk Layout Worksheet (Page 1)

		Drive Name *sd(dgsc(0),0,0)* *1,200* Mbytes		Drive Name *sd(dgsc(0),1,1)* *4,800* Mbytes		Drive Name __________ _____ Mbytes	
Virtual Disk or Mirror Name	Mount Point Directory	Piece or Image	Size in Blocks	Piece or Image	Size in Blocks	Piece or Image	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*		
srv	*/database /srv*			*1*	*5,000*		
srv_root	*/database /srv/release/ PRIMARY*			*1*	*142,000*		
srv_dump	*database /srv/dump*			*1*	*37,000*		
srv_swap	*database /srv/swap*			*1*	*200,000*		
Total Used	*1,067,000*			*5,544,000*			
Total Capacity	*2,457,600*			*9,830,400*			
Free Space	*1,390,600*			*4,286,400*			

Figure 2–5 Sample Virtual Disk Planning Worksheet



Virtual Disk Layout Worksheet (Page 1)

Virtual Disk or Mirror Name	Mount Point Directory	Drive Name ________ ______ Mbytes		Drive Name ________ ______ Mbytes		Drive Name ________ ______ Mbytes	
		Piece or Image	Size in Blocks	Piece or Image	Size in Blocks	Piece or Image	Size in Blocks
swap	NA	1					
root	/	1					
usr	/usr	1					
usr_opt_X11							
usr_opt _networker							
var_opt _networker							
usr_opt_xdt							
Total Used							
Total Capacity							
Free Space							

Figure 2–6 Virtual Disk Planning Worksheet, Page 1

Virtual Disk Layout Worksheet (Page of)

Virtual Disk or Image Name	Mount Point Directory	Drive Name ________ ______ Mbytes		Drive Name ________ ______ Mbytes		Drive Name ________ ______ Mbytes	
		Piece or Image	Size in Blocks	Piece or Image	Size in Blocks	Piece or Image	Size in Blocks
Total Used							
Total Capacity							
Free Space							

Figure 2–7 Virtual Disk Planning Worksheet, Second or Subsequent Page

Remote file systems

Figure 2–8 is a sample worksheet for the remote hosts and mount points listed above. Again, sample entries appear in italic typeface. Figure 2–9 is a remote file system planning worksheet for you to complete.

Remote Hostname	Remote Mount Point	Local Mount Point
cobra	*/pdd/sam/image*	*/pdd/elixer/image*
miracle	*/pdd/notes*	*/pdd/notes*
thing	*/pdd/otis/papers*	*/pdd/proceedings/papers*
spleen	*/pdd/spleen/games*	*/pdd/spleen/games*

Figure 2–8 Sample Remote File System Planning Worksheet

Remote Hostname	Remote Mount Point	Local Mount Point

Figure 2–9 Remote File System Planning Worksheet

Where to go next

The following chapter explains how to use **sysadm** to create the virtual disks and file systems you planned in this chapter. Generally, you will want to continue to the next chapter. If not, do what you wish as defined in the task table included in the Preface, "About this manual."

End of Chapter

3 Creating virtual disks and file systems

This chapter shows how to create each virtual disk you planned in Chapter 2 and how to add, mount, and optionally export its file system. For a definition of virtual disk and comparison to logical disk, see Chapter 1, section "Terms and concepts."

Major sections in the chapter proceed as follows.

- Registering a disk drive
- Creating a virtual disk
- Creating a local file system
- Adding a local file system
- Adding a remote file system
- Converting a physical disk to virtual disk format
- Where to go next

We assume that all your disk drives were software formatted and identified to the DG/UX kernel at DG/UX system installation. If you have added drives since installation and have not identified the drives to the kernel and software formatted them, do so as explained in Chapters 10 and 9; then return here.

Registering a disk drive

Registering a disk drive makes its virtual disks known to the system. You cannot access a virtual disk or file system until the physical disk that holds it is registered. To register a disk, follow this path through **sysadm**:

```
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.

To list the registered physical disks (disk drives), follow this path through **sysadm**. A sample “normal” display 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 [here shown as **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
<Various system partitions>                    1315922           16
tmp                                            1315938        25000
var_mail                                       1340938        20000
  .                                               .              .
```

Creating a virtual disk

As explained in the previous chapter, you will typically create a virtual disk on a hard disk medium. Single file-system devices on which you would generally not create virtual disks include magneto-optical disk drives, diskettes, and memory file systems. Using file systems on these devices is explained later in this chapter in the appropriate “Adding a local file system” section.

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 the Virtual Disk Planning Worksheets you completed in Chapter 2 or Appendix D for the following information:

- Virtual disk name
- Whether or not you want software disk striping
- Names of DG/UX disk(s) that will hold the virtual disk, unless you plan to let **sysadm** choose the physical disk, in which case someone should have written “n/a” in the physical disk specification box in the worksheet
- Size of virtual disk partition(s) 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

IMPORTANT: If you have OS clients, create the required virtual disks as soon as possible; your OS client cannot operate until you have created the required virtual disks. Procedures for making the OS client operational are described in Chapter 7.

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 disk(s) the space comes from.
- Creating a virtual disk by physical disk(s) to partition. Use this when you *do* care which physical disk(s) 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 disk(s) (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 disk(s) 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 these steps. To the **sysadm** prompts, get help with **?** as needed.

1. Follow this path through **sysadm**:

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

   Sysadm prompts

   ```
   New Virtual Disk Name:
   ```

2. Enter a name for the new virtual disk. This name will identify the virtual disk for mounting purposes by its entry in directory **/dev/dsk**. After a file system is created on the virtual disk and the file system is mounted, users will access the virtual disk's file system by the mount point directory name. The name should be descriptive and memorable; for example,

   ```
   New Virtual Disk Name: test ↵
   Striped? [no]
   ```

3. Press Enter for the default, no. (If you want to create a striped disk, skip to the section "Creating a virtual disk by by physical disk(s) to partition.") For example,

   ```
   Striped? [no] ↵
   Create File System? [yes]
   ```

4. Each virtual disk needs a file system before people can use it to store files. You can have **sysadm** create a file system now or later (explained later in this chapter). 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]
   ```

5. Press Enter for the default, yes. If you want to create a virtual disk by any other method, skip to the section that describes doing so. For example,

   ```
   Select Space by: [Size alone] ↵
   Size in Blocks: (1-n) [1]
   ```

6. Enter the size you want, as a decimal number of 512-byte disk blocks, for the new virtual disk. **Sysadm** will take space from available physical disk(s) to match the amount you specify. You can enlarge or shrink the virtual disk later on, if you want. 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. If you want to create another virtual disk, go to the appropriate section. Eventually, even if you did create a file system, you will need to add it. Creating and adding a file system are explained later in this chapter.

Creating virtual disk by physical disk(s) to partition

Use this method when you care which physical disk(s) 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 the back-end virtual disk on a standard disk and the front end virtual disk, perhaps on a NVRAM memory board. Note that, like any writable disk, a NVRAM board must be software formatted (Chapter 9) before you can create a virtual disk on it.

1. Follow this path through **sysadm**:

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

   Sysadm prompts

   ```
   New Virtual Disk Name:
   ```

2. Enter a name for the new virtual disk. This name will identify the virtual disk for mounting purposes by its entry in directory **/dev/dsk**. After a file system is created on the virtual disk and the file system is mounted, users will access the virtual disk's file system by the mount point directory name. 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.

   For example, you might type

   ```
   New Virtual Disk Name: pdd ↵
   Striped? [no]
   ```

Software disk striping provides quick random access and quick sequential reads. When planning for a software striped virtual disk, remember the following rules.

- The virtual disk must consist of multiple same-size pieces.
- For best performance, each piece should reside on a different disk drive.
- 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 size of each piece must be an integer multiple of the stripe size; for example, if the stripe size is 16, the piece size could be 800,000 blocks.
- You cannot change the size of a striped virtual disk after creating it.
- You cannot use striping with the **root** and **usr** virtual disks.

For more background information on software striping, see "Software disk striping" in Chapter 2.

3. If you want to create a striped virtual disk, enter **y** and continue with this step. If you don't want the new disk to be striped, enter **n** and skip to step 5. For example, if you don't want software striping,

```
Striped? [no] ↲
```

4. **Sysadm** prompts

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

The stripe size is 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 should be a power of 2, and must be an integral divisor of the number of blocks in each piece. Generally, we suggest the default. Whatever number you specify, remember later to make the virtual disk size an integer multiple of that size. For example,

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

5. **Sysadm** prompts

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

6. Each virtual disk needs a file system before people can use it to store files. You can have **sysadm** create a file system now or later (explained later in this chapter). Generally, it's easier to have the file system created here.

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.

Some database management systems create their own file systems on disks. If you plan to use such a system on this virtual disk, you do not need to create a file system on it.

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

```
Create File System? [yes] ↲
Select Space by: [xxx]
```

The default displayed depends on your response to the `Striped?` prompt.

7. Make sure that your answer to this prompt is "Disk to partition." You can specify this by a unique abbreviation (like **Di**) or if the "Disk to partition" value is the default, taking the default. For example,

```
Select Space by: [xxx] Disk ↲
Disk to Partition From:
```

8. Your answer will select the physical disk from which the new virtual disk or piece will be created.

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 mirror or striped disk.

If you are creating a virtual disk that will be the front end of a software cache, and you plan to specify battery backed up NVRAM memory as the disk to partition from (use the form **nvrd()**), then the NVRAM memory board must be installed.

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)
```

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:

```
Disk to Partition From: 2↲
Length of Piece in Blocks: (1-n)
```

9. This sets 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, later on you can enlarge or shrink the size later, if you want.

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.

IMPORTANT: 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):

```
Length of Piece in Blocks (1-n) 204800 ↵
Starting Block (optional):
```

10. This prompt lets you specify the starting address of the virtual disk on the physical disk. Unless you have computed your own disk layout for some special purpose, we suggest the default:

```
Starting Block (optional): ↵
Do you want to specify more pieces for this virtual
   disk? [yes]
```

11. If this virtual disk has all the space you want for it (the virtual disk is complete or a complete image), enter **no**. If you want to specify another piece (for example, if you are creating a striped disk and want to specify a stripe on a different physical disk), answer **yes**.

    If you enter yes, **sysadm** prompts for the disk to use; return to step 8.

    After you answer **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 "pdd" created.
Virtual disk "pdd" made a volume.
File system created on virtual disk /dev/dsk/pdd
```

You have created a virtual disk by physical disk(s) to partition. 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. Eventually, even if you did create a file system, you will need to add it. Creating and adding a file system are explained later in this chapter.

Example — Creating a virtual disk by physical disk(s) to partition

The following example shows creation of two virtual disks to be used for mirroring. It does not actually create the mirror; to create the mirror, see “Creating a mirror virtual disk,” later.

```
Device -> Disk -> Virtual -> Create ->
New Virtual Disk Name: pdd.image1 ↲
Striped? [no] ↲
Create File System? [yes] ↲
Select Space by: [xxx] Disk↲
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 "pdd.image1" created.
Virtual disk "pdd.image1" made a volume.
File system created on /dev/dsk/pdd.image1

                Virtual Disk Menu

  1  Create  ...
Enter ...  1 ↲
New Virtual Disk Name: pdd.image2 ↲
Striped? [no] ↲
Create File System? [yes] n↲
Select Space by: [xxx] Disk ↲
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 "pdd.image2" created.
Virtual disk "pdd.image2" made a volume.
```

Creating a virtual disk by name of existing virtual disk(s) (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, and/or none of these. To create an aggregation, follow these steps.

IMPORTANT: Creating an aggregation will render useless any data currently residing on the virtual disk(s) you specify.

1. Follow this path through **sysadm**:

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

   Sysadm prompts

   ```
   New Virtual Disk Name:
   ```

2. Enter a name for the new aggregation virtual disk. This name will identify the virtual disk for mounting purposes by its entry in directory **/dev/dsk**. After a file system is created on the virtual disk and the file system is mounted, users will access the virtual disk's file system by the mount point directory name. The name should be descriptive and memorable; for example,

   ```
   New Virtual Disk Name: temp_storage ↵
   Striped? [no]
   ```

3. If you want the entire aggregation to be software striped, answer **yes** and continue with the next step. If you want only those virtual disks that are already striped to be striped (or none if there are none), answer **no** and skip to step 5. For example, if you don't want software striping,

   ```
   Striped? [no] ↵
   ```

4. If you answer yes, **sysadm** prompts

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

   The stripe size is the number of blocks that are used on one partition of a striped virtual disk, before going on to the next partition. The stripe size must be a power of 2, and must be an integral divisor of the number of blocks in each partition. Generally, we suggest the default. If you specify the default, remember later to make the virtual disk size an even multiple of 16. For example,

   ```
   Stripe Size (in blocks): [16] ↵
   Create File System? [yes]
   ```

5. **Sysadm** prompts

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

6. Each virtual disk needs a file system before people can use it to store files. You can have **sysadm** create a file system now or later (explained later in this chapter). 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]
```

7. Since you are creating an aggregation virtual disk, it's likely that you will select the choice "Name of an existing virtual disk." If so, select that choice; for example,

```
Select Space by: [xxx]  Name ↵
Child Virtual Disk:
```

8. **Sysadm** is asking 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 aggregate. For example, to pick **temp** from the list above,

```
Child Virtual Disk:  13 ↵
Do you want to specify more pieces for this virtual
   disk? [yes]
```

9. If you want to specify another virtual disk or create a partition to be part of the aggregation, enter **yes**. If you enter yes, **sysadm** prompts for the child virtual disk again; return to step 8.

When 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. Eventually, if you did not create a file system on the disk, you will need to create one; even if you did create a file system, you will need to add it.

Creating and adding a file system are explained later in this chapter.

Example — Creating a virtual disk by names of virtual disks (aggregation)

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: [xxx] Name⤶
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
```

Creating a mirror virtual disk

Read this section when you want to create a software mirror from two or three existing virtual disks (created using the "Creating a virtual disk by physical disk(s) to partition" steps earlier).

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. Conceptual details appear in Chapter 2, section "Software disk mirroring" and the manual *Managing the DG/UX™ System.*

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.

- 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.
- Mirror the virtual disk images as described in this 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, and *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.
- Add and mount the local file system as explained in a later section, "Creating and adding a local file system."

Building the mirror virtual disk

To create a mirror from two or three existing virtual disks, follow these steps.

1. Follow this path through **sysadm**:

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

   Sysadm prompts

   ```
   Virtual Disk:
   ```

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

3. Enter the number of images of the mirror virtual disk that must be available and in synchronization before the disk can be mounted. 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 ↵
Maximum Number of Lost Images to Tolerate: (0-3) [0]
```

4. This governs how many lost images DG/UX 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, we suggest 1 less than the number of images; this will give users access to the mirror's information if one image fails. You can then fix the problem and start synchronization to 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↵
Automatically Synchronize on System Boot? [yes]
```

5. 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 using **sysadm**. On the down side, the process of synchronization may slow response time throughout the system; also, if you answer no, you (not DG/UX) can control which image is the master. Decide whether you want automatic synchronization for this mirror virtual disk and answer. For example,

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

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

7. This prompt lets you 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 2. 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 ↵
New Name for Image:
```

8. This prompt lets you specify a different name for the virtual disk *image* you specified in step 2. 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 ↵
Another image? [yes]
```

9. If you have specified all the virtual disk images you want (at least two), enter **no**; skip to step 11.

   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] ↵
Child Virtual Disk:
```

10. Enter the name of the second (or third) virtual disk image. As always, 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 ↵
```

11. **Sysadm** prompts

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

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 2 above to the child virtual disk you specified in step 10. 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 2 is source you want and the image you specified in step 10 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, explained later in this chapter, and then use **sysadm** to synchronize the images. Or if you said **no** to the synchronize prompt, you will need to use **sysadm** to synchronize the images.

After the images have been synchronized, the system will maintain them as copies. While synchronized, the images will not be accessible as individual virtual disks. However, you can unsynchronize the mirror with **sysadm** and then access one of the disks (as for backup), and then synchronize.

Example — Creating a mirror 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"
```

Software disk caching

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

A software cache can include one or more front ends (often, but not necessarily, NVRAM memory boards on which virtual disks have been created) 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.

Software disk caching involves concepts that you may want to understand. Generally for these, you can take the **sysadm** defaults without seriously degrading the performance of the cache. All the concepts are defined in *Managing the DG/UX System.*

1. Make sure the back-end virtual disk exists. (You can create the front-end virtual disk when you create the cache.) If not, create it as explained in "Creating a virtual disk by physical disk(s) to partition," earlier.
2. Follow this path through **sysadm**:

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

   Sysadm prompts

   ```
   Virtual Disk:
   ```

3. Enter 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 **customer_data**,

   ```
   Virtual Disk: customer_data ↵
   Cache Reads? [yes]
   ```

4. For this and the following prompts, you can take the defaults (as shown), or specify other values after you get help using **?** and/or consult *Managing the DG/UX System* for additional information.

   ```
   Cache Reads? [yes] ↵
   Cache Writes? [yes] ↵
   Cache Only File System Metadata? [no] ↵
   Asynchronous Write Policy: [First write] ↵
   Read Weight: [1] ↵
   Write Weight: [1] ↵
   Search Percentage: (0-100) [10] ↵
   Flusher Type: [Cyclic] ↵
   New Name for Cache: [xxx]
   ```

5. The cache name should differ from the names of virtual disks that comprise it. The default cache name is the name of the back-end virtual disk (shown as **customer_data** above). **Sysadm** asks this question to let you specify another cache name; later it will let rename the back-end disk. You might choose to provide a name that implies the entire cache. If not, you can keep the existing name. For example,

   ```
   New Name for Cache: [customer_data] customer_cache ↵
   New Name for Back End:
   ```

6. You might want to specify a new name for the back-end virtual disk. If not, press Enter to retain the original back end name. For example, to retain the back-end name,

   ```
   New Name for Back End: ↵
   Specify a Front End? [yes]
   ```

7. You need at least one front-end virtual disk. So answer **yes**:

   ```
   Specify a Front End? [yes] ↵
   Front End Virtual Disk: cache_front ↵
   Specify Another Front End? [yes]
   ```

8. Enter the name of the front-end virtual disk (which might, but need not, be a NVRAM memory board). If the virtual disk you specify does not exist, **sysadm** will ask the name of the disk to partition and the size of partition; then **sysadm** will create the disk. For example, for the existing front end **nvram_board_1**,

   ```
   Front End Virtual Disk: nvram_board_1 ↵
   Specify Another Front End? [yes]
   ```

9. 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, finished, enter **n**. For example,

   ```
   Specify Another Front End? [yes] n ↵
   Virtual disk inserted at customer_cache.
   Child virtual disk made a volume.
   ```

   Sysadm has created the cache (here, **customer_cache**). If you want to create another virtual disk, go to the appropriate section.

   If the cache back-end virtual disk contains a file system, you need not create one; you can simply add (mount) the file system to make it accessible to users. If the back-end disk does not have a file system, you will probably want to create one (unless your application recommends against this). Adding and creating file systems are explained later in this chapter.

   For a discussion of software disk caching concepts, see *Managing the DG/UX™ System.*

Creating and adding a local file system

A local file system is one you create on a virtual disk attached to your computer system. A file system is needed to let store files on the virtual disk. Some database management systems create their own file systems on virtual disks. If you plan to use such a system on a virtual disk, you do not need to create a file system on it.

If you created a file system when you created each virtual disk, you do not need to create any file systems. You do need to *add* the file systems, however, as explained in later sections.

In the process of adding a file system, you *can* create a virtual disk. If the virtual disk name you specify for the file system does not exist, **sysadm** will eventually ask for a number of disk blocks and then create the virtual disk for you. If you want to control the creation of the physical disk, however, we suggest that you create the virtual disk first as explained earlier in this chapter.

There are two steps involved:

- Creating the file system, which actually creates the software structure on the disk; and
- Adding the file system, which adds its name to the file system table (**/etc/fstab**, which lets the file system be mounted automatically at future startups) and then mounts the file system.

The Create function does both; the Add function simply adds the file system.

Creating a file system of type DG/UX

This section describes how to use the Create function to create and add a file system of type DG/UX.

IMPORTANT: For the following devices, you do not need to create a file system. For a CD-ROM, skip to the section “Adding a file system for a CD-ROM device.” For a file system on a diskette formatted for DOS, skip to “Adding a file system for a diskette formatted for DOS.” For a memory-resident file system, skip to “Adding a memory file system.”

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

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

Sysadm prompts

```
Virtual Disk: []:
```

2. Enter the name of the virtual disk on which you want to create the file system. If the virtual disk you specify does not exist, **sysadm** will create it; however, you may need to create virtual disks using the **sysadm** Virtual disk path, as covered earlier.

   If a list of virtual disks is not displayed, as with ASCII **sysadm**, enter **?** for such a list. To specify **pdd**, you would enter

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

3. **Sysadm** is asking for custom options you want to add the **mkfs** command that will create the file system. Among the options are items like the disk allocation region (DAR) size and inode density. You can access the shell (for example with **!csh)** and look at the **mkfs** man page for more information. Generally, you do not need any options, in which case press Enter:

   ```
   Mkfs options: ↵
   Mount Directory:
   ```

4. The mount directory is the place in the directory structure where the system will automatically graft the file system at system startup. This will form the pathname through which users will access the file system, therefore it is very important. The mount directory you specify here need not exist; **sysadm** will create it for you later on.

   Specify the mount directory's full pathname from the root directory. We suggest that you use the virtual disk name as the last entry in the mount point specification. For example,

   ```
   Mount Directory: /pdd ↵
   Exportable? [no]
   ```

   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 via Modify, without harm to the file system.

5. If you want to export the file system, answer yes (**y**) and continue with this step; you will need to specify export options. If you do not want the file system to be exportable, answer **n** (or just Enter) and skip to step 7.

   Sysadm prompts

   ```
   Export options?
   ```

   Export options restrict the access to an exported file system. The export options are as follows in Table 3–1.

Table 3–1 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]...*	Exports the directory as read-only to all computer systems except those specified by this option. Computer systems specified in this option have read-write access to the directory. Allowing another computer system read-write access to the directory does not override normal file and directory permissions. Read-write is the default.
anon=*uid*	Users can access an exported file system only if their password entries exist in the remote computer system's **/etc/password** file. Alternatively, users can access the exported file system by way of the **anon** option, which establishes an effective user ID for anonymous users. Superusers (user ID **0**) can access an exported file system only through the **root** option (see next option). 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.
Example: **-ro,root=shaman:hawk:crawdaddy** exports the file system as read-only and allows root access only to superusers from hosts **shaman**, **hawk**, and **crawdaddy**.	

6. Decide on the export options you want and specify them. Precede the first option with a hyphen (–) and no intervening space, and follow with a list of the export options you want separated by commas as shown in the example. For example,

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

7. **Sysadm** warns you that

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

8. 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), confirm; for example,

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

Sysadm now tries to create the file system on the virtual disk, add the file system (to the **fstab** file) and mount 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
```

You have created a file system on the virtual disk. Repeat these steps or use the Add function (below) for each virtual disk listed in the Virtual Disk Planning Worksheets that you completed in Chapter 2 or Appendix D.

Adding a local file system

Adding a file system includes both of the following steps: adding the file system name to the file system table, **/etc/fstab**, so that the file system will be mounted automatically at system startup; and mounting the file system. **Sysadm**'s Create function includes these two steps. Do not use Add when you want to add a file system that you (or someone) has manually unmounted; instead, use the **sysadm** Mount sequence.

This section explains adding a file system for a hard disk, CD-ROM, DOS diskette, and memory file. Continue to the appropriate section.

Adding a file system of type DG/UX

Generally, a type DG/UX file system is the type you choose for a virtual disks on standard hard disk drives and for software disk caches. 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.

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

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

Sysadm prompts

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

2. Specify a type. The most common type of file system is DG/UX. There are also **cdrom** and **ramdisk** types, but these are explained in later sections. Generally, take the default:

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

3. Specify the name of the virtual disk that holds the file system. As always, if **sysadm** does not display a list of virtual disks, you can get one by entering **?**.

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, if you are adding a file system for the mirror named **clients**, whose constituent virtual disk images are **clients.image1** and **clients.image2**, specify **clients** for the virtual disk name, and **/clients** as the mount directory.

For example, for the virtual disk **pdd**:

```
Virtual Disk: pdd ⤵
Mount Directory:
```

4. The mount directory is the place in the directory structure where the system will automatically graft the file system at system startup. This will form the pathname through which users will access the file system, therefore it is very important. The mount directory you specify here need not exist. If it does not exist, **sysadm** will offer to create it for you later on.

   Specify the mount directory's full pathname from the root directory. We suggest that you use the virtual disk name as the last filename in the mount point pathname. For example,

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

5. This determines whether, when the file system is mounted, users will be able to write to it. To make the file system read-only (not writable by any user, including root), enter **ro** or **read-only**. But generally you will take the default by pressing Enter. For example,

```
Write Permission: [Read/Write] ⤵
Dump Frequency: [Daily]
```

6. Dump frequency determines how often your file system is archived onto tape. You can base a file system's dump frequency on the importance of the data and its volatility. If **sysadm** does not display a list of frequencies, enter **?** to display such a list.

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

   Decide on the dump frequency you want and specify it. For example,

```
Dump Frequency: [Daily] ⤵
Fsck Pass Number: [1]
```

7. **Fsck**(1M) is the file system-checking facility. The Fsck pass number indicates the order in which 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 *Managing the DG / UX*™ *System* for details.

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

   ```
   Fsck Pass Number: [1] ↵
   Fsck Logging? [no]
   ```

9. 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. We recommend using logging primarily when rapid recovery and high availability are crucial.

   Decide on your answer, **no** (default) or **yes**, and specify it. For example,

   ```
   Fsck Logging? [no] ↵
   Exportable? [no]
   ```

   Making a file system exportable file system lets other hosts connected to the network mount it on their own file systems. Making a file system exportable creates an entry in the **/etc/exports** file. The Exportable value is one you can change later on via Modify, without harming the file system.

10. If you want to be able to export the file system, answer yes (**y**) and continue with this step; you will need to specify export options. If you do not want the file system to be exportable, answer **n** (or just press Enter) and skip to step 12.

    Sysadm prompts

    ```
    Export options?
    ```

11. For the export options, see Table 3–1 in a previous section; specify the export options you want (for example, **rw=accounts,purchasing**); and then return here to continue. After you answer, **sysadm** prompts

12. `Mount and export the file system? [yes]`

13. If the answers are the ones you want, confirm by pressing Enter, and answer the confirmation question with Enter as well. For example,

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

    If the mount directory you specified in step 4 does not exist, **sysadm** prompts

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

14. If you want to create the directory, answer **yes** or press Enter; if not, enter **n** to have **sysadm** abort and start over.

    Sysadm displays

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

    The file system is now mounted at the desired mount point. Repeat this procedure for each virtual disk whose file system you want added to the **fstab** file and mounted. Consult the Virtual Disk Planning Worksheets you completed in Chapter 2 or Appendix D.

Adding a file system for a CD-ROM device

A CD-ROM device is a compact disk read-only memory disk drive that is formatted in either the High Sierra or ISO 9660 standard. 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 Virtual Disk Planning Worksheets that you completed from Chapter 2 or Appendix D.

A file system for a CD-ROM is a read-only file system. To add a file system that's on a CD-ROM disk drive, follow these steps:

1. Make sure a CD has been inserted correctly in the CD drive.
2. Follow this path through **sysadm**:

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

    Sysadm prompts

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

3. Enter **cdrom**:

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

4. Enter the name of the device file in the **/dev** directory that belongs to the CD-ROM disk drive. For example, if you know the drive device filename is **/dev/pdsk1**:

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

5. The mount directory is the place in the directory structure where the system will automatically graft the file system at system startup. This will form the pathname through which users will access the file system, therefore it is very important. The mount directory you specify here need not exist; **sysadm** will create it for you later on.

   Specify the mount directory's full pathname from the root directory. We suggest that you use the virtual disk name as the last filename in the mount point pathname. For example,

   ```
   Mount Directory: /usr/resources ↵
   Exportable? [no]
   ```

   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.

6. A CD-ROM offers a large read-only resource. Generally, you will want to answer yes; for example,

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

7. See the export options description, Table 3–1 in a previous section, for a review; specify the export options you want; and then return here to continue. After you answer, **sysadm** prompts

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

8. If the answers are the ones you want, confirm by pressing Enter, and answer the confirmation question with Enter as well. For example,

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

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

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

9. If you want to create the directory, answer **yes** or press Enter; if not, enter **n** to have **sysadm** abort and start over.

   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 whose name you want added to the **fstab** file and which you want mounted.

Adding a file system for a diskette formatted for DOS

A DOS file system is on a diskette formatted for MS-DOS. For these files, 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. Make sure the diskette has been inserted correctly in the diskette drive.
2. Follow this path through **sysadm**:

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

   Sysadm prompts

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

3. Enter **dos**:

   ```
   File System Type: [dg/ux] dos ↵
   Device file:
   ```

4. Enter the name of the device file in the **/dev** directory that belongs to the diskette drive that holds the DOS diskette. For example, if you know the drive device filename is **/dev/pdsk3**:

   ```
   Device file: /dev/pdsk3 ↵
   Mount Directory:
   ```

5. The mount directory is the place in the directory structure where the system will automatically graft the file system at system startup. This will form the pathname through which users will access the file system, therefore it is very important. The mount directory you specify here need not exist; **sysadm** will create it for you later on.

   Specify the mount directory's full pathname from the root directory. We suggest that you use the virtual disk name as the last entry in the mount point specification. For example,

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

6. This determines whether, when the file system is mounted, users will be able to 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] ↵
   Exportable? [no]
   ```

7. Exporting a file system gives other systems in your network access to the file system by allowing them to mount it on their own system. You may grant such access by specifying the correct export option. Generally, you will not want to make a diskette file system exportable; if you do, answer yes. If you do not want the file system to be exportable, answer **n** (or just Enter) and skip to step 9.

   Sysadm prompts

   ```
   Export options?
   ```

8. For a list of export options, see the export options description, Table 3–1 in a previous section; specify the export options you want; and then return here to continue.

9. **Sysadm** prompts

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

10. If the answers are the ones you want, confirm by pressing Enter, and answer the confirmation question with Enter as well. For example,

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

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

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

11. If you want to create the directory, answer **yes** or press Enter; if not, enter **n** to have **sysadm** abort and start over. **Sysadm** displays

```
File system added: /spreadsheets
File system mounted: /spreadsheets
```

    After you answer all the prompts for information, the file system is mounted at the desired mount point. When you have finished, return to the **sysadm** Main Menu by entering **^**.

Adding a memory file system

A memory file system is a diskless file system that, when added (mounted), 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. The contents of a memory file system are temporary; they are lost at system shutdown. A memory file system is not same as a disk cache, which may include a virtual disk created in NVRAM (battery backed up) memory.

1. To add a memory file system, follow this path through **sysadm**:

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

    Sysadm prompts

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

2. Enter **ramdisk**:

```
File System Type: [dg/ux] ramdisk ↵
Device file:
```

3. 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 ↲
   Mount Directory:
   ```

4. The mount directory is the place in the directory structure where the system will automatically graft the file system at system startup. This will form the pathname through which users will access the file system, therefore it is very important. The mount directory you specify here need not exist; if it does not, **sysadm** will offer to create it for you later on.

   Specify the mount directory's full pathname from the root directory. For example,

   ```
   Mount Directory: /memdisk ↲
   Use Wired Memory? [no]
   ```

5. If you specify wired memory, then when the memory file system is mounted the system will retain the entire file system in your computer's physical 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 **yes**, before mounting the file system make sure your computer has enough physical memory to hold the memory file system. For example,

   ```
   Use Wired Memory? [no]  yes↲
   Maximum File Space: [2048] ?
   ```

6. 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 a lot of 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] ↲
   Maximum File Count: [16384]
   ```

7. Enter the maximum number of disk files that can be present in the memory file system. For example,

   ```
   Maximum File Count: [16384] 128 ↲
   Exportable? [no]
   ```

8. Exporting a file system gives other systems in your network access by allowing them to mount the file system on their own system. Allowing another system to mount your file system does not necessarily give that system superuser access to the file system. You may grant such access by specifying the correct export option. Decide on your answer and enter it; for example,

   ```
   Exportable? [no] ↵
   ```

   If you answered **no** to the Exportable prompt, skip to step 10. If you answered **yes**, **sysadm** prompts

   ```
   Export Options:
   ```

9. For the export options, see Table 3–1 in a previous section; specify the export options you want; and then return here to continue.

10. **Sysadm** prompts

    ```
    Mount ... the file system? [yes]
    ```

11. The prompt includes the words “and export” if you specified that the file system was exportable. If you want to mount the file system now, answer yes; otherwise answer **n** and **sysadm** will simply add the file system name to **fstab** for later automatic mounting. For example,

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

    If the mount directory you specified in step 4 does not exist, **sysadm** prompts

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

12. If you want to create the directory, answer **yes** or press Enter; if not, enter **n** to have **sysadm** abort and start over.

    Sysadm displays

    ```
    File system added: /memdisk
    File system mounted: /memdisk
    ```

    After you answer all the prompts for information, the file system is mounted at the desired mount point. When finished, return to the **sysadm** main menu by entering **^**.

Adding 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 you completed in Chapter 2 or Appendix D for this information.

Note that before you can mount a remote file system, the remote host must export that file system. A remote host can 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 ↲
   Remote host name:
   ```

3. Enter the remote hostname. For example,

   ```
   Remote Host Name: igor ↲
   Remote Mount Directory:
   ```

4. Enter the remote mount directory name. The file system must actually be mounted at that mount point on that host. For example,

   ```
   Remote Mount Directory:  /accounts/midwest/minn_brands↲
   Write Permission: [Read/Write]
   ```

5. Specify the write permissions; the default is read/write. For example,

   ```
   Write Permission: [Read/Write] ↲
   NFS Mount Type: [Hard]?
   ```

6. 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. Alternatively, a soft mount allows your operation to abort (time out) so that resources are not tied up while the remote host is hung.

   In general, a hard mount is most suitable for write-intensive operations performed with remotely mounted file systems. A soft mount is most suitable for read-only operations in which you risk no loss of data. If you anticipate infrequent write operations, you may still choose a soft mount to guard against unnecessary down time for read operations while accepting the risk of occasional data loss.

Furthermore, 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. For example, 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.

For example,

```
NFS Mount Type: [Hard]? soft ↵
Interruptible?
```

7. This value determines whether 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 whatever key to which you mapped the interrupt function) cancels the operation. For example,

   ```
   Interruptible? no ↵
   Retry in background? [yes]
   ```

8. This value determines whether or not the network will retry in background mode if the OS server does not respond. For example,

   ```
   Retry in background? [yes] ↵
   Mount the file system? [yes]
   ```

9. If you want to mount the file system now, answer yes; otherwise answer **n** and **sysadm** will simply add the file system name to **fstab** for later automatic mounting. For example,

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

10. If the answers are the ones you want, confirm by pressing Enter, and answer the confirmation question with Enter as well. For example,

    ```
    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 /xxx does not exist.
    Do you wish to create it? [yes]
    ```

11. If you want to create the directory, answer **yes** or press Enter; if not, enter **n** to have **sysadm** abort and start over.

    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. When finished, return to the **sysadm** Main Menu.

Converting a physical disk to virtual disk format

When you installed DG/UX 5.4 Release 3.00, **sysadm** prompted you for permission to convert each physical disk it found from logical disk format to physical disk format. If you answered yes to these prompts, all your physical disks are now in virtual disk format. However, the DG/UX system lets you register and use any logical-format disk in compatibility mode, so it isn't mandatory to convert. The manual *Installing the DG/UX™ System* includes a table for you to note disks to convert and not to convert.

The conversion process converts only the virtual disk metadata format; it does not touch file system metadata or user data. The conversion utility converts the virtual disk 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 format takes only a moment per disk. Since the virtual disk format requires no additional space, there is no risk of format conversion failure because of limited disk 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, make sure all the physical disks are converted. Otherwise, you will have to join the missing pieces later on.
- 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.

1. To convert any logical-disk format disks to virtual-disk format, you can do it as the superuser via the **sysadm** sequence:

```
Device -> Disk -> Physical -> Convert
```

Sysadm will prompt for disk(s) to convert. As always, if a list of valid responses is not on display, enter **?** to display one.

Sysadm prompts

```
Conversion target format: [Virtual disk format]
```

2. To convert to virtual disk format, press Enter or enter **Virtual**; an alternative is **Logical**. For example,

```
Conversion target format: [Virtual disk format] ↵
Forcefulness: [Careful]
```

This prompt pertains to logical/virtual disks that span more than one physical disk. If a logical/virtual disk piece exists on a disk that is offline, the default mode, **Careful**, will tell **sysadm** to abort the conversion. An alternative, **Forceful**, will tell **sysadm** to convert the existing pieces, which will result in an incomplete logical/virtual disk. Use **Forceful** only if the disks with the missing pieces are not available. You can always specify **No–write**, which will have **sysadm** check for such problems and write nothing; if you choose this, you will later need to use one of the other choices. Answer, for example,

```
Forcefulness: [Careful] ↵
Physical disk(s): all
```

3. To convert the logical disks on all physical disks, take the default. Or specify the physical disks one by one.

Where to go next

The following chapter explains how to add user accounts. If your DG/UX installation is new and does not have accounts defined for all users, continue to the next chapter. Otherwise, you might want to add one or more new mass storage devices (Chapter 9), proceed and build a new DG/UX kernel (Chapter 10) or do something else as defined in the task table included in the preface.

End of Chapter

4 Adding user accounts

This chapter provides instructions for establishing accounts (login privileges and home directories) for each user on the DG/UX system. Major sections proceed as follows.

- Completing the user account worksheet
- Creating default login parameters
- Creating a user group
- Creating a user account
- Changing a password
- Where to go next

Completing the user account worksheet

Use the user account worksheet following in Figure 4–1 (and copies, if needed) to record information about the users on your system. To complete the worksheet, you need login names, group names, and each user's preferred shell.

User Name (Login Name)	Group Name	Shell
Example: johnson	*general*	*/usr/bin/csh*

Figure 4–1 User Account Planning Worksheet

Creating default login parameters

To create a default set of login account parameters, follow these steps.

1. Follow this path through **sysadm.**

```
User -> Login Account -> Defaults-> Set
```

Sysadm guides you through a series of prompts. At each prompt, press Enter to accept the displayed default, or type a new value. The first prompt is

```
Group: [general]
```

2. The set of permissions for every file includes three parts: user access, group access, and other access. Each user in the group can access the file even if by individual login name he or she could not do so. The group name you specify here identifies a user as part of a group. The group name here must already exist. The default group, **general**, is included with DG/UX. If the group name you want does not exist, first create it, as described in the next section, and then return to this step.

The group name is limited to 32 alphanumeric characters; the first one must be alphabetic. However, we suggest shorter group names: eight characters or fewer. Specify a group name that is unique within the NIS domain. For more information on establishing a group, see *Managing the DG / UX™ System*. For example,

```
Group: [general] ↲
Base Directory: [/home]
```

3. The base directory is the parent directory of all users' home directories. The base directory must exist, and you must specify its full path from the root. Check your Virtual Disk Planning Worksheets that you completed in Chapter 2 or Appendix D for the mount point directory for users' home directories. For example,

```
Base Directory: [/home] ↲
Skeleton Directory: [/etc/skel]
```

4. The skeleton directory contains the prototypes of startup files such as **.profile**, **.login**, and **.cshrc** files for each user's home directory. The default directory, **/etc/skel**, is included with DG/UX. For example,

```
Skeleton Directory: [/etc/skel] ↲
Shell Program: [/sbin/sh]
```

5. Several shells (command-line interpreters) are available from which you can operate after you log in to the DG/UX system. Enter **/sbin/sh** to select the Bourne shell (the default), **/usr/bin/ksh** to select the Korn shell, or **/usr/bin/csh** to select the C shell. For more information on shells, see *Using the DG/UX™ System*. For example,

```
Shell Program: [/sbin/sh] /usr/bin/csh ↵
OK to perform operation? [yes]
```

6. Review the answers given and if they are correct, press Enter. If not, enter No and respecify as needed.

   You have created a set of default account parameters.

Creating a user group

Membership of users in a group grants them certain file and directory privileges while excluding other users' access. A user can belong to multiple groups; however, the user's shell is associated with only one group at a time. To create a user group, follow these steps.

1. Follow this path through **sysadm**.

```
User -> Group -> Add
```

   Sysadm displays an informative prompt, depending on whether your host is the NIS master, and requests the group name. For example,

```
This host is not the NIS master. Only the local
       Group database will be used.
Group Name:
```

2. The group name can contain up to 32 alphanumeric characters; the first must be a letter. However, we suggest shorter group names: eight characters or fewer. Specify a group name that is unique within the NIS domain. For more information on establishing a group, see *Managing the DG/UX™ System*. For example,

```
Group Name: testers ↵
Group ID: [101]
```

3. The group ID (or GID), which is a number from 0 to 60,000. Numbers less than 100 are reserved for system use. The GID also must be unique. As a default, **sysadm** supplies the highest assigned GID plus one. For example,

```
Group ID: [101] 101 ↵
Group Members:
```

4. As group members, you can specify the login names of users who will be members of the group. You need not specify any login names now; when you add user accounts later, you can specify group names. If you do specify any login names, they must already exist as entries in your local **passwd** file, or in the global NIS database, whichever applies. To separate login names, use a comma. For example,

   ```
   Group Members: johnson, dupree ↵
   OK to perform operation?  [yes]
   ```

5. Review the answers given and if they are correct, press Enter. If not, enter no and respecify as needed.

   You have created a user group.

Creating a user account

A user login account establishes each user's working environment. As default answers, **sysadm** displays those specified when you created a default set of login account parameters (if you did so). To create a user account, follow these steps.

1. Follow this path through **sysadm.**

   ```
   User -> Login Account -> Add
   ```

   Sysadm displays an informative prompt, depending on whether your host is the NIS master, and requests the group name. For example,

   ```
   This host is not the NIS master. Only the local
      Password database will be used.
   Login Name:
   ```

   By default, your machine is set up to be an NIS client, not an NIS master, thus the local password database is used. If an OS server is the NIS master, the **yppasswd** database is used, instead of the local password database. Refer to *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for more information on NIS.

2. Each login name must be unique. It can contain a combination of these characters: **a-z 0-9 -** and _ . The name should not exceed 8 characters, and it must begin with a lowercase letter. For example,

   ```
   Login Name: johnson ↵
   Add password aging for the login account? [no]
   ```

3. Password aging is a security feature that lets you control how long a password may be in effect before the user must change it. If you select this feature, you will define the aging period later. The default aging period is 4 weeks. If system security is very important to you, we recommend this feature. It requires the user to create, specify, and remember a new password periodically. For example,

   ```
   Add password aging for the login account? [no] yes ↵
   User ID: [n]
   ```

4. The user ID is a number in the range 0 to 60000 assigned sequentially. Internally, the system uses this user ID, not the login name to identify the user. ID numbers 0 to 100 are reserved for system use. As a default, **sysadm** supplies the next unassigned ID, which is one number higher than the last user ID. For example,

   ```
   User ID: [16512] ↵
   Group: [general]
   ```

5. Membership of users in a group grants them certain file and directory privileges while excluding other users' access. A user can belong to multiple groups; however, the user's shell is associated with only one group at a time.

   The group name must already exist. To create a user group, see "Creating a user group" earlier in this chapter. For example,

   ```
   Group: [general] testers ↵
   Home Directory: [/home/johnson]
   ```

6. The default home directory name is based on the Base directory you specified when you created a default set of login account parameters (if you did so). The system will create this directory as it creates the account. If you want a name other than the default, specify the full path from the root; otherwise, take the default. For example,

   ```
   Home Directory: [/home/johnson] ↵
   User Comment:
   ```

7. As a user comment, you might enter the user's full name, the date, and perhaps the user's telephone number. For example,

   ```
   User Comment: Samuel Johnson, 3/13/94, 555-9667 ↵
   Shell Program: [/sbin/xxx]
   ```

8. Several shells (command-line interpreters) are available from which you can operate after you log in to the DG/UX system. Take the default, or enter **/sbin/sh** to select the Bourne shell, **/usr/bin/ksh** to select the Korn shell, or **/usr/bin/csh** to select the C shell. For more information on shells, see *Using the DG/UX™ System*. For example,

   ```
   Shell Program: [/sbin/csh] ↵
   Create home directory for johnson? [no]
   ```

9. Generally, you should have **sysadm** create the home directory. If you don't do so, the user will login to the root directory. For example,

```
Create home directory for johnson? [no] yes↵
Set an initial password for johnson? [yes]
```

10. We recommend a Yes answer to this prompt. If you answer No, the user will not be able to login until you set a password (which you can do with **sysadm** or at the command **passwd** *login–name*, For example,

```
Set an initial password for johnson? [yes] ↵
```

Now, If you answered **yes** to the prompt for password aging, continue with the next step. If you said no to password aging, skip to step 13.

```
Minimum number of weeks before johnson may
   change the password: (0-64) [0]
```

11. This query sets the minimum number of weeks that must pass before the user may change his or her password. Password changing will fail if the user attempts to do before this period has elapsed. The default, 0, lets the user change password immediately at login, which is generally desirable. So you might take the default:

```
Minimum number of weeks before johnson may
   change the password: (0-64) [0] ↵
Maximum number of weeks until johnson
   must change the password: (0-64) [4]
```

12. This query sets the maximum number of weeks that may pass before the user must change his/her password. If the time expires before the user changes the password, the system will prompt the user for a new password at the next login. Four weeks is a good general purpose default. To take the default, press Enter:

```
Maximum number of weeks until johnson
   must change the password: (0-64) [4] ↵
```

13. **Sysadm** asks for confirmation:

```
OK to perform operation?  [yes]
```

14. Review the answers given and if they are correct, press Enter. If not, enter no and respecify as needed. For example,

```
OK to perform operation?  [yes] ↵
User 'johnson' has been added.
New password:
```

15. The password you specify will become effective immediately; you will need to tell it to the user so that he or she can login. If you specified a nondefault answer for password aging (steps 11 or 12), this password is only temporary; the system will ask the user to change it at the first login. You can leave the password blank; if you do this, the user can log in without a password (and may later set a password with the **passwd** command).

    For convenience, you might choose the user's login name. If you do this, make sure to have the user change password at first login. For example,

    ```
    New password: johnson ↵  (Password does not echo as you type.)
    Re-enter new password:
    ```

16. Respecify the password you entered in the previous step. If you make a mistake, you will need to respecify the password from the beginning. For example,

    ```
    Re-enter new password: johnson ↵  (Password does not echo.)
    ```

    For every user you want to add, select Add and repeat steps 2 through 16. When finished, return to the **sysadm** Main Menu by entering **^**.

Changing a password

Each user should change the temporary password immediately to ensure system security. Use these guidelines to change a password:

- Your password must be different from your login name.
- Your password should not be any obvious rearrangement of the characters in your login name. For example, if your login name is **anemone**, do not use the password **nemonea**.
- The new password must have at least six characters. At least two characters should be upper- or lowercase alphabetic characters (a-z and A-Z), and at least one character should also be a numeric or special character, such as 0 through 9, ?, !, @, $, or space.

To assign a new user password, log in with the user's login name.

At the shell prompt, type either the **passwd** or **yppasswd** command followed by the user's login name and the Enter key. If you installed the NIS package, use **yppasswd**; otherwise, type **passwd** followed by Enter. You type the old password followed by the new password, which you are asked to confirm. If you type the password correctly both times, the passwords match and the shell prompt appears. To promote security, the password does not appear on the screen as you type.

A typical dialog for changing the temporary password follows:

```
$ yppasswd ↵
Changing yp password for sara
Old yp password: sara45 ↵     (sara45 is the old password; it does
                              not echo as you type.)
New password: sarac6 ↵        (sarac6 is the new password;
                              it does not echo as you type.)
Retype new password:sarac6↵ (Does not echo.)
yellow pages passwd changed on nis-master-name
```

Refer to the **yppasswd**(1) or **passwd**(1) manual page for more information.

Where to go next

The following chapter explains how to add terminals. If your DG/UX installation is new and does not have terminals defined for all its asynchronous lines, continue to the next chapter. Otherwise, you might want to add one or more new mass storage devices (Chapter 9), proceed and build a new DG/UX kernel (Chapter 10) or do something else as defined in the task table included in the Preface.

End of Chapter

5 Adding terminals

This chapter explains how to set up terminals to operate in the DG/UX system environment. It assumes the terminal hardware has been installed.

IMPORTANT: This chapter emphasizes terminals, but printers and modems also can be connected to terminal line controllers. Printers and modems are covered in separate manuals: *Installing and Configuring Printers on the DG / UX™ System* and *Managing Modems and UUCP on the DG / UX™ System*. In this chapter, the term *port* applies to both terminals and modems.

In this chapter, the term *terminal line controller* means the RS-232/422 interfaces for the ports on the computer unit, the VAC/16 controller, and the VDA/128 and VDA/255 host adapters. The term *ports* on a VDA host adapter actually refers to the ports on the cluster controllers which are connected to the host adapter.

Major sections in this chapter proceed as follows.

- Port tty line numbers
- Completing terminal controller worksheets
- Determining the tty line for terminal line controller ports
- Adding terminals
- Creating a port monitor to manage groups of terminals
- Listing terminals, port monitors, and port services
- Where to go next

Port tty line numbers

The DG/UX system automatically assigns a tty line number to each attached port in the hardware configuration when the system boots. A tty line number takes the form:

```
tty*x*
```

where *x* is a sequentially assigned number. For example, **tty00** refers to the first port, **tty01** the second port, and so on. A file with the name of the tty line number is created in the **/dev** directory each time the system boots.

If you have ports attached to multiple, different terminal line controllers, you must determine each port's tty line number — as explained next.

You can add terminals (and printers) to your DG/UX system all at once or one at a time. If you add them all at once, you assign the same characteristics to all asynchronous ports on your computer. In other words, the RS-232/422 ports on the computer unit, the ports on any Systech asynchronous controllers (VAC/16 controllers), and the ports on the cluster controllers for any Systech asynchronous distributed host adapters (VDA/128 and VDA/255 host adapters) would all have the same characteristics. If you add terminals one at a time, you may assign different characteristics to different ports. The actual steps of adding terminals using **sysadm** are explained toward the end of this chapter.

To add a terminal to your DG/UX system, you must know

- The tty line number that the DG/UX system assigned to the terminal line controller port where the terminal is connected. This number depends on the terminal controller, number and type of previous terminal controllers, and port number on the controller. To learn it, you may need to perform the following tasks:

  a. Finding the order of controller names in your system file.

  b. Learning the terminal line controller type and cluster controller type

  c. Learning the port where each device is connected

  After learning the tty line number(s) for each terminal controller, you will complete a worksheet so you won't need to perform these tasks again later.

- The type of terminal connected to the port (for example, VT100 terminal or PostScript printer). You will then complete another worksheet for all these devices.

  After gathering this information and completing the worksheets, you can then add the terminals themselves using **sysadm** — a relatively simple and straightforward procedure.

Completing terminal controller worksheets

This section contains samples and blank originals of the following worksheets:

- Terminal Line Controllers Worksheet
- RS/232/422 Ports on the Computer Unit Worksheet
- VAC/16 Controller Worksheet
- VDA Host Adapter Worksheet

Each VDA Host Adapter Worksheet contains space for recording 32 ports. So, if you have a VDA host adapter, you will need two additional sheets for a VDA/128 host adapter and six additional sheets for a VDA/255 host adapter.

Preceding each worksheet is a sample worksheet that has been filled out for an AViiON 5000 computer with three terminal line controllers: RS-232/422 terminal/modem port on the computer unit, one VAC/16 controller, and one VDA/128 host adapter. The host adapter has one 8-line cluster controller and seven 16-line cluster controllers, providing ports for a maximum of 120 serial devices. These terminal line controllers have the device names given below, and these device names are listed in your system file in the relative order shown below:

duart( )	terminal/modem port on computer unit
syac( )	VAC/16
syac(1)	VDA/128

Figures 5–1 and 5–2 show a sample terminal line controllers worksheet and a copy for you to complete.

Figures 5–3 and 5–4 show a sample RS-232/422 ports on computer unit worksheet and a copy for you to complete.

Figures 5–5 and 5–6 show a sample VAC/16 controller worksheet and a copy for you to complete.

Figures 5–7 and 5–8 show a sample VDA host adapter worksheet and a copy for you to complete.

Appendix D contains additional copies of the blank worksheets.

Sample Worksheet

Terminal Line Controllers Worksheet

Board No.	Device Name	Configuration File Position	Board or Port Type	Cluster Controllers			
				Address	No. Lines	Address	No. Lines
	duart()	1					
	duart(1)						
0	syac()	2	*VAC / 16*	01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
1	syac(1)	3	*VDA / 128*	01	*8*	09	
				02	*16*	0A	
				03	*16*	0B	
				04	*16*	0C	
				05	*16*	0D	
				06	*16*	0E	
				07	*16*	0F	
				08	*16*	10	
2	syac(2)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
3	syac(3)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
4	syac(4)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	

Figure 5–1 Sample Worksheet for Terminal Line Controllers

Terminal Line Controllers Worksheet

Board No.	Device Name	Configuration File Position	Board or Port Type	Cluster Controllers			
				Address	No. Lines	Address	No. Lines
	duart()	1					
	duart(1)						
0	syac()	2		01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
1	syac(1)	3		01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
2	syac(2)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
3	syac(3)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
4	syac(4)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	

Figure 5–2 Worksheet forTerminal Line Controllers

Sample Worksheet

Computer Unit RS-232/422 Port Worksheet

Port type:	*terminal / modem*	**Device name:** *duart()*
tty Line	**Device Type**	**Description**
oo	*modem to VT100*	*lab B2, conn 2203*
Port type:		**Device name:** *duart(1)*
tty Line	**Device Type**	**Description**

Figure 5–3 Sample Worksheet for RS-232/422 Ports On Computer Unit

Computer Unit RS-232/422 Port Worksheet

Port type:		**Device name:** *duart()*
tty Line	**Device Type**	**Description**
Port type:		**Device name:** *duart(1)*
tty Line	**Device Type**	**Description**

Figure 5–4 Worksheet for RS-232/422 Ports On Computer Unit

Sample Worksheet

VAC/16 Controller Worksheet

Board no: *0* **Device name:** *syac()* **Range of tty lines:** *01 –16*

Port No.	tty Line	Device Type	Description	Port No.	tty Line	Device Type	Description
0	*01*	*4558 printer*	*officeA1 conn 1100*	**8**	*09*	*D216+*	*officeA10 conn 1118*
1	*02*	*Epson printer*	*officeA2 conn 1102*	**9**	*10*	*VT100*	*officeA11 conn 1120*
2	*03*	*D216+*	*officeA3 conn 1104*	**10**	*11*	*D462+*	*officeA12 conn 1122*
3	*04*	*D462+*	*officeA4 conn 1106*	**11**	*12*	*D462+*	*officeA14 conn 1124*
4	*05*	*D216+*	*officeA5 conn 1108*	**12**	*13*	*D462+*	*officeA14 conn 1124*
5	*06*	*D216+*	*officeA6 conn 1110*	**13**	*14*	*D462+*	*officeA18 conn 1128*
6	*07*	*D216+*	*officeA5 conn 1114*	**14**	*15*	*D462+*	*officeA20 conn 1130*
7	*08*	*D216+*	*officeA5 conn 1116*	**15**	*16*	*D462+*	*officeA21 conn 1132*

Figure 5–5 Sample Worksheet for a VAC/16 Controller

VAC/16 Controller Worksheet

Board no: *0* **Device name:** *syac()* **Range of tty lines:** *01 –16*

Port No.	tty Line	Device Type	Description	Port No.	tty Line	Device Type	Description
0				**8**			
1				**9**			
2				**10**			
3				**11**			
4				**12**			
5				**13**			
6				**14**			
7				**15**			

Figure 5–6 Worksheet for a VAC/16 Controller

Sample Worksheet

VDA Host Adapter Worksheet

Sheet *1* of *4*

Board type: *VDA / 128* Board no: *1* Device name: *syac(1)* Range of tty lines: *17–271*									
Cluster Address	**Port No.**	**tty Line**	**Device Type**	**Description**	**Cluster Address**	**Port No.**	**tty Line**	**Device Type**	**Description**
01	0	*17*	*D216+*	*office B1,conn 1200*	*02*	0	*33*	*VT100*	*office B9,conn 1216*
|	1	*18*	*D216+*	*office B2,conn 1202*	|	1	*34*	*VT100*	*office B10,conn 1218*
|	2	*19*	*D462+*	*office B3,conn 1204*	|	[illegible]	*35*	*D216+*	*office B11,conn 1220*
|	3	*20*	*VT100*	*office B4,conn 1206*	|	3	*36*	*D462+*	*office B10,conn 1222*
|	4	*21*	*VT100*	*office B5,conn 1208*	|	4	*37*	*D462+*	*office B13,conn 1224*
|	5	*22*	*VT100*	*office B6,conn 1210*	|	5	*38*	*VT100*	*office B14,conn 1226*
|	6	*23*	*D216+*	*office B7,conn 1212*	|	6	*39*	*D462+*	*office B15,conn 1228*
|	7	*24*			|	7	*40*	*D462+*	*office B16,conn 1230*
↓	8	*25*	*6772ptr*	*lab B2,conn 2204*	|	8	*41*	*D462+*	*office B17,conn 1232*
	9				|	9	*42*	*D462+*	*office B18,conn 1234*
	10				|	10	*43*	*D462+*	*office B19,conn 1236*
	11				|	11	*44*	*D462+*	*office B20,conn 1238*
	12				|	12	*45*	*D462+*	*office B21,conn 1240*
	13				|	13	*46*	*D462+*	*office B22,conn 1242*
	14				|	14	*47*	*D462+*	
	15				↓	15	*48*	*D462+*	

Figure 5–7 Sample Worksheet for a VDA Host Adapter

VDA Host Adapter Worksheet
Sheet _____ of _____

Board type: Board no: Device name Range of tty lines:									
Cluster Address	Port No.	tty Line	Device Type	Description	Cluster Address	Port No.	tty Line	Device Type	Description
	0					0			
	1					1			
	2					2			
	3					3			
	4					4			
	5					5			
	6					6			
	7					7			
	8					8			
	9					9			
	10					10			
	11					11			
	12					12			
	13					13			
	14					14			
	15					15			

Figure 5–8 Worksheet for a VDA Host Adapter

Finding the order of controller names in your system file

You will need to know the relative order of the terminal line controller device names in your system file.

Change to the current directory of the system file and use the **more** command to view its contents. An example follows:

```
# cd /usr/src/uts/aviion/Build ↵
# more system.aviion ↵
```

IMPORTANT: Your system file may have a name other than **system.aviion**. View the appropriate file. Refer to Chapter 10 for information on building kernels and editing system files.

Figure 5–9 shows an excerpt from a system file.

```
#-------------------------------------------------------------
# Automatically Configured Hardware Devices:
#
# These hardware devices were found on the system by probedev(1M).
#
#    kbd()           ## Workstation keyboard
     grfx()          ## Workstation graphics display
     lp()            ## Integrated parallel line printer controller
     duart(0)        ## Dual-line terminal controller (number 0)
     inen()          ## Integrated Ethernet controller
     syac()          ## Systech terminal line controller
     syac(1)         ## second Systech terminal line controller
     sd(insc(),0)    ## SCSI disk 0 on Integrated SCSI adapter
     sd(insc(),1)    ## SCSI disk 1 on Integrated SCSI adapter
     st(insc(),4)    ## SCSI tape 4 on Integrated SCSI adapter
     st(insc(),6)    ## SCSI tape 6 on Integrated SCSI adapter
```

Figure 5–9 Automatically Configured Devices in the System Configuration File

Viewing the system file, you can see the relative order of the terminal line controllers. The ordering follows:

duart(0)	position 1
syac()	position 2
syac(1)	position 3

The order in which terminal line controllers are configured and listed in the system file is important because it affects the assignment of tty line numbers.

Learning the terminal line controller type and cluster controller type

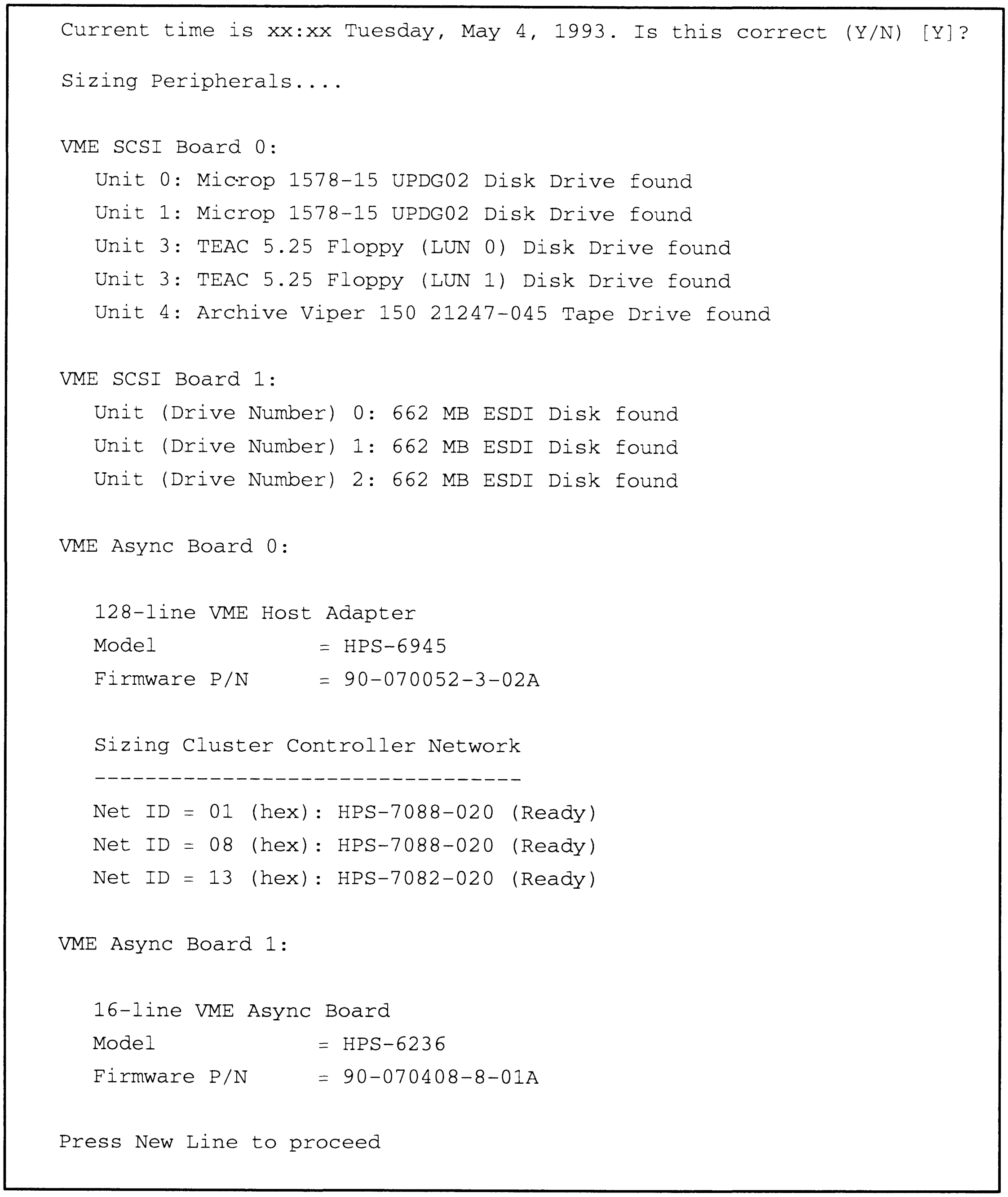
This section describes how to use your AViiON System Diagnostics to help you get the following information:

- Board type (VAC/16, VDA/128, or VDA/255) for the **syac** terminal line controller device name.
- Cluster controller type (8-line, 16-line) for each controller connected to a VDA host adapter.
- VAC/16 or cluster controller port to which a specific terminal is connected.

IMPORTANT: The person installing your computer hardware should have recorded this information on Device Worksheets that were supplied in the manual *Setting Up and Installing VMEbus Options in AViiON® Systems*. If these worksheets are available from the hardware installer and they have all the information listed above, then go to the section "Determining the tty line for terminal line controller ports" later in this chapter. If these worksheets are not available or they are missing information, continue with this section.

To determine the terminal line controller type and cluster controller type, perform the following steps.

1. Make sure your DG/UX system is shut down. If the SCM prompt is showing on the screen, the DG/UX system is shut down. If the DG/UX system is running, shut it down as in Chapter 11, section "Shutting down the DG/UX system."
2. Start the AViiON system diagnostics as explained in the manual *Using AViiON® Diagnostics and the AV/Alert™ Diagnostic Support*, Chapter 5.
3. The diagnostics program initializes the components that it found and displays information. After you verify that the date displayed is correct, the diagnostic program displays lists the peripheral devices connected to initialized controllers, as shown in Figure 5–10 following.



```
Current time is xx:xx Tuesday, May 4, 1993. Is this correct (Y/N) [Y]?

Sizing Peripherals....

VME SCSI Board 0:
   Unit 0: Microp 1578-15 UPDG02 Disk Drive found
   Unit 1: Microp 1578-15 UPDG02 Disk Drive found
   Unit 3: TEAC 5.25 Floppy (LUN 0) Disk Drive found
   Unit 3: TEAC 5.25 Floppy (LUN 1) Disk Drive found
   Unit 4: Archive Viper 150 21247-045 Tape Drive found

VME SCSI Board 1:
   Unit (Drive Number) 0: 662 MB ESDI Disk found
   Unit (Drive Number) 1: 662 MB ESDI Disk found
   Unit (Drive Number) 2: 662 MB ESDI Disk found

VME Async Board 0:

   128-line VME Host Adapter
   Model             = HPS-6945
   Firmware P/N      = 90-070052-3-02A

   Sizing Cluster Controller Network
   ---------------------------------
   Net ID = 01 (hex): HPS-7088-020 (Ready)
   Net ID = 08 (hex): HPS-7088-020 (Ready)
   Net ID = 13 (hex): HPS-7082-020 (Ready)

VME Async Board 1:

   16-line VME Async Board
   Model             = HPS-6236
   Firmware P/N      = 90-070408-8-01A

Press New Line to proceed
```

Figure 5–10 Sample Peripherals Display from AViiON System Diagnostics

On the Terminal Line Controllers Worksheet in Figure 5–2 earlier, record each VME async board type that was listed under "Sizing Peripherals...." The names that the diagnostics system uses for the different terminal line controller boards are as follows:

16–line Async Board	VAC/16
128–line VME Host Adapter	VDA/128
255–Line VME Host Adapter	VDA/255

For example, using the sample screen above, you would write "VDA/16" in the Board or Port Type column for Board No. 0, and "VAC/128" in the Board or Port Type column for Board No. 1.

On the Terminal Line Controllers Worksheet in Figure 5–2 earlier, record the number of lines for each cluster controller address (Net ID) listed under a VME host adapter. The model numbers for the 8-line and 16-line cluster controllers are as follows:

HPS-7082	8-line cluster controller
HPS-7088	16-line cluster controller

For example, using the sample screen above, for Board 0, under "Cluster Controllers" you would write "8" in the "No. Lines" column for Address 01, and write "16" in the "No. Lines" column for Addresses 02 and 08.

When you have finished recording this information, press Enter. If more sizing information appears, repeat these procedures appropriately. Continue this process of viewing and recording sizing information until the Main Menu appears.

Before continuing, you should transfer information from the Terminal Line Controllers Worksheet to the tty worksheets as described below. If you do not do this, you will have difficulty determining the tty line assigned to each port on these terminal line controllers.

VAC/16

For each VAC/16 controller in your computer, record the board number and its device name on a VAC/16 Controller Worksheet.

VDA host adapter

For each VDA host adapter in your computer, record its board number, device name, and device type (VDA/128 or VDA/255) on a VDA Host Adapter Worksheet. For each cluster controller connected to the host adapter, record the cluster address. If the cluster controller is an 8-line controller, draw a vertical arrow from the cluster address you entered down to the dashed line. This indicates that only nine ports are available on this controller. If the controller is a 16-line controller, draw the vertical line through the dashed line all the way down to the bottom of the column to indicate that 16 ports are available.

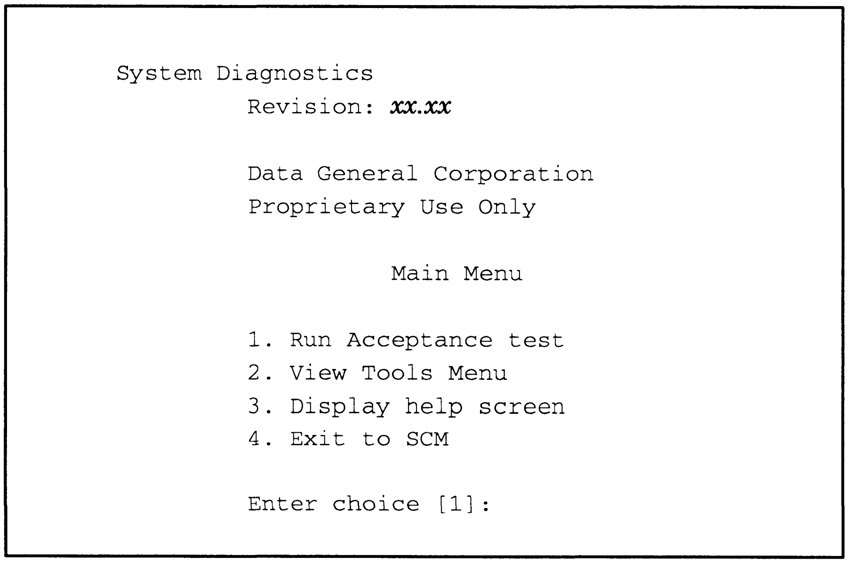
You now have enough information to determine which tty line your DG/UX system assigns to a specific port on any of the terminal line controllers in your computer. In addition, if you know what type of terminal is connected to each terminal line controller port, you can add these devices to your DG/UX system without having to complete the following section. In this case, you should record the device type for each port on the appropriate tty worksheet, Figure 5–12 later.

Next, exit to the SCM by selecting option 4 on the Main Menu, and go to the “Determining the tty lines for terminal line controller ports” section later in this chapter. If you do not know which type of terminal is connected to each terminal line controller port, continue on to the next section.

Learning the port where each device is connected

While you do not need to know the port to which a specific device is connected to determine the tty line for that port, you will need this information when you add a terminal to your DG/UX system, or if you need to troubleshoot problems with a terminal line controller. This section will help you obtain that information.

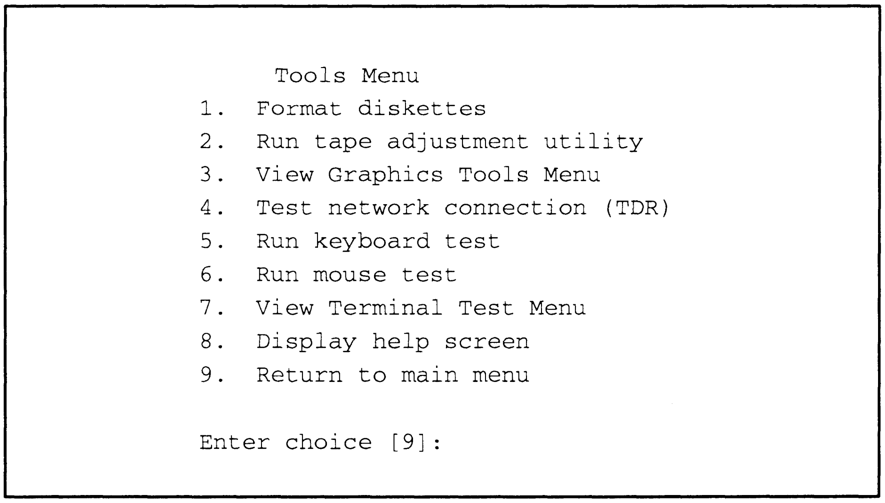
The following figure shows the System Diagnostics’ Main Menu that appears on the system console.



```
System Diagnostics
         Revision: xx.xx

         Data General Corporation
         Proprietary Use Only

                   Main Menu

         1. Run Acceptance test
         2. View Tools Menu
         3. Display help screen
         4. Exit to SCM

         Enter choice [1]:
```

1. At the Main Menu, select **2** from the View Tools Menu and press Enter.

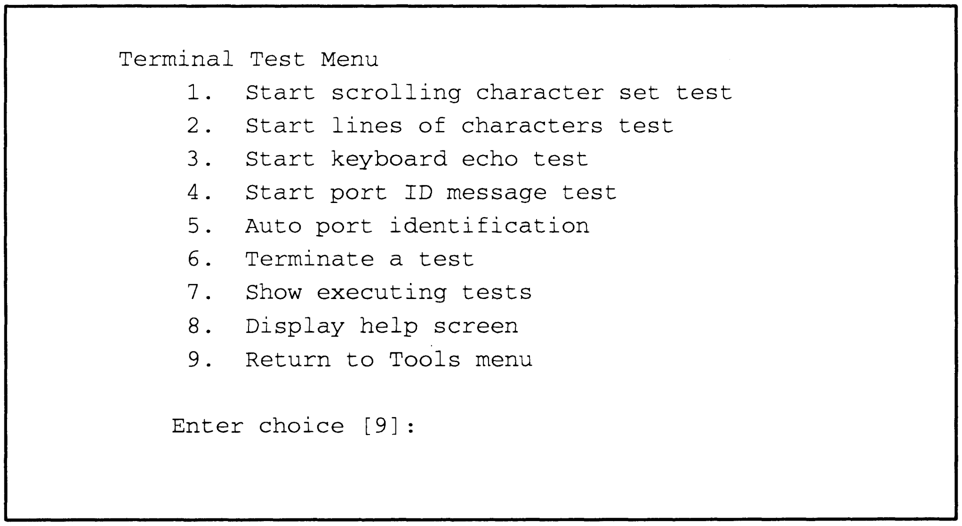
   The Tools Menu shown below appears.



```
                Tools Menu
           1.   Format diskettes
           2.   Run tape adjustment utility
           3.   View Graphics Tools Menu
           4.   Test network connection (TDR)
           5.   Run keyboard test
           6.   Run mouse test
           7.   View Terminal Test Menu
           8.   Display help screen
           9.   Return to main menu

           Enter choice [9]:
```

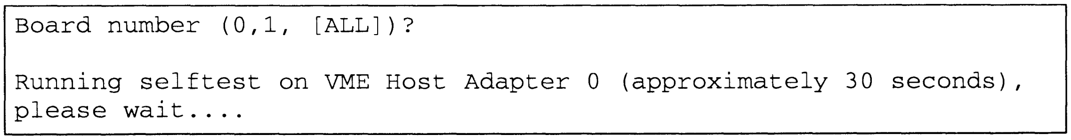
2. **Select 7 from the View Terminal Test menu and press Enter.**

   The Terminal Test Menu shown below appears.



```
Terminal Test Menu
    1.   Start scrolling character set test
    2.   Start lines of characters test
    3.   Start keyboard echo test
    4.   Start port ID message test
    5.   Auto port identification
    6.   Terminate a test
    7.   Show executing tests
    8.   Display help screen
    9.   Return to Tools menu

    Enter choice [9]:
```

3. **Select 4 from the Start port ID message test and then press Enter. The system displays the following prompt:**



```
Board number (0,1, [ALL])?

Running selftest on VME Host Adapter 0 (approximately 30 seconds),
please wait....
```

4. Press Enter.

   This message occurs the first time you select a host adapter for testing. On each terminal (or printer) connected to a port on a VAC/16 controller or VDA host adapter in your computer, you will see a port ID message similar to the one below. This message lists the board number, cluster address (VDA host adapter only), and the port number for the device displaying the message.

   ```
   128-line VME Host Adapter 0, Cluster address: 01, port: 0
   ```

IMPORTANT: If you have an 8-line cluster controller without a parallel printer connected to port 8, or the printer is not on line and ready, a message appears telling you this. If such a message appears, simply press the Esc key to skip the test on that port.

5. Look at the message displayed on each device for which you do not know the port number, and determine the board number, cluster address, and port number for the device. On the appropriate Device Worksheet earlier (Figure 5–6 or 5–8) under the specified board number, cluster address (if applicable), and port number, record the type of device (for example, D460 terminal or Model 6640 parallel laser printer) displaying the message. Also record a description that locates the device (for example, office 3B, connector #1356). While the port ID messages are being displayed, the Terminal Test Menu appears on the system console screen.
6. After you finish recording the information for each terminal, select **6** from the Terminal test menu and then press Enter.

   The following prompt is displayed:

   ```
   Board number (0,1, [ALL])?
   ```
7. Press Enter to accept the default response to the next prompt: From the Terminal Test Menu, press Enter again to select the Return to Tools menu.
8. From the Tools Menu, press Enter to select Return to Main Menu.
9. At the Main Menu, exit to the SCM by pressing **4** and Enter.

Determining the tty lines for terminal line controller ports

This section explains how the DG/UX system allocates tty lines to asynchronous terminal line controller ports. With this information, you can determine the tty lines that your DG/UX system assigns to each such port on your computer. Then you can complete the last worksheet and use **sysadm** to add the terminals.

How the DG/UX system allocates tty lines

When you booted your machine, it automatically allocated a specific tty line for each port on each terminal line controller in your computer. Table 5–1 lists the number of tty lines that the DG/UX system allocates to each type of terminal line controller.

Table 5–1 Number of tty Lines Allocated to Terminal Line Controllers

Terminal Line Controller	Lines Allocated
RS–232/422 ports on computer unit	1
VAC/16 controller	16
VDA/128, VDA/255, or VTC/256 host adapter	256

Notice that the DG/UX system allocates 256 tty lines to a VDA/128 host adapter. Since a VDA/128 host adapter has only 128 ports, this means that only the first 128 tty lines are actually assigned to specific ports on a VDA/128; the remaining 128 tty lines are unused.

The DG/UX system assigns a specific tty line to each port sequentially in the order in which the names of the terminal line controllers are listed in your system file. It starts with **tty00**. For each subsequent port, the numerical portion of the tty name is increased by one.

Let's look at an example of how the DG/UX system assigns tty lines. The system file lists the following terminal line controllers in the order shown below:

```
duart()    ## integrated Duart terminal line controller
syac()     ## first Systech terminal line controller
syac(1)    ## second Systech terminal line controller
```

Further suppose that **syac( )** is a VAC/16 controller and **syac(1)** is a VDA/128 host adapter. For this configuration, the DG/UX system assigns tty lines as follows:

tty00 to the RS-232/422 port on the duart.

tty01–**tty16** to the VAC/16 controller.

tty17—tty272 to the VDA/128 host adapter.

If the **duart** was listed after the two **syac** devices, the tty line assignment would be as follows:

tty00–tty15 to the VAC/16 controller.

tty16–tty271 to the VDA/128 host adapter.

tty272 to the RS-232/422 port on the duart.

The DG/UX system assigns each of the 16 tty lines that it allocates to a VAC/16 host adapter to its 16 ports in sequential order. In other words, in the example above where **tty01** through **tty16** are allocated to the VAC/16 controller, **tty01** is assigned to port 0, **tty02** is assigned to port 1, **tty03** is assigned to port 2, and so on.

Since devices connect to a VDA host adapter through ports on cluster controllers, the DG/UX system allocates specific tty lines to those ports. Table 5–2 shows how the DG/UX system allocates 16 tty lines to each cluster controller address, 01 through 10 hexadecimal (16 decimal). This means that the DG/UX system allocates 16 tty lines to an 8-line cluster controller with one of these addresses. Since an 8-line cluster controller has eight asynchronous ports (ports 0 through 7) and one parallel printer port (port 8), the last seven tty lines allocated to this controller's address are unused. The DG/UX system assigns the 16 tty lines that it allocates to a cluster controller address as follows: the tty line with the lowest number (call it *n*) to port 0; the next tty line with the next highest number (*n*+1) to port 1; the one with the next highest number (*n*+2) to port 2, and so on.

Table 5–2 tty Lines Allocated to Cluster Controller Addresses

Cluster Controller Address	tty Lines Allocated
01	**tty**(*n*) through **tty**(*n*+15)
02	**tty**(*n* + 16) through **tty**(*n* + 31)
03	**tty**(*n* + 32) through **tty**(*n* + 47)
04	**tty**(*n* + 48) through **tty**(*n* + 64)
05	**tty**(*n* + 64) through **tty**(*n* + 79)
06	**tty**(*n* + 80) through **tty**(*n* + 95)
07	**tty**(*n* + 96) through **tty**(*n* + 111)
08	**tty**(*n* + 112) through **tty**(*n* + 127)
09	**tty**(*n* + 128) through **tty**(*n* + 143)
0A	**tty**(*n* + 144) through **tty**(*n* + 159)
0B	**tty**(*n* + 160) through **tty**(*n* + 175)
0C	**tty**(*n* + 176) through **tty**(*n* + 191)
0D	**tty**(*n* + 192) through **tty**(*n* + 207)
0E	**tty**(*n* + 208) through **tty**(*n* + 223)
0F	**tty**(*n* + 224) through **tty**(*n* +239)
10	**tty**(*n* + 240) through **tty**(*n* + 255)

IMPORTANT: The cluster controller address is also called the node address. For more information on these addresses, refer to the *HPS Downloadable Cluster* Controller Installation Guide. The last tty line, **tty**(*n*+255), allocated to the cluster controller with address 10 is not used. If this cluster controller is a 16-line controller, this tty line is assigned port 15, so cannot be used. If it is an 8-line cluster box, this tty line is one of the seven unused tty lines allocated to the controller.

Continuing with this example, let's assume that the VDA/128 host adapter with one 8-line cluster controller and one 16-line cluster controller is allocated tty lines **tty16** through **tty270**. If the 8-line cluster controller has address 01 and the 16-line controller has address 02, then the tty lines for the ports on the controllers are as follows.

Examples:

8-line cluster controller (address 01):

tty16 through **tty24** for ports 0 through 8 (port 8 is the parallel printer port).
tty25 through **tty31** are unused.

16-line cluster controller (address 02):

tty32 through **tty47** for ports 0 through 15.
tty48 through **tty271** are unused.

You should now have enough information on your Terminal Line Controller Worksheet and your tty worksheets to determine the specific tty line that your DG/UX system assigned to each port on a terminal line controller. Using these worksheets, proceed as follows:

1. Using the Terminal Line Controller Worksheet that you completed earlier in this chapter, find the device name with Configuration File Position 1, then get the tty worksheet that has that device name written on it.

2. Determine the tty line or range of tty lines assigned to this device name using the procedure below. Note that the AViiON 4600 series machines have three asynchronous ports, which could be **tty00**, **tty01**, or **tty02**.

   If the device name is **duart()** or **duart(1)** write "00" in the tty Line column on the tty worksheet for that device name. If the device name is **syac( )**, **syac(1)**, **syac(2)**, **syac(3)**, or **syac(4)**, use the formula in Table 5–3 for the board type of that device name to calculate the range of tty lines assigned to that board (where *n* = 00), and record this range on the tty worksheet for the board.

3. Using the Terminal Line Controller Worksheet that you completed earlier in this chapter, find the device name with previous higher Configuration File Position, and get the tty worksheet that has that device name written on it.

4. Determine the tty line or range of tty lines assigned to this device as described below. Use the following value for *n*:

   n = **1** + [*highest tty line number you calculated for next* device name]

   If the device name is **duart( )** or **duart(1)**, write the value for *n* in the tty Line column on the tty worksheet for that device name. If the device name is **syac( )** , **syac(1)**, **syac(2)**, **syac(3)**, or **syac(4)**, use the formula in the next table for the board type of that device name to calculate the range of tty lines assigned to that board, and record this range on the tty worksheet for the board.

Table 5–3 Lines Allocated to Systech Terminal Controllers and Cluster Controllers

Board Type	**Range of tty Lines Allocated**
VAC/16	**tty**(*n*) through **tty**(*n* + 15)
VDA/128 or VDA/255	**tty**(*n*) through **tty**(*n* + 255)
Cluster controller with address:	
01	**tty**(*n*) through **tty**(*n*+15)
02	**tty**(*n* + 16) through **tty**(*n* + 31)
03	**tty**(*n* + 32) through **tty**(*n* + 47)
04	**tty**(*n* + 48) through **tty**(*n* + 64)
05	**tty**(*n* + 64) through **tty**(*n* + 79)
06	**tty**(*n* + 80) through **tty**(*n* + 95)
07	**tty**(*n* + 96) through **tty**(*n* + 111)
08	**tty**(*n* + 112) through **tty**(*n* + 127)
09	**tty**(*n* + 128) through **tty**(*n* + 143)
0A	**tty**(*n* + 144) through **tty**(*n* + 159)
0B	**tty**(*n* + 160) through **tty**(*n* + 175)
0C	**tty**(*n* + 176) through **tty**(*n* + 191)
0D	**tty**(*n* + 192) through **tty**(*n* + 207)
0E	**tty**(*n* + 208) through **tty**(*n* + 223)
0F	**tty**(*n* + 224) through **tty**(*n* +239)
10	**tty**(*n* + 240) through **tty**(*n* + 255)

IMPORTANT: *n* = the lowest tty line number assigned to the board type.

5. Repeat steps 3 and 4 to determine the tty line or range of tty lines assigned to any other terminal line controllers in your computer. If your computer contains a VAC/16 controller, continue to step 6; otherwise, go to step 7.

6. On the tty worksheet for each VAC/16 controller, write the appropriate tty line in the tty Line column for each port number. The lowest numbered tty line allocated to the controller is the one for port 0, the next higher numbered tty line allocated is the one for port 1, and so on.

7. If your computer contains a VDA host adapter, continue to step 8; if not, then you have finished determining the specific tty line that the DG/UX system assigns to each terminal line controller port in your system.

8. On the tty worksheet for each VDA host adapter, write the appropriate tty line number in the tty Line column for each port number for each cluster address. The lowest numbered tty line allocated to the controller is the one for port 0 on the cluster

controller with address 01, and the next higher numbered tty line allocated is the one for port 1 on the same cluster controller, and so on. Since the DG/UX system assigns 16 tty lines to each cluster controller regardless of the number of ports it has, the seven highest tty lines assigned to an 8-line cluster controller are not used. To determine the tty line for port 0 on successive cluster controllers, use the formulas in the preceding table.

You have finished determining the specific tty line that the DG/UX system assigns to each terminal line controller port in your system.

Adding terminals with sysadm

Before you can use a terminal (other than your system console) that is connected to your computer either directly or indirectly through a modem and a dial-up line, you must first add information to your DG/UX system that tells how the terminal operates. In this section, we determine the operating information that DG/UX requires for a terminal, and we add this information to your DG/UX system.

The operating values that your DG/UX system requires you to specify for a terminal include the *lineset* and the *TERM variable.*

About linesets

The *lineset* establishes the speed and other line characteristics of the port to which the terminal or the modem for a terminal is connected. Further, the port line characteristics and the corresponding terminal or modem line characteristics must match. Table 5–4 lists the lineset name that the DG/UX system supports, together with their line characteristics. The DG/UX system uses **9600** as the default lineset name. For more information on linesets, refer to *Managing the DG / UX™ System.* For information about setting the line characteristics of terminals and/or modems, see the documentation for the specific device(s).

Table 5–4 Lineset Names for Terminals

Lineset Name	Baud Rate	Parity	Data Bits	Mode
Terminal				
9600	9600	None	8	ANSI
9600EP	9600	Even	7	ANSI
19200	19200	None	8	ANSI
19200EP	19200	Even	7	ANSI

IMPORTANT: Printers and modems are covered in separate manuals: *Installing and Configuring Printers on the DG / UX™ System* and *Managing Modems and UUCP on the DG / UX™ System.*)

About TERM variables

The *TERM variable* is an environmental variable that identifies the device type of a terminal connected to the computer directly, or indirectly through a modem. Programs, such as the **vi** Editor, use the TERM variable to determine the type of terminal used by a person invoking the program.

The DG/UX system uses **vt00** as the default TERM variable. If you have terminals that can operate in VT100 mode, you will find it easiest to set up these terminals to operate in this mode and use **vt100** as their TERM variable. If you have other types of terminals, you can use other TERM variables.

For a description of the TERM variables, check the **term** command as follows.

```
# man  term | more -f ↵
```

The **man** command invokes a man (manual) page; **term** is the name of the man page you want to view; **more** allows you to scroll through the man–page file each time you press you keyboard's space bar. You can use this command format later to view any electronic man page. For more information about man pages, refer to the *User's Reference for the DG / UX*™ *System* manual.

For a complete list of all the TERM variables your current DG/UX system supports, enter the **ls** command:

```
# ls -C /usr/lib/terminfo/? | more ↵
```

Then use the space bar to scroll through the list. You can return to the # prompt at any time by entering **q**.

To display the operating values defined by a particular TERM variable, enter the **infocomp** command in the form

infocmp *TERM–variable*

and substitute the particular TERM variable for *TERM–variable*.

Before adding a terminal to your DG/UX system, you should first determine its lineset and TERM variable. For a terminal connected through a modem, you need the lineset for the modem and the TERM variable for the terminal. For more information on the operating values of a terminal or modem, refer to the documentation for the device. If you cannot determine a TERM variable for a terminal, use the default value of **vt100**.

You also need to know which tty line your DG/UX system will assign or has assigned to the terminal line controller port for the terminal because the system associates the operating values with this tty line. In the DG/UX system terminology, you add information about the operating values of a terminal by "Adding a tty entry."

The earlier section "Determining the tty lines for terminal line controller ports" explains the relationship between terminal controllers and the devices they control.

Filling out the tty lines worksheet

To help you keep track of the lineset names and TERM variables for terminals and their associated tty lines, we have included a sample tty Lines Worksheet on the next page.

You should fill out the tty lines worksheet for each tty line as follows.

- If a device worksheet lists a terminal for a tty line, then on the tty lines worksheet record the tty line, the device type for the terminal, the lineset for the terminal or the modem it uses, and the TERM variable for the terminal.
- If a device worksheet lists a printer for the tty line, then on the tty lines worksheet, record the tty line and the device type for the printer. (Or use one of the printer worksheets in the manual *Installing and Configuring Printers on the DG / UX™ System.*)
- If a device worksheet does not list a device for a tty line, then on the tty lines worksheet, record the tty line and write UNUSED beside it.

The sample tty lines worksheet follows in Figure 5–11; the worksheet for you to complete follows in Figure 5–12.

Sample tty Lines Worksheet — Sheet ___ of ___

tty Line	Device Type	Lineset or Model	TERM Variable or Printer Name	tty Line	Device Type	Lineset or Model	TERM Variable or Printer Name
00	*modem*	*M2400*	*vt100*	*24*	*UNUSED*		
01	*serial printer*	*async 9600*	*lp1*	*25*	*6640 printer*	*parallel*	*async_9600*
02	*serial*	*async 9600*	*lp2*	*33*	*VT100*	*9600*	*vt100*
03	*D216+*	*9600*	*vt100*	*34*	*VT100*	*9600*	*vt100*
04	*D462+*	*9600*	*vt100*	*35*	*D216+*	*9600*	*vt100*
05	*VT100*	*9600*	*vt100*	*36*	*D462+*	*9600*	*vt100*
06	*VT100*	*9600*	*vt100*	*37*	*D462+*	*9600*	*vt100*
07	*D216+*	*9600*	*vt100*	*38*	*VT100*	*9600*	*vt100*
08	*D462+*	*9600*	*vt100*	*39*	*VT100*	*9600*	*vt100*
09	*D216+*	*9600*	*vt100*	*40*	*D462+*	*9600*	*vt100*
10	*VT100*	*9600*	*vt100*	*41*	*D462+*	*9600*	*vt100*
11	*D462+*	*9600*	*vt100*	*42*	*D462+*	*9600*	*vt100*
12	*D462+*	*9600*	*vt100*	*43*	*D462+*	*9600*	*vt100*
13	*D462+*	*9600*	*vt100*	*44*	*D462+*	*9600*	*vt100*
14	*D413*	*9600*	*vt100*	*45*	*D462+*	*9600*	*vt100*
15	*D413*	*9600*	*vt100*	*46*	*D413*	*9600*	*vt100*
16	*D413*	*9600*	*vt100*	*47*	*UNUSED*		
17	*D216+*	*9600*	*vt100*	*48*	*UNUSED*		
18	*D216+*	*9600 vt100*					
19	*D462+*	*9600 vt100*					
20	*VT100*	*9600 vt100*					
21	*VT100*	*9600 vt100*					
22	*VT100*	*9600 vt100*					
23	*D216+*	*9600 vt100*					

Figure 5–11 Sample tty Lines Worksheet

tty Lines Worksheet — Sheet ___ of ___

tty Line	Device Type	Lineset or Model	TERM Variable or Printer Name	tty Line	Device Type	Lineset or Model	TERM Variable or Printer Name

Figure 5–12 tty Lines Worksheet

After completing or modifying the tty lines worksheet, you can use **sysadm** to add the terminals. Continue to the appropriate section: Adding a single terminal or Adding a group of identical terminals.

Adding a single terminal

To add a single terminal to a new or existing hardware configuration, follow these steps.

1. Follow this path through **sysadm**:

```
Device -> Port -> Terminal -> Add
```

Sysadm guides you through a series of prompts. At each prompt, press Enter (↵)to accept the displayed default, or type a new value. The first prompt asks for the tty device number of the terminal you are adding.

```
Tty device(s):
```

2. Enter **?** to display your choices:

```
Tty device(s): ? ↵
  1 tty00
  2 tty01
  ...
```

3. Check the display against the planning worksheets that you completed and decide on the number you want to add. For example, to add tty07 you can enter either the tty07 entry number or tty07:

```
Tty device(s): tty07 ↵
TTY Definition Label: [9600]
```

4. The TTY definition label selects the **/etc/ttydefs** file to use for setting the initial terminal I/O settings. The **ttydefs** labels appear as the first field in each line of the **/etc/ttydefs** file. For example, a typical asynchronous terminal label is **9600** (the default). If 9600 is not the label you want, enter **?** to display your choices.

IMPORTANT: If you attach the terminal to a VME terminal controller, be sure you select the tty definition label M9600.

For example, to select the default,

```
TTY Definition Label: [9600] ↵
TERM Variable:
```

5. The TERM variable identifies your terminal type. This value must correspond to a terminal definition in the **/usr/lib/terminfo** database. From the shell, you can see the entire list of **terminfo** definitions by entering **ls -RC /usr/lib/terminfo/* | more.**

   Your terminal hardware documentation should specify the correct terminal type. See Appendix C for valid Data General terminal TERM variable settings. For example, you might specify vt100:

   ```
   TERM Variable: vt100 ↲
   Disabled response message:
   ```

6. This Disabled prompt lets you specify a message to be displayed when users attempt to log in to a disabled port. You might type, for example,

   ```
   Disabled response message: This terminal is disabled. ↲
   Initial state: [ENABLED]
   ```

7. This prompt enables or disables terminal port service when the terminal is added. Press Enter to accept the default, ENABLED.

   ```
   Initial state: [ENABLED] ↲
   OK to perform operation? [yes]
   ```

8. Review your answers and, if they are correct, press Enter. If not, enter no and respecify any wrong answers. For example,

   ```
   OK to perform operation? [yes] ↲
   Terminal /dev/tty07 have been added.
   ```

   If this is the first terminal added to your system, you will also see messages about the default port monitor **ttymon1**. A port monitor configures and controls access to terminals. If no port monitor exists when a terminal is added, the system assigns the default port monitor named **ttymon1** to manage the terminal. (The **ttymon** port monitor replaces the **getty** and **uugetty** programs from pre-5.4 releases of the DG/UX system.)

   You have added a terminal. If you have any other terminal to add, select add and repeat steps 2 though 8 in this section. If you have a group of identical terminals to add, continue to the next section.

Adding a group of identical terminals

You can add a group of terminals having the same tty definition label and terminal type at one time. The label comes from the **/etc/ttydefs** file, which establishes initial terminal I/O settings. The labels appear as the first field in each line of the **ttydefs** file. For example, you could add ten terminals having an asynchronous terminal label of **9600** and a **vt100** terminal type.

If you have more than 128 terminals in your configuration, you may want to divide them among several port monitors. Go to the next section, “Creating a port monitor to manage groups of terminals,” for information on creating multiple port monitors to which you can add the desired groups.

If each terminal in a group has a different definition label and terminal type, you must add the terminals one at a time. Follow the instructions in the preceding section.

To add a group of identical terminals to a new or existing hardware configuration, follow these steps.

1. Follow this path through **sysadm**:

   ```
   Device -> Port -> Terminal -> Add
   ```

   Sysadm guides you through a series of prompts. At each prompt, press Enter (↲)to accept the displayed default, or type a new value. The first prompt asks for the tty device number of the terminal(s) you are adding.

   ```
   Tty device(s):
   ```

   A tty device number is assigned automatically to each physically attached terminal when the system is booted.

2. Enter **?** to display your choices:

   ```
   Tty device(s): ? ↲
     1 tty00
     2 tty01
     3 tty02
     ...
     10 tty09
     11 tty10
   ```

3. Check the display against the planning worksheets that you completed and decide on the terminals you want to add.

4. Select a range of tty devices by specifying a range of menu entries, separated by commas or, for a range, dashes. For example, to specify menu entries 1 through 3 and 11 (terminals tty00 through tty02 and tty10):

   ```
   Tty device(s): 1–3,11 ↲
   TTY Definition Label: [9600]
   ```

5. The TTY definition label selects the **/etc/ttydefs** file to use for setting the initial terminal I/O settings. The **ttydefs** labels appear as the first field in each line of the **/etc/ttydefs** file. For example, a typical asynchronous terminal label is **9600** (the default). If 9600 is not the label you want, enter **?** to display your choices.

IMPORTANT: If you attach the terminal to a VME terminal controller, be sure you select the tty definition label M9600.

For example, to select the default,

```
TTY Definition Label: [9600] ↵
TERM Variable:
```

6. The TERM variable identifies the terminal type. This value must correspond to a terminal definition in the **/usr/lib/terminfo** database. From the shell, you can see the entire list of **terminfo** definitions by entering **ls -RC /usr/lib/terminfo/* | more**.

   Your terminal hardware documentation should specify the correct terminal type. See Appendix C for valid Data General terminal TERM variable settings. For example, you might specify vt100:

   ```
   TERM Variable: vt100 ↵
   Disabled response message:
   ```

7. This prompt lets you specify a message to be displayed when users try to log in to a disabled port. You might type, for example,

   ```
   Disabled response message: This terminal is disabled. ↵
   Initial state: [ENABLED]
   ```

8. This prompt enables or disables terminal port service when the terminal is added. Press Enter to accept the default, `ENABLED`.

   ```
   Initial state: [ENABLED] ↵
   OK to perform operation? [yes]
   ```

9. Review your answers and, if they are correct, press Enter. If not, enter no and respecify any wrong answers. For example,

   ```
   OK to perform operation? [yes] ↵
   Terminal /dev/tty00 has been added.
   Terminal /dev/tty01 has been added.
   Terminal /dev/tty02 has been added.
   Terminal /dev/tty10 has been added.
   ```

   If this is the first terminal group added to your system, you will also see messages about the default port monitor **ttymon1**. A port monitor configures and controls access to terminals. If no port monitor exists when a terminal is added, the system assigns the default port monitor named **ttymon1** to manage the terminal.

   If more than one **ttymon** port monitor exists on the system, you are asked for the name of the desired port monitor. You can create a tty port monitor as explained in the next section.

   You have added a group of terminals. If you have any other groups to add, select add and repeat steps 2 though 9 in this section. If you have an individual terminal to add, return to the previous section.

Creating a port monitor to manage groups of terminals

Port monitors are used to configure and control access to groups of terminals. The number of terminals you attach to a port monitor depends on your particular hardware configuration and your performance requirements. The default port monitor **ttymon1** is created automatically when the first terminal is added. If you have more than 128 terminals in your configuration, you may want to divide them among several port monitors.

To add a port monitor to a new or existing hardware configuration, follow these steps.

1. Follow this path through **sysadm**:

   ```
   Device -> Port -> Port Monitor -> Add
   ```

   This operation creates a directory structure for the new port monitor and adds an entry for it in the Service Access Controller's database. You assign a port service to each terminal that is controlled by the port monitor.

   Sysadm guides you through a series of prompts. At each prompt, press Enter (↲) to accept the displayed default, or type a new value. The first prompt asks for the type of port monitor you are adding.

   ```
   Port monitor type: [ttymon]
   ```

2. You may select from two types of port monitors: **ttymon** or **listen**. If you are adding a port monitor for typical asynchronous tty lines, press Enter to accept the default **ttymon**. For a Transport Layer Interface (TLI)-based network system, enter **listen**. For example,

   ```
   Port monitor type: [ttymon] ↲
   Port monitor tag:
   ```

3. Provide a tag (name) name of up to 14 alphanumeric characters for the port monitor. For example,

   ```
   Port monitor tag: ttymon2 ↲
   Command to start port monitor: [/usr/lib/saf/ttymon]
   ```

4. The prompt sets the command to start the port monitor, and the version of the port monitor. You can usually accept the default answers of **/usr/lib/saf/ttymon** and release **1** although your port monitor name differs from the displayed program name, ttymon. For example,

   ```
   Command to start port monitor: [/usr/lib/saf/ttymon]↲
   Version number: [1] ↲
   Initial run state: [STARTED]
   ```

5. Select the initial run state for the port monitor: STARTED or STOPPED. The default choice, STARTED, means that the Service Access Controller starts the port monitor immediately each time the Service Access Controller starts at boot time. If you select STOPPED, the port monitor does not become active until you start it using **sysadm**, or the **sacadm** or **admportmonitor** commands. We recommend the default, STARTED. For example,

```
Initial run state: [STARTED] ↲
Start state: [ENABLED]
```

6. Specify the state of the port monitor when started: ENABLED or DISABLED. If you select ENABLED (the default), the port monitor begins management duties and accepts connection requests when started. If you select DISABLED, the port monitor runs but does not accept connection requests. For example,

```
Start state: [ENABLED] ↲
Restart count: (0-10) [0]
```

7. Specify the restart count, which is the number of times the Service Access Controller should attempt to restart the port monitor if it fails. If the port monitor cannot be restarted after this number of tries, it is placed in the FAILED state. For example,

```
Restart count: (0-10) [0] 5↲
File name of configuration script:
```

8. If there as a file that contains the port monitor configuration script, specify its full pathname here. When the port monitor is added, the contents of this file are copied to the port monitor configuration script file. The port monitor reads configuration files only when it starts. Changes to a script file made while the monitor is running are not effective until you restart the monitor. For more on configuration scripts, see the **doconfig**(3N) manual page.

   If you don't want to specify a configuration script, press Enter.

```
File name of configuration script: ↲
Comment:
```

9. You may supply a comment about the port monitor to be displayed when you select an operation that lists information about the port monitor. Or you can press Enter to skip the comment. For example,

```
Comment: Port monitor to control ttys on hallway12. ↲
OK to perform operation: [yes]
```

10. Review your answers and, if they are correct, press Enter. If not, enter no and respecify any wrong answers. For example,

```
OK to perform operation: [yes] ↲
Port monitor ttymon2 has been added.
```

The final message informs you that the named port monitor **ttymon2** was successfully added.

Listing terminals, port monitors, and port services

Sysadm offers choices for listing terminals, port monitors, port services, and selected variables.

Listing terminals

For a current list of terminals and selected variable settings, follow this path through **sysadm**:

```
Device -> Port -> Terminal -> List
```

A typical list follows.

```
/dev/tty00
        port monitor: ttymon1
        ttydefs label: 9600
               state: ENABLED
        TERM variable: vt100
        disabled msg: This terminal is disabled.
        comment: Tty00
```

Listing port monitors

For a current list of port monitors and selected variable settings, follow this path through **sysadm**:

```
Device -> Port -> Port Monitor -> List
```

A typical list follows.

```
PMTAG      PMTYPE    FLGS      RCNT      STATUS     COMMAND
ttymon1    ttymon    -         3         ENABLED    /usr/lib/saf/ttymon #
ttymon2    ttymon    -         0         ENABLED    /usr/lib/saf/ttymon
Port monitor to control ttys on hallway 12.
```

Listing port services

For a current list of port services and selected variable settings, follow this path through **sysadm**:

```
Device -> Port -> Port Service -> List
```

A typical list follows:

```
PMTAG          PMTYPE         SVCTAG         FLGS ID       <PMSPECIFIC>
tcp            listen         0              -    root
\x00020ACE000000000000000000000000 - c - /usr/lib/saf/nlps_server #NLPS
server
tcp            listen         lpd            -    root
\x00020203000000000000000000000000 - p - /var/spool/lp/fifos/listenBSD
#Berkley lp service
tcp            listen         lp             -    root      - - p -
/var/spool/lp/fifos/listenS5 #lp service
tcp            listen         failover       -    root
\x000252D0000000000000000000000000 - c - /usr/bin/failoverd #
tcp            listen         shareddisk     -    root
\x00025398000000000000000000000000 - c - /usr/sbin/shareddiskd #Shared
disk coordination
```

Where to go next

You have finished specifying terminals to DG/UX. You might want to try them by bringing the environment down to single user status (**init s**) and then back to init 1 or higher. Next, you might want to load additional software packages, as explained in the next chapter.

To ensure that the correct terminal line controller name(s) are listed in the system file, which is the basis for the kernel, see Chapter 10. Or to select custom local character sets for your terminals, see Appendix C.

End of Chapter

6 Loading and setting up additional packages

This chapter outlines the procedure for loading and setting up additional software packages (such as TCP/IP or X windows) you received with DG/UX. Major sections proceed as follows.

- Loading a package from its release medium
- Setting up packages
- Where to go next

Adding a package (obtained from either Data General or a third-party vendor) requires two steps: loading from the release tape, and setting up the package to run with the DG/UX system. If you received a Data General release tape and postponed the setup of any of the bundled packages (such as the DG/UX X Window system, TCP/IP, ONC, or NFS), you can follow the procedures in this chapter. However, you will need to supply some answers to prompts about TCP/IP and ONC during the **sysadm** setup dialogue. Refer to the Installation Planning Worksheets that you completed in *Installing the DG / UX™ System* for this information.

Loading a package from its release medium

If you are setting up only the DG/UX packages for an OS client, refer to Chapter 7 in this manual for instructions.

Also, if you are loading or setting up the ONC/NFS package, you must declare your computer system as an NIS master or NIS server. Refer to *Managing ONC™ / NFS® and Its Facilities on the DG / UX™ System.*

These instructions assume that the package conforms to Data General package specifications. Most vendors' packages are loaded at **/usr/opt/***package-name*, where *package-name* is the name of the package. However, the package's release notice specifies the location.

1. 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 disk, file system, and mounting phases. But if you need help with these, Chapters 2 and 3 explain them.
2. Insert the software medium, tape or CD-ROM, in its drive and make sure the drive is ready.

3. To load a package, follow this path through **sysadm**:

   ```
   Software -> Package -> Load
   ```

   If there is just one release area, **sysadm** prompts for the load device; skip to step 4. If there is more than one release area, **sysadm** prompts

   ```
   Release Area: [PRIMARY]
   ```

   You have a choice about where to install the package, in the primary area or a secondary area. If you want to install it in a secondary area (perhaps for access by an OS client that has special needs not satisfied by the primary DG/UX release), you can do so. If **sysadm** is not displaying a list of release areas, enter ? to see the list of areas; then specify the release area you want. Chapter 8 explains adding a DG/UX release in a secondary area. In most cases, you will want the release in the primary area. For example,

   ```
   Release Area: [PRIMARY] ↵
   ```

4. **Sysadm** prompts

   ```
   Release Medium: [/dev/rmt/0]
   ```

5. For a list of load devices, enter **?**; then specify the device (tape or CD-ROM drive) that holds the release medium. For example,

   ```
   Release Medium: [/dev/rmt/0]  ? ↵
   Choices are

        1   /dev/rmt/0
        2   /dev/rmt/1
        3   /dev/rmt/2
        4   /dev/pmt/0
        5   /dev/pmt/1
        6   /dev/pmt/3
        7   /dev/dsk3
   Release Medium: [/dev/rmt/0] 2 ↵
   Is /dev/xxx/n ready? [yes]
   ```

6. If the device is ready, press Enter; otherwise, answer **n**, make sure the device is ready, and press Enter. For example,

   ```
   Is /dev/xxxn ready? [yes] ↵
   Package Name(s): [all]
   ```

7. For a list of the packages on the release tape, enter ?:

   ```
   Package Name(s): [all] ?↵
   Choices are

   1 all
   2 fredware
   3 fredware.man
   4 fredware.rn
   5 fredware.int
   ```

8. After reading the package's release notice, select the desired package(s) for loading. If you do not select all packages, specify the number(s) or name(s) of the individual packages. For multiple packages, type each package's name followed by a comma (,) or space. For example,

   ```
   Package Names(s) [all] 2–4 ↵
    or
   Package Names(s) [all] fredware, fredware.man ↵
   ```

   To load all packages on the release tape, press Enter to accept the **all** default.

   At this point for tape, the tape drive will advance and rewind; for CD-ROM, the drive will access the disk The time required to load a package and the exact messages written to your screen depend on the particular package being loaded. A sample dialog for package **fredware** follows.

   ```
   Package Name(s) [all] ↵
   List file names while loading? [no] ↵
   OK to perform operation? [yes] ↵

   Positioning the tape to load: fredware .....
   Loading the package: fredware ...
   Loading the package: fredware.man  ......
   Loading the package: fredware.rn ......
   Loading the package: fredware.int ......

   The package "fredware" has been loaded.
   Package load is finished.
   The selected packages have been loaded.
   ```

Setting up packages

After loading packages, you will use **sysadm** to set up the packages. Setup involves running scripts that accompany the package. A script initializes files and moves them to the appropriate locations in the DG/UX file system. Most packages are set up in **/usr/opt/***package*, where *package* is the name of the package. Some scripts require some customizing; if so, the script will prompt you for answers. You may need to read the package's release notice for details.

The next three sections explain setup on a stand-alone computer system, an OS server, and an OS client. Continue to the appropriate section.

Stand-alone computer package setup

Use the steps in this section if your computer is neither a server or a client; that is, your system will run using the software on its own local disk(s) and no other system will use that software.

1. To set up one or more packages, follow this path through **sysadm**:

   ```
   Software -> Package -> Set up
   ```

   If there are no packages to set up, **sysadm** reports this in an error message. If there are packages to set up, **sysadm** prompts for the package name to be set up:

   ```
   Package Name(s):  [all]
   ```

2. Enter **?** for a list of packages that have been loaded but not yet set up. For example,

   ```
   Package Name(s):  [all] ? ⤶

   Select the package(s) you want to set up.  If you
   want to set up all packages, select 'all.'  If you
   select 'all,' do not select any individual package
   names.

   Choices are

   1 all
   2 fredware
   3 fredware.man
   4 fredware.rn
   5 fredware.int

   Package Name(s) [all]
   ```

3. Specify the package name(s) or press Enter for the default; for example,

   ```
   Package Name(s) [all] ⤶
   OK to perform operation? [yes] ⤶
   ```

4. If your answers are correct, confirm by pressing Enter:

   ```
   OK to perform operation? [yes] ⤶

   Setting up fredware in usr.

   Package fredware has been successfully set up in usr.
   Setting up fredware in MY_HOST root.
   Package fredware has been successfully set up in
       MY_HOST root.
   Package setup for fredware is complete.
   ```

The time required to set up the package(s) and the prompts depend on the package. Consult the package release notice for details.

OS server package setup

You can set up packages at the OS server for the server and/or for OS clients. You can also set up packages for an OS client from that client. This section assumes you will set up packages from the OS server.

1. To set up one or more packages on a server, follow this path through **sysadm**:

   ```
   Software -> Package -> Set up
   ```

   If there are no packages to set up, **sysadm** reports this in an error message. If there are packages to set up, **sysadm** prompts for the client names.

   ```
   Client(s) to be set up: [none]
   ```

   If you want to perform package setup for some or all OS clients, at the first prompt enter **?** for a list of OS clients. For example,

   ```
   Client(s) to be set up: [none] ? ↵

   Select 'all' to set up package(s) in the root file
   system of all clients attached to the selected
   release area.  Select individual clients if you want
   to set up the package only for this client.

   Select nothing or 'none' to set up the package only
   in your own root file system and not in the root
   file system of any clients.  Typically, clients set
   up their own packages.

   Choices are

   1 all
   2 none
   3 bart
   4 junior

   Client(s) to be set up: [none]
   ```

2. To perform package setup for the OS server only, take the default, **none**. To perform package setup for one or more clients, enter their names, separated by commas, or enter **all** for all clients. For example,

   ```
   Client(s) to be set up: [none] all ↵
   Package Name(s):  [all]
   ```

3. To list of packages that have been loaded but not yet set up, enter **?** For example,

```
Package Name(s):  [all] ? ↵

Select the package(s) you want to set up.  If you
want to set up all packages, select 'all.'  If you
select 'all,' do not select any individual package
names.

Choices are

1 all
2 fredware
3 fredware.man
4 fredware.rn
5 fredware.int

Package Name(s):  [all]
```

4. Specify the package name(s) or press Enter for the default; for example,

```
Package Name(s) [all] ↵
OK to perform operation? [yes] ↵
```

5. If your answers are correct, confirm by pressing Enter:

```
OK to perform operation? [yes] ↵

Setting up fredware in usr.

Package fredware has been successfully set up in usr.
Setting up fredware in MY_HOST root.
Package fredware has been successfully set up in
    MY_HOST root.
Package setup for fredware is complete.
```

The time required to set up the package(s) and the prompts depend on the package. Consult the package release notice for details.

OS client package setup

You can set up an OS client package at either the OS server or at each OS client that is fully operational (the OS client has booted its kernel). This section assumes you will perform setup at the OS client after its kernel is booted. To perform setup for OS clients at the server, see the previous section.

1. To set up one or more packages on an OS client, follow this path through **sysadm**:

   ```
   Software -> Package -> Set up
   ```

   If there are no packages to set up, **sysadm** reports this in an error message. If there are packages to set up, **sysadm** prompts for the package names:

   ```
   Package Name(s):  [all] ? ↵
   ```

2. Enter **?** for a list of packages that have been loaded but not yet set up. For example,

   ```
   Package Name(s):  [all] ? ↵

   Select the package(s) you want to set up.  If you
   want to set up all packages, select 'all.'  If you
   select 'all,' do not select any individual package
   names.

   Choices are

   1 all
   2 fredware
   3 fredware.man
   4 fredware.rn
   5 fredware.int

   Package Name(s):  [all]
   ```

3. Specify the package name(s) or press Enter for the default; for example,

   ```
   Package Name(s) [all] ↵
   OK to perform operation? [yes] ↵
   ```

4. If your answers are correct, confirm by pressing Enter:

   ```
   OK to perform operation? [yes] ↵

   Setting up fredware in usr.

   Package fredware has been successfully set up in usr.
   Setting up fredware in MY_HOST root.
   Package fredware has been successfully set up in
       MY_HOST root.
   Package setup for fredware is complete.
   ```

   The time required to set up the package(s) and the prompts depend on the package. Consult the package release notice for details.

Where to go next

Adding and setting up packages may require that you rebuild and reboot the DG/UX kernel. Check the package release notice for any requirements to add or change a tunable variable in the system file. For information on building and booting a kernel, see Chapter 10.

End of Chapter

7 Adding OS clients and X terminal clients

This chapter explains how to make client computers operational. An OS server supplies a bootable operating system and file system space over a local area network (LAN) to an OS client. If you do not have this OS configuration, skip this chapter.

Also, this chapter explains procedures for adding an X terminal to your configuration. An X terminal provides X Window system graphics support. Lacking its own disk and operating system, the X terminal depends on an OS server for its secondary bootstrap.

IMPORTANT: This chapter assumes that the clients will run DG/UX 5.4 Release 3.00 from the server's primary release area; it also assumes that the server is running a DG/UX 5.4 Release 3.00 kernel. If you want to connect an OS client to a secondary release area, you must create a secondary release area and execute the other steps covered in Chapter 8.

Major sections proceed as follows.

- Types of OS client
- Virtual disk requirements for OS clients
- Types of OS releases
- Adding a diskless OS client
- Adding an OS client that has local **root** and **swap** virtual disks
- Adding an OS client that has a local **swap** virtual disk
- Adding an X terminal client
- Where to go next

Types of OS client

The OS server supplies a bootable operating system and file system space over a local area network (LAN) to an OS client. The OS client comprises the **root** and **usr** file systems and **swap** area which are typically exported from the OS server as the **/srv/release/PRIMARY/root**, **/usr**, and **/srv/swap** file systems. In addition, **/srv/dump** is needed for OS clients. The **srv_root**, **usr**, **srv_swap**, and **srv_dump** virtual disks are needed for these file systems and special areas. Normally, an OS client with no attached disk drive receives its entire operating system and file system space from an OS server. However, if an OS client has an attached disk drive, you may choose to put part of the operating system on the local disk drive to maximize system resources and to improve system performance. The three required virtual disks (**root**, **usr**, and **swap**) can be divided between two disk drives, one attached to the OS server and the other to an OS client. The three supported types of OS clients are

Diskless The OS client has no disk drive and receives its entire operating system and file space from the OS server over a LAN.

Local root and swap An OS client with an attached disk drive provides its own local **root** and **swap** virtual disk resources, while receiving its **usr** virtual disk resources from the OS server via a LAN.

Local swap An OS client with an attached disk drive provides only its own local **swap** virtual disk resources, while receiving its **root** and **usr** virtual disk resources from the OS server via an LAN.

Virtual disk requirements for OS clients

Before you add OS clients, at the OS server you must create the appropriate virtual disks to accommodate them. If you haven't created virtual disks, refer to Chapter 2 and do so before continuing with these instructions. The required virtual disks are as follows. Chapter 2 explains how to calculate their sizes.

- **srv**
- **srv_root**
- **srv_swap**
- **srv_dump**

Secondary OS releases

In addition to providing DG/UX 5.4 Release 3.00 as the primary release to its OS clients, an OS server can provide other (secondary) releases of DG/UX or a foreign operating system. The DG/UX file system supports operating system releases in different release directories. DG/UX 5.4 Release 3.00 is in **/srv/release/PRIMARY**; secondary releases are in **/srv/release/***release-name*, where *release-name* is the name of the secondary release, such as **dgux_542**. For a picture of the primary and secondary file structure, see Chapter 2, Figure 2–3.

When you have created the secondary release area (procedures are in Chapter 8), adding OS clients to the primary and secondary release areas is identical. You can attach an OS client to only one release at a time.

The examples in this chapter emphasize the addition of OS clients to the primary release because they are the most common. If you are adding clients to a secondary release, you can substitute the current *release-name* for each instance of PRIMARY in the pathnames given in this chapter.

Now, continue to the section that describes the kind of client you want to add, as follows.

- Adding a diskless OS client
- Adding an OS client with local **root** and **swap** virtual disks
- Adding an OS client with a local **swap** virtual disk
- Adding an X terminal client

Adding a diskless OS client

Perform these procedures at the OS server:

- Complete the planning worksheets.
- Add an OS client to the network databases.
- Build a first-time custom kernel for the client.
- Add an OS client to the operating system release.
- Boot the OS client kernel.
- Set up packages.

These procedures are discussed below.

Network planning for OS clients

Two OS Client Network Planning Worksheets follow this section to help you prepare for installation. They list network parameters for which you need to supply values. Completing the worksheets before you begin the installation procedures for OS clients will speed up the installation process considerably.

If you have experience installing the DG/UX system or other operating systems, you may prefer to go directly to the OS Client Network Planning Worksheets. The following list defines the network parameters you will need to complete the planning worksheets.

Hostname

A unique name you assign to your computer system functioning as an OS client that can contain up to 31 alphabetic and numeric characters. However, you are advised to keep the names short. Host names that relate to the use or location of the system are particularly helpful in networked environments where hosts may share file systems. Examples of hostnames are **fred**, **jamaica**, and **writer_doc**. Do not use the capitalized names **MY_HOST** or **PRIMARY**; these names are reserved by the system.

Internet address

The network administrator provides the Internet address of the host being set up. An example of an Internet address is **128.223.2.1**. For details, see *Managing TCP/IP on the DG/UX™ System*.

Ethernet address

Host address that is unique to the particular hardware. The factory sets this address. It consists of six 2-digit (hexadecimal) fields separated by colons in the form *nn:nn:nn:nn:nn:nn*.

NIS domain name

A named set of NIS maps located in **/etc/yp/***domain-name*. Computer systems having this directory as their default NIS domain share the data found in its maps.

Do you subnet?

Subnetting allows you to view multiple physical networks as a single logical network. You must answer "yes" or "no" to the query about subnetting.

Network mask

A hexadecimal bit pattern that specifies the number of bits used to identify the network part of an Internet address. **0xff000000** and **0xffff0000** are examples of network masks. You need to specify a network mask only when your network is subnetted.

Planning worksheets

Figure 7–1 provides a worksheet in which you record the Internet address and Ethernet address for each OS client on your system; Figure 7–2 is another worksheet for entering the appropriate ONC and TCP/IP parameter values required for each OS client's package setup.

OS Client Hostname	Internet Address	Ethernet Address
Example: junior	*123.227.3.14* *(use periods)*	*08:00:1B:03:45:11* *(use colons)*

Figure 7–1 OS Client Network Planning Worksheet (Part 1)

ONC and TCP/IP Parameters	Example Values	Actual Values
NIS domain name	*my_domain*	
Do you subnet?	*yes*	
Network mask	*0xffffff00*	

Figure 7–2 OS Client Network Planning Worksheet (Part 2)

Adding OS clients to the OS server's network databases

To add an OS client to a number of network databases, you perform the following tasks at the OS server:

- Add the OS client to the OS server's **hosts** database.
- Add the OS client to the OS server's **ethers** database.
- Add users on the OS client to the OS server' **host.equiv** database.

IMPORTANT: If you installed the ONC/NFS package on the OS server and you declared the OS server an NIS server (discussed in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*), then you must set up OS clients in the NIS **hosts** and **ethers** databases instead of local **hosts** and **ethers** databases. However, you must delay the addition of OS clients to a release until the additions of OS clients to the NIS **hosts** and **ethers** databases have time to propagate through the network. Therefore, you should add OS clients to the NIS **hosts** and **ethers** databases well before (for example, the day before) adding OS clients to the operating system release, which is covered in "Adding an OS client to a release," later in this chapter.

Adding an OS client to the OS server's hosts database (/etc/hosts)

The **hosts** database contains a list of Internet address and OS client hostname pairs to enable network communications between the OS server and its OS clients. To add an entry to the **hosts** database, perform the procedures in this section at the OS server.

1. Follow this path through **sysadm** and respond to the prompts:

   ```
   Networking -> TCP/IP -> Databases -> Hosts -> Add
   ```

   The **sysadm** prompt depends on whether your host is an OS server that has also been declared as the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.* If your host is the NIS master, **sysadm** displays

   ```
   Hosts database to use: [NIS (YP)]
   ```

   If your server is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Hosts database will be used.` **Sysadm** will let you insert a new entry in the *local* **hosts** database for each OS client. Skip to step 3.

2. On the NIS master, press Enter to accept the default value **NIS**:

   ```
   Hosts database to use: [NIS (YP)] ↲
   ```

3. **Sysadm** displays

   ```
   Host Name:
   ```

4. Consult the OS Client Network Planning Worksheet that you completed and enter the hostname; for example,

   ```
   Host Name: junior ↵
   Internet Address:
   ```

5. From your OS Client Network Planning Worksheet, enter the four-field (four octet) Internet address. For example,

   ```
   Internet Address: 123.227.2.14 ↵
   Alias List:
   ```

6. You may specify an alternate name (alias) for your host. For example, if your department uses long hostnames, you may choose a shorter alias for convenience. For example,

   ```
   Alias List: jr ↵
   OK to perform operation? [yes]
   ```

7. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example,

   ```
   OK to perform operation?  [yes] ↵
   junior has been added.
   ```

   Sysadm has added the OS client to the appropriate hosts file. Repeat this procedure to add as many OS clients as desired to the **hosts** database.

Adding an OS client to the OS server's Ethernet database (/etc/ethers)

The **ethers** database contains a list of Ethernet address and OS client hostname pairs to enable network communications between the OS server and its OS clients. To add an entry to the **ethers** database, perform these steps at the OS server.

1. Follow this path through **sysadm**:

   ```
   Networking -> TCP/IP -> Databases -> Ethers -> Add
   ```

   The **sysadm** prompt depends on whether your host is an OS server that has also been declared as the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. If your host is the NIS master, **sysadm** displays

   ```
   Ethers database to use: [NIS (YP)]
   ```

   If your host is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Ethers database will be used.` **Sysadm** will let you insert a new entry in the *local* **ethers** database for each OS client. Skip to step 3.

2. On the NIS master, press Enter to accept the default value **NIS**:

```
Hosts database to use: [NIS (YP)] ↵
```

3. **Sysadm** prompts

```
Host Name:
```

4. Consult the OS Client Network Planning Worksheet that you completed and enter the hostname; for example,

```
Host Name: junior ↵
Internet Address:
```

5. From your OS Client Network Planning Worksheet, enter the Internet address, form *hh:hh:hh:hh:hh* where *h* is a hexadecimal number. For example,

```
Internet Address: 08:00:1B:03:32:43 ↵
OK to perform operation?  [yes]
```

6. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example,

```
OK to perform operation?  [yes] ↵
junior has been added.
```

Sysadm has added the OS client to the appropriate **ethers** file. To add another OS client to the **ethers** database, repeat these steps.

Adding an OS client user to the OS server's trusted hosts database (/etc/host.equiv)

Allowing a user to access the server using a remote login may be helpful for such tasks as checking the print queue for a printer on the OS server. To add a user login name to the trusted **hosts** database, perform the procedures in this section at the OS server.

1. Follow this path through **sysadm**:

```
Networking -> TCP/IP -> Databases -> Trusted Hosts -> Add
```

Sysadm prompts

```
Host Name:
```

2. Enter the hostname to which the user will make remote requests and remotely log in. For example,

```
Host Name: junior ↵
User Name: [all]
```

3. Specify the login name of the user who will have remote request and login privileges. Or you can specify all (local) users. For example,

```
User Name: [all] johnson ↵
OK to perform operation?  [yes]
```

4. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example,

   ```
   OK to perform operation?  [yes] ↵
   johnson on junior has been added.
   ```

   The named user on the named remote system (the OS client) in the **/etc/host.equiv** file can access the OS server system using the remote shell (**rsh** or **remsh**) or the remote login (**rlogin**). Refer to the **rsh**(1), **remsh**(1), and **rlogin**(1) manual pages for more information.

Building a first-time custom kernel for an OS client

OS clients as well as the OS server must have kernels to provide operating system services. (Refer to Chapter 10 for complete information on kernels.) Perform the procedures in this section at the OS server.

The OS server must build each OS client's first kernel to establish an OS client's basic operation. You may build one kernel for all OS clients, or you may build individual kernels for each OS client. After an OS client is operational, it can build its own kernel if necessary.

Deciding whether to build a common or unique kernel

If you are adding multiple OS clients, decide whether to build a common kernel for all OS clients or individual kernels for each OS client. If all OS clients have identical hardware and software configurations (for example, none has an attached I/O device), you can build a common kernel for all OS clients to share.

A primary advantage of a common kernel is to save mass-storage disk space. Conversely, a disadvantage is reduced security. Because all OS clients have root access to the kernel, any user with superuser privileges on an OS client computer can alter the kernel and change the common kernel.

The standard location for all kernels of the primary release of the DG/UX system is **/srv/release/PRIMARY/root/_Kernels**. Each OS client has a link named **dgux** in its primary release directory (**/usr/release/PRIMARY/root/***client-hostname*) to its kernel in the _Kernels directory. Multiple OS clients sharing the same kernel are linked to a common kernel from their client root directories. By convention, the common kernel is named **dgux.diskless**.

We recommend a unique kernel if an OS client's hardware configuration differs from the common OS clients' (for example, if the OS client has an attached parallel line printer). If the kernel includes an I/O device that is not physically attached to a given OS client computer, you can still boot that kernel, but an error message will appear indicating that the device could not be configured. A disadvantage of each unique kernel is that it requires more megabytes of disk space on the server.

Generally, you should build the most common kernel first, and then build each individual kernel (as needed). To create an individual kernel for an OS client, be sure to supply a unique system configuration filename.

1. To build a first-time kernel for an OS client, follow this path through **sysadm**:

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

   Sysadm prompts

   ```
   System configuration file name: [xxx]
   ```

2. Enter the system configuration filename extension. A conventional name for a system file for a shared kernel is **system.diskless**. Entering **diskless** here yields the system filename **system.diskless**. For example,

   ```
   System configuration file name: [xxx] diskless ↵
   [system.diskless] Correct? [yes]
   ```

3. If you are happy with the name, confirm with Enter. Otherwise, enter **n** and enter a different name. For example,

   ```
   [system.diskless] Correct? [yes] ↵
   Build for this host or for OS client(s) of
        this host: [this host]
   ```

4. You want to build this for an OS client, so enter **OS Client** in either upper- or lower-case.

   ```
   Build for this host or for OS client(s) of
        this host: [this host] OS client ↵
   Editor: [/usr/bin/vi]
   ```

5. We suggest you use the **vi** editor (the default) to edit the system file. Appendix A has a **vi** command summary. Or you can specify another editor by entering its complete pathname at the prompt. For example,

6. `Editor: [/usr/bin/vi]` ↵

   Sysadm now runs the editor you chose on the system file. The system file is long; it spans several screens. You should comment out configuration variables that do not apply to your system. Place a # in the first column of a line you want commented out Read the contents of the system file before you edit it.

The three sections that you are most concerned about are

- OS client configuration variables
- General configuration variables
- OS client hardware devices

OS client configuration variables

The system file section on OS client configuration variables follows.

```
# OS Client Configuration Variables:
#
# These configuration variables specify that the root and swap
# file systems will be mounted over NFS.
#

      NETBOOTDEV          "inen()"
      ROOTFSTYPE          NETWORK_ROOT
      SWAPDEVTYPE         NETWORK_SWAP
```

NETBOOTDEV, ROOTFSTYPE, and SWAPDEVTYPE provide resources to the OS client over the network. Make sure no comment symbols (#) appear in the first column of the lines containing these variables.

General configuration variables

The system file section on general configuration variables follows.

```
# General Configuration Variables:
#
# The NODE variable controls your nodename for uname(1) and
# uucp(1).
#
      NODE                "diskless"
```

NODE refers to the hostname of your computer. After you supply the name of the system file, **diskless**, it becomes the NODE name. However, each time the OS client is rebooted, this value is overridden by the hostname provided during TCP/IP package setup (covered in Chapter 6). The node name is used by the **uname** command to report the name and other attributes of the current system. Also, the UUCP facility uses the node name for performing file transfers on UNIX systems. It is also presented as part of the log-in banner message when you log in to your system. The node name is restricted to 31 characters.

The master files located in the **/usr/etc/master.d** directory contain complete lists of general configuration variables and default values. You accept the master file defaults by not resetting any variables in your system file. If you wish to override a master file default, reset it in your system file. *Managing the DG/UX™ System* explains the entire list of variables and legal settings.

OS client hardware devices

Two sections of the system file list I/O devices that can be attached to an OS client. The first section relates to common hardware devices attached to an OS client. In the sample that follows, the I/O device names are not commented out.

```
# OS Client Hardware Devices:
#
# These hardware devices are found on most operating system
# clients.  You must add any other hardware devices found on
# the client (using the examples below as a guide), and delete
# any devices which are not found on the client.
#
      kbd()     # -- keyboard
      grfx()    # -- graphics display
      inen()    # -- integrated Ethernet controller
      duart()   # -- integrated Duart terminal line controller
```

A second section offers more I/O devices that can be attached to the OS client. These I/O device names are commented out. Remove the pound sign (#) from the first line of those devices that reflect the OS client's configuration. As a convenience, the SCSI disk and tape device names contain an asterisk in the SCSI ID field, which is a metacharacter that matches any SCSI ID value on any attached device. The section on typical OS client devices follows.

```
# Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both.  Determine which situation applies.

#     kbd()        # -- keyboard
#     grfx()       # -- graphics display
#     sd(insc(),*) # -- all SCSI disks on integrated SCSI adapter
#     st(insc(),*) # -- all SCSI tapes on integrated SCSI adapter
#     inen()       # -- integrated Ethernet controller
#     duart()      # -- integrated Duart terminal line controller
#     duart(1)     # -- second Duart (if present on system)
#     lp())        # -- integrated printer controller (if present)
```

If you have just loaded a new package on your system, check its release notice for information on possible variable tuning. If you need to tune a variable, either enter or modify it in the appropriate location in the system file.

A common source of kernel build failures is the absence of a comment symbol (#) used to flag notes to be ignored. Be sure you comment out all text you want ignored. Check your spelling and verify the DG/UX device names.

7. After you finish editing the system file, write the file and quit the editor by typing the **ZZ** command.

   The prompts that appear during a kernel build depend on whether your system has existing OS clients. The following prompt appears if your system has no existing OS clients.

   ```
   There are no operating system clients of this machine
   for which to link the new kernel.  The new kernel may be
   used by any clients added later. The kernel pathname will
   be /srv/release/PRIMARY/root/_Kernels/dgux.diskless.
   Continue with the build?  [yes]
   ```

   The following prompt appears if your system *has* existing OS clients.

   ```
   Link the new kernel to which OS clients: [all] ↵
   ```

8. If you see the “`Continue`” prompt, confirm by pressing Enter; for example,

   ```
   Continue with the build?  [yes] ↵
   Configuring system...
   Building kernel...
   Successfully built dgux.diskless.
   ```

   Skip to step 12.

9. If you see the “`Link...to clients`” prompt, this means your system has OS clients. Continue with this step. **Sysadm** is asking

   ```
   Link the new kernel to which OS clients: [all]
   ```

   To boot a kernel, a client must be able to resolve the filename **dgux** in that client’s release root directory (**/usr/release/PRIMARY/root/***client-hostname*) to a kernel. **Sysadm** handles this by creating a link from filename **dgux** in each client’s release root directory to the kernel name (in **/usr/release/PRIMARY/_Kernels**).

   If all (or most) of your clients can use the same kernel, you can save a lot of disk space by having links created in each client’s release root directory to a single kernel in the **_Kernels** directory. This lets you use one kernel for multiple clients. To have **sysadm** create these links, select **all**, the default.

   If you’re creating an individual kernel for a client that has special needs, you don’t want to link for all clients to that kernel. To link a specific OS client to this kernel, enter the hostname of the OS client instead. If **sysadm** is not displaying a list of previously added OS clients (as with ASCII **sysadm**), enter **?** for a list.

   Whatever selection you make, the link replaces any existing links. If you select **all,** the program links all OS clients to the kernel. If you enter a client hostname, the program replaces any existing link only for that client; other existing client links are retained. This means, as mentioned earlier, that you should build the most common kernel first, have it linked for all clients, and then if needed create any individual kernels and have those linked for individual clients.

Decide on your response and enter it. For example,

```
Link the new kernel to which OS clients: [all] ↵
```

After you answer this prompt, **sysadm** asks

```
Save the old kernel? [yes]
```

10. If you can afford the disk space, we recommend you accept the **yes** default response.

```
Save the old kernel? [yes] ↵
Continue with the build? [yes]
```

11. To continue the build process, press Enter:

```
Continue with the build? [yes] ↵
Configuring system...
Building kernel...
Successfully built dgux.temp
```

12. After the new kernel is built, the bootable kernel file is in the **/srv/release/PRIMARY/root/_Kernels** directory as **dgux.***name* and, if you so specified, it is also linked to **dgux** for the specified OS client(s) in each of their client root directories.

    The new kernel does not become effective until someone boots it over the network from an OS client. Do not boot the kernel yet. At the OS server, you must first add each OS client to the DG/UX system release as explained next.

Adding an OS client to a release

Perform these procedures at the OS server.

After the OS server is operational — that is, a custom DG/UX system is running in the server's root, and a DG/UX release has been installed and a kernel built in the primary release area (and perhaps secondary release area) — then you can attach each OS client to the release it wants. You identify a release to an OS client by the name of its release area, the name of the OS server; the pathname of the OS client's home directory, the OS client's swap size, and the pathname of the OS client's bootable kernel and bootstrap. If you have several clients to add, you can create a default client set with the above values you want and use them as a base for each client you add.

It may seem that a server running DG/UX 5.4 Release 3.00 cannot support a client with an attached disk with that uses a release before 5.4 Release 3.00; however, such a server *can* support such a client, and vice-versa. This is true because the systems are communicating over the network, not using the same physical disk, and disk format restrictions between the releases do not apply.

To add several OS clients to a release, you will perform the following tasks:

- Create a default OS client set.
- List the contents of a default OS client set.
- Modify a default OS client set.
- Add each OS client (based on the values in the default OS client set).
- List the added OS clients.

To add a single OS client, just add the OS client; you can skip the client set tasks.

IMPORTANT: If you installed the ONC/NFS package on your system and you declared the OS server an NIS master (declaring an OS server as the NIS master is discussed in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System),*you set up OS clients in NIS hosts and **ethers** databases instead of local hosts and **ethers** databases. You must delay the addition of OS clients to an operating system release so that the additions of OS clients to the NIS hosts and **ethers** databases will have time to propagate through the network. Therefore, you should add OS clients to the NIS hosts and **ethers** databases at least several hours (perhaps a day) before adding OS clients to the operating system release; the section “Adding an OS client to a release” explains this procedure. If you do not wait long enough, addition of OS clients to the operating system release will fail. Refer to “Adding OS clients to the OS server’s network databases” for more information.

Creating a default OS client set

A client set specifies values to be shared by OS clients linked to a common kernel. The default client set provides default values for each client you add. Creating your own default set and making it the default can save you a lot of time as you add OS clients.

A default client set named **unnamed** is supplied with DG/UX, but you may want to create a custom set with a different name. To create a default client set, perform these steps at the OS server.

1. Follow this path through **sysadm** :

```
Client -> OS Client -> Defaults -> Create
```

Sysadm prompts

```
Set Name: [unnamed]
```

2. You can either accept the default client set name, **unnamed**, or you can create a client set to be customized. Each client set name must be unique, so if the set **unnamed** has already been created, you will need to use another name. To accept the default client set, **unnamed**, and its default attributes press Enter; or to create a custom client set, enter its name. For example, to create a custom client named **dgset**.

```
Set Name: [unnamed] dgset ↵
OK to perform operation?  [yes]
```

3. To create the new default client set, press Enter:

```
OK to perform operation?  [yes] ↵
```

You have created a default client set. You can want to list or modify its contents as explained in following sections.

Listing the contents of a default client set

A client set specifies general attributes of the operating system environment shared by all OS clients linked to a common kernel. To list the attributes of a client set, perform the procedures in this section at the OS server.

1. Follow this path through **sysadm**:

```
Client -> OS Client -> Defaults -> List
```

Sysadm prompts

```
Set Name: [default]
```

2. Enter a name or ? for a list of names. For example,You then confirm your request:

```
Set Name: [default] dgset ↵
OK to perform operation? [yes]
```

3. Confirm with Enter and **sysadm** will list parameters for the set:

```
OK to perform operation? [yes] ↵

Parameter         Value
---------         -----
client_release    PRIMARY
client_server
client_homedir    /home
client_swapsize   24
client_kernel   /srv/release/PRIMARY/root/_Kernels/dgux.diskless
client_bootstrap /usr/stand/boot.aviion
```

The values displayed for your client set may differ.

Modifying a default OS client set

After creating a custom default client set, you must modify it. If you accept the default client set named **unnamed**, you do not need to modify it.

1. To modify the default client set, follow this path through **sysadm**.

   ```
   Client -> OS Client -> Defaults -> Modify
   ```

   Sysadm prompts

   ```
   Set Name: [default]
   ```

2. Enter the name of the default set you created with the Create function. For example, **dgset**.

   ```
   Set Name: [default] dgset ↲
   Release Area: [PRIMARY]
   ```

3. The default release name is PRIMARY. If you want to specify a different release area, that release area must already have been created (Chapter 8, section “Creating a secondary release area”), and enter its full pathname. For the primary area, press Enter:

   ```
   Release Area: [PRIMARY] ↲
   Server Host Name: [xxx]
   ```

4. The default server hostname, *xxx*, is your current hostname. Generally, since you are running **sysadm** on the server, this default is correct. If so, take the default. For example,

   ```
   Server Host Name: [boss] ↲
   Home Directory: [/home]
   ```

5. This sets the name of the file system on the server that will contain the home directories for the users on the OS client system. (Someone on the client will need to add these users via `User -> Login account`.) **Sysadm** will create a mount point for the file system relative to the OS client; then it will add the directory name to the OS client’s file system table **/etc/fstab** so it will be mounted automatically at client startup. The file system you specify must already be mounted on the OS server and be exportable; that is, someone on the server must have added the directory as a file system and made it exportable. The file system directory should be the same as the parent directory you specify when adding a user on the OS client.

   If you do not specify a directory, **sysadm** does not add a home directory entry to the OS client’s **/etc/fstab** file. To not specify a directory in the ASCII **sysadm**, enter **none**; in the X window version, erase the default name.

   For example, to select the default file system, /home:

   ```
   Home Directory: [/home] ↲
   Swap Size (in megabytes): (4-128) [24]
   ```

6. For the swap size, accept the default or enter the size of **/srv/swap** you recorded in your planning worksheet for virtual disks.

```
Swap Size (in megabytes): (4-128) [24] ↵
Kernel Pathname:
  [/srv/release/PRIMARY/root/_Kernels/dgux.diskless]
```

7. This specifies the pathname of the kernel that all clients will use. (**Sysadm** will create a link in each client's release directory to this pathname.) You created this kernel in a previous section. If you used the default kernel name, diskless, then you can take the default. If you specified a different kernel name, specify the full pathname (**/srv/release/PRIMARY/root/_Kernels/dgux.***xxx*), where *xxx* is the name you gave). If you created a special kernel for any client that has unique needs, you can modify this value later for that client using the **sysadm** Client menu. For example, for the default,

```
Kernel Pathname:
  [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵
Bootstrap File: [/usr/stand/boot.aviion]
```

8. The bootstrap file contains the bootstrap needed by each OS client to boot a kernel. Accept the default bootstrap pathname:

```
Bootstrap File: [/usr/stand/boot.aviion] ↵
OK to perform operation? [yes]
```

9. Press Enter to accept the values you have entered.

```
OK to perform operation? [yes] ↵
Defaults for set dgset have been modified.
```

You have finished the modify/add operation. Next, if you have not already done so, you should make your custom set the default.

Selecting the default client set

Selecting a default client set makes it the system default. You must select the default client set before using it.

1. To select a default client set, follow this path through **sysadm**:

```
Client -> OS Client -> Defaults -> Select
```

Sysadm prompts

```
Set Name: [system]
```

2. Specify the name you want as the system default. If the default set names are not on display, enter a question mark (?) for a list. For example

```
Set Name: [system] ? ↵
```

```
...Choices are

     1  system
     2  dgset
     3  unnamed
...
Set Name: [system]
```

3. Specify the name or menu number of the set you want to be the default set. For example,

```
Set Name: [system] dgset ↵
OK to perform operation? [yes]
```

4. Confirm your choice:

```
OK to perform operation? [yes] ↵
Default set dgset has been selected.
```

Adding an OS client using the default client parameters set

After adding entries for an OS client in the **hosts** and **ethers** databases, and creating and selecting a default set, you can add the OS client.

1. Following this path through **sysadm**:

   ```
   Client -> OS Client -> Add
   ```

   Sysadm prompts

   ```
   Client Host Name:
   ```

2. Enter the hostname for the OS client to add using the default set. For example,

   ```
   Client Host Name: junior ↵
   Release Area: [PRIMARY]
   ```

3. This Release Area prompt and all the remaining prompts are identical to those for modifying a default client set. Return to the section “Modifying a default OS client set,” step 3.

   For every client you want to add, repeat steps 1 through 3 in this section.

Listing OS clients

To verify the successful addition of OS client(s), follow this path through **sysadm** and respond to the prompts:

```
Client -> OS Client -> List
```

Specify a client hostname or **all**. If you specify all clients, a list of name and release areas appears. For example,

```
Client Host Name: all ↵
OK to perform operation?  [yes] ↵

Client Name              Release Area
-----------              ------------
matilda                  PRIMARY
junior                   PRIMARY
client_dgux542           dgux542
```

Booting an OS client kernel

A kernel must be running on an OS client before anyone can set up packages and log in that client. Perform these procedures on the OS client.

1. If the client you want to boot has a DG/UX system running, you must first shut down the system to reboot. If it does not have DG/UX running, skip to step 2.

   Use these commands to shut down the client system.

   ```
   # cd / ↵
   # shutdown -g0 -y ↵
   # halt ↵
   SCM>
   ```

   Your screen should display the SCM> prompt.

2. Boot the OS client's kernel over the network to a run level of i, using the command form **b** *network-controller* **–i**, where the *network-controller* specifies the server's network controller connected to this client, controller type **dgen** or **inen**. For example,

   ```
   SCM> b dgen(0) –i ↵
   ```

   Your screen displays a series of booting messages during which you must confirm the current date and time. You must also specify the boot device. The first set of booting messages looks like the following.

   ```
   Booting inen(0) -i
   DG/UX System 5.4R3.00 Bootstrap
   Loading image ........
   Network boot device [?]
   ```

3. Enter the same network controller name you specified in step above; for example,

   ```
   Network boot device [?] dgen(0) ↵
   ```

   DG/UX displays more boot messages as follows.

   ```
   Linking short names for /dev device nodes .....
   Starting disk daemons ....
   Mounting local file systems .......
   Starting installation steps ....

   Set up package(s)? [yes]
   ```

You will now set up packages as explained next.

Setting up packages

Before the OS client can be fully operational, someone must set up DG/UX packages.

Perform these procedures at the OS client. Use the OS Client Package Setup Worksheet you completed earlier when setting up packages.

When prompted to set up packages, request help by entering a question mark (?). Figure 7–3 shows a typical dialog for starting the package setup process; the dialog selects NFS, ONC, TCP/IP, and X11.

```
Package Name(s):   [all] ? <Enter>

Select the package(s) you want to set up.  If you want to set up all
packages, select 'all.'  If you select 'all,' do not select any
individual package names.

Choices are

1 all
2 dgux
3 gcc
4 networker
5 nfs
6 onc
7 tcpip
8 X11
9 X11.sde
10 xdt
11 dgux.man
...
Enter a number, a name, ...q to quit: 5, 6, 7,8, 9 <Enter>
OK to perform operation? [yes] <Enter>
```

Figure 7–3 Typical Package Setup Dialog

The choices listed depend on the packages that were loaded and set up on the OS server.

Setting up NFS

The following messages appear for **NFS**.

```
Setting up nfs in MY_HOST root.

    Setting up NFS in MY_HOST root...........

    Creating NFS run level links.........

    Initializing NFS prototype files........

    NOTE:  See /srv/release/PRIMARY/root/MY_HOST/var/setup.d
           /log/nfs.root for a detailed account of the root
           setup of NFS.

Package nfs has been successfully set up in MY_HOST root.
Package setup for nfs is complete.
```

Setting up ONC

In setting up **ONC**, you need to supply one value: the name of the NIS domain. Use the OS Client Package Setup Worksheet you completed earlier when setting up packages.

```
Setting up onc in MY_HOST root.

       Setting up ONC in MY_HOST root..........

       Creating ONC run level links...........

       Initializing ONC prototype files.......

       Enter the NIS Domain name []: work-net ↵
       [work-net] Correct? [yes]: ↵

       NOTE:  This host will first run as an NIS client.

       NOTE: See /srv/release/PRIMARY/root/MY_HOST/var/setup.d
             /log/onc.root for a detailed account of the root
             setup of ONC.

Package onc has been successfully set up in MY_HOST root.
Package setup for onc is complete.
```

IMPORTANT: If you are upgrading, the NIS domain name is supplied by default.

Setting up TCP/IP

Setup of TCP/IP requires more planning than other packages. You must supply configuration information to the **sysadm** prompts. Use the OS Client Package Setup Worksheet you completed earlier when setting up packages. Also, refer to *Managing TCP/IP on the DG/UX™ System* for information on setting up the TCP/IP package.

```
Setup package tcpip in MY_HOST root

   Setting up tcpip...

   The following queries refer to the primary
         network interface.

   Enter host name: junior ↵
   [junior] Correct? [yes] ↵
   Enter host Internet address: 128.222.8.60 ↵
   [128.222.8.60] Correct? [yes] ↵
   Is your local network subnetted? [no] yes ↵
   Enter the network mask: 0xffffff00 ↵
   [0xffffff00] Correct? [yes] ↵

   NOTE: Using inen0 as the primary network interface
             controller.

   Package tcpip has been ... set up in MY_HOST root.

   NOTE:  See /var/setup.d/log/tcpip.root file for a
            verbose description of the package setup for root.

   Package setup for tcpip is complete.
```

Setup for TCP/IP is complete.

Setting up X11

The following messages appear for **X11**:

```
Setting up X11 in MY_HOST root.
Package X11 has been ... set up in MY_HOST root.
Package setup for X11 is complete.
```

Setup for X11 is complete.

Moving to run level 3

Perform these procedures at the OS client.

After you set up packages, **sysadm** asks if you want to build and reboot another kernel; answer **no** as follows.

```
Build kernel? [yes] no ↵
Reboot now? [yes] no ↵
DG/UX Operating System      Release n

Console Login:
```

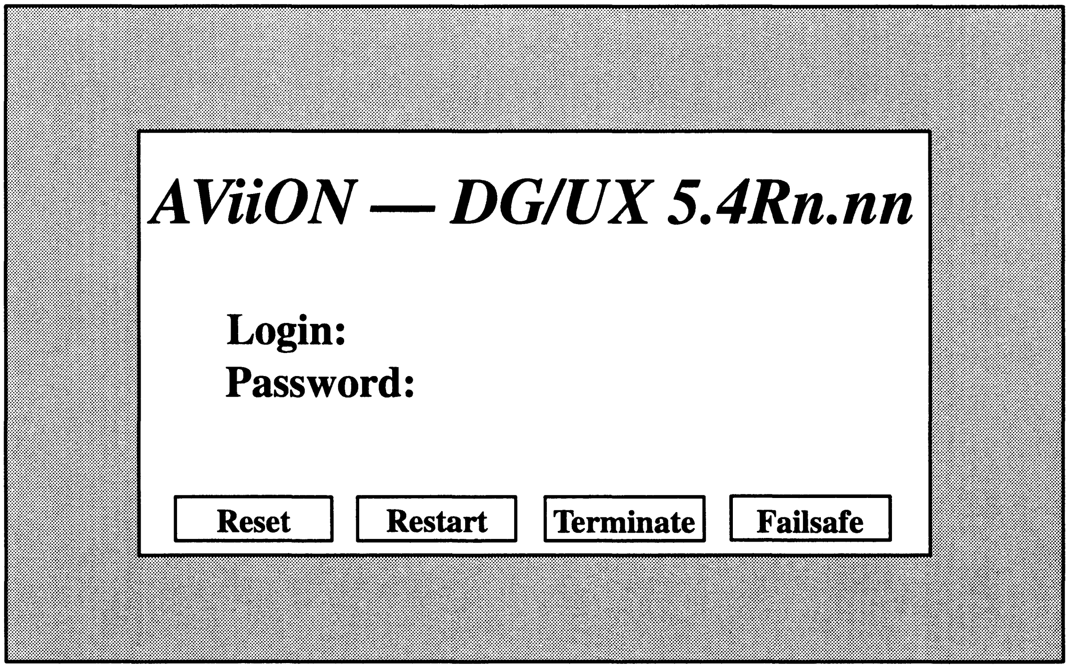
At the prompt, type the **sysadm** login name. The screen displays a shell prompt (#) to indicate a successful login. At the prompt, enter **init 3**, to move to run level 3, as follows.

```
# init 3 ↵
```

The screen displays a series of messages indicating a transition from run level **i** to **3**. A login prompt appears at the conclusion of the process. The type of login prompt depends on whether or not your computer system is equipped with the DG/UX X Window System package. Choose the appropriate section following.

Logging in the X Window system

The screen displays a login box as follows in Figure 7–4.



AViiON — DG/UX 5.4Rn.nn

Login:
Password:

Figure 7–4 Login Screen for the X Window System

The four buttons at the bottom of the screen control the X server, the program that governs the display of information to the screen.

Reset Resets but does not terminate the X server.

Restart Terminates the X server, then restarts it.

Terminate Terminates the X server, giving control to a single VT100 terminal emulation screen.

Failsafe Is a toggle button that limits startup to a single **xterm** window.

IMPORTANT: If you choose to exit from the X Window environment and perform the procedures in a single VT100 terminal screen environment, move the cursor to the "Terminate" button and click. To return to the X Window environment, enter **xdm** at the login prompt. The X Window environment is restored.

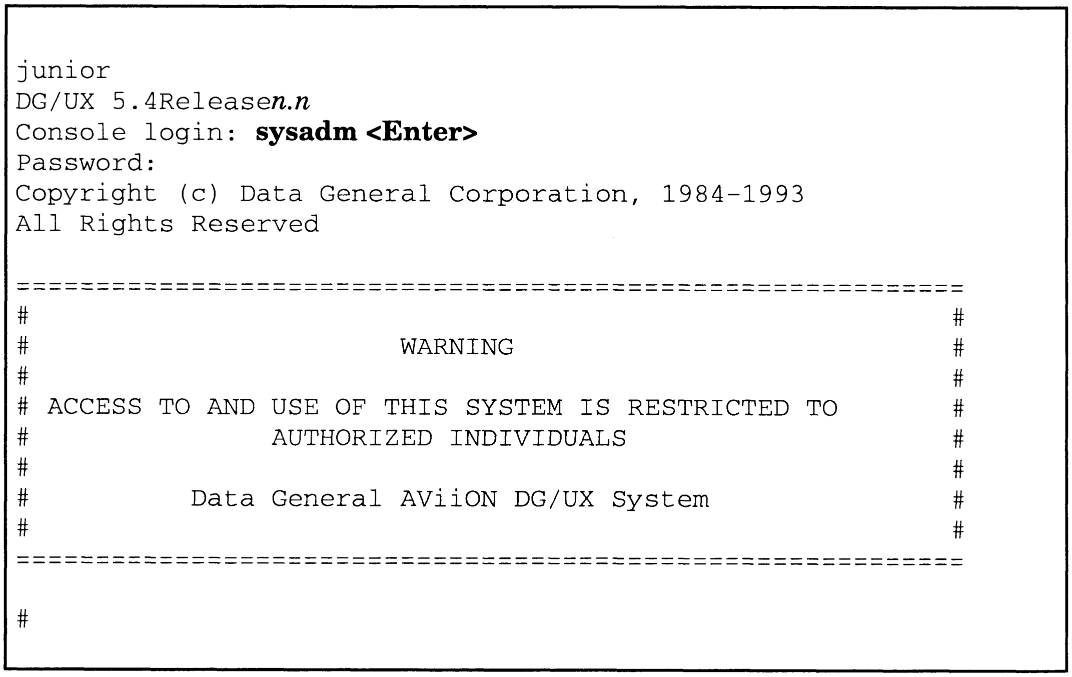
A single window and a login prompt appear.

Log in as **sysadm** to gain access to **sysadm** for continued customizing activities. You don't need a password. If you are ready to start your normal work, you must have a user account. Refer to Chapter 4 for information on adding user accounts, which includes assigning a password.

Logging in a console window

Log in as **sysadm** to gain access to **sysadm** for continued customizing activities. You don't need a password. If you are ready to start your normal work, you must have a user account.

Figure 7–5 shows the login display for a console window. The example shows user **sysadm** logging in to system **junior**. As shown, you may not need to enter a password.



```
junior
DG/UX 5.4Releasen.n
Console login: sysadm <Enter>
Password:
Copyright (c) Data General Corporation, 1984-1993
All Rights Reserved

============================================================
#                                                          #
#                        WARNING                           #
#                                                          #
# ACCESS TO AND USE OF THIS SYSTEM IS RESTRICTED TO        #
#                AUTHORIZED INDIVIDUALS                    #
#                                                          #
#           Data General AViiON DG/UX System               #
#                                                          #
============================================================

#
```

Figure 7–5 Logging in a Console Window

The superuser prompt (#) appears on your screen for continued system administration work. Refer to Chapter 4 for information on adding user accounts, which includes the assignment of a password.

What next?

You have added one or more diskless OS clients. If you want to add another type of client, go to the pertinent section in this chapter:

- Adding an OS client that has local **root** and **swap** virtual disks
- Adding an OS client that has a local **swap** virtual disk
- Adding an X terminal client

If you want to add any local devices, such as a printer (manual *Installing and Configuring Printers on the DG / UX™ System*) or disk drive, find the pertinent chapter using the task table in the Preface, "About this manual."

Adding an OS client that has a local root and swap virtual disk

This type of OS client has an attached disk drive that can provide its own local **root** and **swap** virtual disk OS resources (/ file system and swap space), while receiving its **usr** virtual disk OS resources (/**usr** file system) from the OS server via a LAN.

These procedures assume familiarity with the steps required to install a diskless OS client, and knowledge of **sysadm**. Before you begin these procedures, make sure that:

- The AViiON computer to be used as an OS server is running the DG/UX system (its kernel has booted to a run level of 3) and is connected to a network.
- You have created the required virtual disks to support OS clients and have mounted the required file systems. See Chapter 2 for a review of these virtual disks, sizes, and mount points. See Chapter 3 for procedures on creating the virtual disks and adding the file systems.
- The AViiON computer to be used as an OS client and its attached disk drive have been successfully powered up.

You will perform these steps:

- Complete the planning worksheet.
- Build a temporary OS client kernel.
- Add the OS client to the DG/UX release.
- Build an OS client kernel with local **root** and **swap** virtual disk resources.
- Load the / file system onto the OS client's local disk.
- Set up the software packages.
- Link the OS client's Internet address to the **/tftpboot** directory.

IMPORTANT: Examples illustrate the procedures documented in this section. Be sure that you provide the values that reflect your particular configuration.

Completing the planning worksheet

Figure 7–6 is a planning worksheet on which you should write information you will supply to **sysadm** prompts and system-level files when adding an OS client with a local **root** and **swap** virtual disk. You supply some items several times in different contexts.

Recording this data ahead of time accelerates the process. Not having this information on hand when **sysadm** requests it or when you need to type an entry in a system-level file will disrupt this procedure. Providing incorrect data may force you to abort the process and begin again.

Parameter Type	Example Value	Actual Value
OS client hostname	*junior*	
OS client Internet address*	*128.222.3.86*	
OS client Ethernet address†	*08:00:1B:18:03:11*	
OS client disk drive name	*sd(insc(0),0,0)*	
OS server hostname	*boss*	
OS server Internet address	*128.222.3.120*	
Temporary kernel name	*dgux.temp*	
NIS domain name	*my_domain*	
Do you subnet?	*yes*	
Network mask	*0xffffff00*	
OS client's Internet address (hexadecimal equivalent†)	*80DE0356*	

*The screen of the computer used as the OS client shows the Internet address in hexadecimal format the first time it boots its temporary kernel to a run level of 1. You can record its hexadecimal address at that point for later use. See the section "Loading the root (/) file system on the OS client's local disk" later for the procedure that produces that screen message.

†The Ethernet address appears when the system is powered on. You can record it then.

Figure 7–6 Local Root and Swap Virtual Disks Configuration Planning Worksheet

Building a temporary OS client kernel

To create a temporary system file, perform these procedures at the OS server.

1. Follow this path through **sysadm**:

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

Sysadm prompts

```
System configuration file name: [xxx]
```

2. Type **temp** and press Enter:

```
System configuration file name: [xxx] temp ↵
[system.temp] Correct? [yes]
```

3. **Sysadm** appends the name you type to the base filename **system** to yield the full name **system.temp**. Confirm with Enter:

```
[system.temp] Correct? [yes] ↵
Build for this host or for OS client(s) of this
      host: [this host]
```

4. Enter **OS client**:

```
Build for this host or for OS client(s) of this
host: [this host] OS Client ↵
Boot Device: [inen(0)]
```

5. Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0)**. Instead of **inen(0)**, you can specify a Data General network controller, named **dgen(*n*)**, where *n* is a number 0 through 5. For example,

```
Boot Device: [inen(0)]  dgen(0) ↵
Editor: [/usr/bin/vi]
```

6. Press Enter to accept the default editor, **/usr/bin/vi**.

```
Editor: [/usr/bin/vi] ↵
```

7. Search for the section in the file labeled "Typical AViiON OS client device configuration." Remove the comment indicator # from the line containing the disk drive name and any other I/O device attached to the OS client. The asterisk (*) is a pattern-matching metacharacter; it recognizes any SCSI ID. A sample section of the file follows in Figure 7–7; it shows the comment indicator removed from the SCSI disk and network controller names.

```
##### Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both.  Determine which situation applies to your system.

#      kbd()  # -- keyboard
#      grfx()               # -- graphics display
       sd(insc(),*)         # -- all SCSI disks on integrated SCSI adapter
#      st(insc(),*)         # -- all SCSI tapes on integrated SCSI adapter
       inen()               # -- integrated Ethernet controller
#      duart()              # -- integrated Duart terminal line controller
#      duart(1)             # -- second Duart (if present on system)
#      lp()                 # -- integrated printer controller (if present)
```

Figure 7–7   Sample AViiON OS Client Device Configuration

8. Save the file and quit the editor by typing **ZZ**.

   Sysadm prompts

   ```
   Link the new kernel for which clients:[all]
   ```

9. Do not link the new kernel to all clients. Instead, enter the OS client's name you gave when you created a diskless OS client earlier. For example, **junior**:

   ```
   Link the new kernel for which clients:[all] junior ↵
   ```

   Now, if a kernel with the name of this kernel already exists, **sysadm** will ask if you want to save the old kernel. Generally, to conserve disk space, we suggest you answer **no**. **Sysadm** prompts

   ```
   Continue with the build? [yes]
   ```

10. Press Enter to continue with the build:

    ```
    Continue with the build? [yes] ↵
    Configuring system...
    Building kernel...
    Successfully built dgux.temp.
    ```

    You have successfully built a temporary kernel named **dgux.temp**.

Adding the OS client to the DG/UX release

Perform these procedures at the OS server.

You will next create, modify, and select a client default set, which provides the default parameters that appear at the system prompts as you add the OS client. Then you will add the OS client to the release. Refer to the section "Adding an OS client to a release" earlier in this chapter for these procedures.

Supply the following data from the planning worksheet to the prompts:

- OS server hostname
- Temporary kernel name
- OS client hostname

By completing these procedures, you add an OS client to the DG/UX release.

Building an OS client kernel with local root and swap virtual disk resources

Perform these procedures at the OS server.

1. Follow this path through **sysadm** to create a new system file:

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

   Sysadm prompts

   ```
   System configuration file name: [temp]
   ```

2. Enter **local_root_swap**:

   ```
   System configuration file name: [temp] local_root_swap↲
   Build for this host or for OS client(s) of this
      host: [this host]
   ```

   The name you type is appended to the supplied base name **system** to yield the full name **system.local_root_swap**.

3. Enter **OS client**:

   ```
   Build for this host or for OS client(s) of this
   host: [this host] OS Client ↲
   Boot Device: [inen(0)]
   ```

4. Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0)**. Instead of **inen(0)**, you can specify a Data General network controller, named **dgen(*n*)**, where *n* is a number 0 through 5. For example,

   ```
   Boot Device: [inen(0)]  dgen(0) ↲
   Editor: [/usr/bin/vi]
   ```

5. Press Enter to accept the default editor **/usr/bin/vi**:

   ```
   Editor: [/usr/bin/vi] ↲
   ```

6. Search for the section in the file labeled "OS Client Configuration Variables." Comment out (insert a # in the first column position of) the lines containing the ROOTFSTYPE and SWAPDEVTYPE variables. At the end of the list, add the NETSTART variable and its assigned value **REAL_NET**. Be sure you type the variable name and value in uppercase letters. Use the Tab key for the desired alignment.

   ```
   OS Client Configuration Variables

   NETBOOTDEV          "inen()"
   # ROOTFSTYPE         NETWORK_ROOT
   # SWAPDEVTYPE        NETWORK_SWAP
   NETSTART             REAL_NET
   ```

7. Search for the section in the file labeled "Typical AViiON OS client device configuration." Remove the # from the line containing the disk drive name and any other I/O device attached to the OS client. The asterisk (*) is a pattern-matching metacharacter; it recognizes any SCSI ID. A sample section follows in Figure 7–8; it shows the comment indicator removed from the SCSI disk and network controller names.

```
##### Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both.  Determine which situation applies.

#       kbd()               # -- keyboard
#       grfx()              # -- graphics display
        sd(insc(),*)        # -- all SCSI disks on integrated SCSI adapter
#       st(insc(),*)        # -- all SCSI tapes on integrated SCSI adapter
        inen()              # -- integrated Ethernet controller
#       duart()             # -- integrated Duart terminal line controller
#       duart(1)            # -- second Duart (if present on system)
#       lp()                # -- integrated printer controller (if present)
```

Figure 7–8 Sample AViiON OS Client Device Configuration

8. After you finish editing the file, save it and quit the editor (**ZZ**). After several messages, **sysadm** prompts

   ```
   Link the new kernel for which clients: [all]
   ```

   IMPORTANT: Do not accept the default.

9. To the Link prompt, enter none:

   ```
   Link the new kernel for which clients: [all] none ↵
   Continue with the build? [yes]
   ```

10. Press Enter to accept the `yes` default response:

    ```
    Continue with the build? [yes] ↵
    ```

    You have successfully built a kernel named **dgux.local_root_swap**.

11. Quit **sysadm**.

12. From the shell, copy the **dgux.local_root_swap** kernel to **/usr/stand**:

    ```
    # cp /srv/release/PRIMARY/root/_Kernels/dgux.local_root_swap  /usr/stand↵
    ```

13. Export the **/usr** file system and temporarily give root permission to the OS client (here **junior**) with the following shell command:

    ```
    # exportfs -iv -o root=junior  /usr ↵
    ```

    The **–i** option causes the entries in the **/etc/exports** file to be ignored. The **–v** option displays the name of the exported directory. The **–o** option assigns exclusive root access to OS client **junior**. See the **exportfs**(8) man page for more information.

    IMPORTANT: Do not edit the **/etc/exports** file.

You have successfully built a kernel named **dgux.local_root_swap** that you will eventually boot from the OS client. This kernel provides local **root** and **swap** virtual disk resources. You have also exported the **/usr** file system (**usr** virtual disk resources) for the OS client's use.

Loading the root (/) file system on the OS client's local disk

Perform these procedures at the OS client.

1. Make sure the DG/UX system has been shut down. Your screen should display the SCM prompt. If it doesn't, shut down your system now using the following commands.

   ```
   # cd ↵
   # shutdown -g -y ↵
   # halt ↵
   SCM>
   ```

2. Boot the OS client's temporary kernel over the network to run level 1, typing the following command from the SCM prompt:

   ```
   SCM> b inen(0) -1 ↵          (Or other network controller name,
                                 such as dgen(0))
   ```

   Your screen displays booting messages as follows.

   ```
   Booting inen( ) -1
   Local Ethernet address is 08:00:1B:18:03
   Local Internet address is 128.222.3.86 or 80DE0356 hex
   .

   temp
   DG/UX Operating System       Release n
   Console Login:
   ```

3. The booting message includes the OS client's Internet address in hexadecimal (in this example, `80DE0356 hex`). Write this address on your planning worksheet; you need it to complete the procedure.

4. Log in as **sysadm**:

   ```
   Console Login: sysadm ↵
   ```

   The screen displays a shell prompt (#) indicating a successful login.

5. Set the TERM variable using the following Bourne shell commands:

   ```
   # TERM=vt100 ↵
   # export TERM ↵
   ```

6. Invoke **sysadm**:

   ```
   # sysadm ↵
   ```

7. If the attached disk is new, make sure it is soft formatted and registered. Use the **sysadm** physical disk software format, all soft formatting steps, explained in Chapter 9. The soft format automatically registers the disk. Skip to step 10.

8. If the disk has been used for data storage, this means it uses logical disk format. You must convert it to virtual disk format. To do this, use the **sysadm** sequence `Device -> Disk -> Physical -> Convert.`

9. If the disk has been used for data storage, make sure the disk has 90,000 free blocks. To determine free disk space, use the **sysadm** sequence `Device -> Disk -> Physical -> List` and select the normal viewing style.

10. Create the **root** virtual disk, create a file system, and specify a length of 40,000 blocks.

11. Create the **swap** virtual disk with a length of 50,000 blocks. You do not need to create a file system this virtual disk.

12. Quit **sysadm**.

13. From the shell, make a directory and mount the local / file system at a temporary mount point using the following shell commands:

```
# mkdir /mnt ↵
# mkdir /mnt/tmp_root ↵
# mount /dev/dsk/root /mnt/tmp_root ↵
```

14. Load the local / file system using the following commands:

```
# cd /usr/root.proto ↵
# tar –cf – . | (cd /mnt/tmp_root ; tar –xf –) ↵
```

   The copy process takes about one minute.

15. Copy the **dgux.local_root_swap** kernel into the OS client's local root directory using the following command:

```
# cp /usr/stand/dgux.local_root_swap /mnt/tmp_root ↵
```

16. Create a **hosts** file using the following commands:

```
# cd /mnt/tmp_root/etc ↵
# cp hosts.proto hosts ↵
```

17. Using **vi**, open and edit the **hosts** file. Add one line with the Internet address (that you copied from the booting message in step 3) and OS server hostname to the file. Use the Tab key for the desired alignment. For example, for the OS server **boss** shown earlier in the sample worksheet Figure 7–6, you would add

```
128.222.3.120     boss
```

   Type **ZZ** to save the file and quit the editor.

18. Create an **fstab** file using the following shell command:

    ```
    # cp fstab.proto fstab ↵
    ```

19. Using **vi**, open and edit the **fstab** file. First, comment out — insert the comment-indicating pound sign (#) in the first column of — the following line:

    ```
    # /dev/dsk/usr    /usr    dg/ux rw 1 0
    ```

    The preceding line specifies a locally mounted **/usr** file system.

    Next, insert this line to specify a remotely mounted **/usr** file system from the OS server. Use the Tab key for the desired alignment.

    ```
    boss:/usr  /usr   nfs  bg,ro,hard x 0
    ```

    Type **ZZ** to save the file and quit the editor.

    You have successfully loaded the / file system on the OS client's locally attached disk, and created the correct entry to remote mount the **/usr** file system from the OS server.

Setting up packages

Perform these procedures at the OS client:

1. Halt the DG/UX system by typing this shell command:

    ```
    # halt -q ↵
    ```

2. Boot the new kernel to run level **i**, supplying the correct boot path at the SCM> prompt:

    ```
    SCM> b sd(insc(0),0,0)root:/dgux.local_root_swap -i ↵
    ```

    The screen displays a series of booting messages during which you must confirm the current date and time. Booting leads automatically to the DG/UX package setup process; press Enter when queried to begin package setup.

    ```
    Booting sd(insc(),0)root:/dgux.local_root_swap -i
    DG/UX System Release n Bootstrap
    Loading image ........
    .
    Linking short names for /dev device nodes .....
    Starting disk daemons ....
    Mounting local file systems ......
    Starting installation steps ....

    Set up package(s)? [yes] ↵
    Package Name(s): [all] ↵
    OK to perform operation? [yes] ↵
    ```

3. Perform package setup (refer to the section “Setting up packages at the OS client” in this chapter for a description of the procedure). Use the data from the planning worksheet to answer package setup prompts.

   After you complete package setup, **sysadm** prompts

   ```
   Build kernel? [yes]
   ```

4. Answer **no**. (Otherwise, **sysadm** would build a custom kernel suitable for a diskless OS client.)

   ```
   Build kernel? [yes] no ↲
   Reboot now? [yes]
   ```

5. Do not reboot:

   ```
   Reboot now? [yes] no ↲

   local_root_swap
   DG/UX Operating System       Release n
   Console Login:
   ```

6. At the login prompt, enter the login name **sysadm**.

   ```
   local_root_swap
   DG/UX Operating System       Release n
   Console Login: sysadm ↲
   ```

   The screen displays a shell prompt (#) indicating a successful login.

7. Change the run level to **3**.

   ```
   # init 3 ↲
   ```

   A series of messages appears verifies transition to run level **3**.

8. Type the following command to link the new kernel to **/dgux** so that the correct kernel is rebooted each time you reboot your OS client.

   ```
   # admkernel -o link local_root_swap ↲
   ```

   Your OS client is now using a local / file system and swap space, and a remote **/usr** file system.

Linking the OS client Internet address to the /tftpboot directory

You will delete the OS client name and then link its Internet address to the **tftpboot** directory. Perform these steps at the OS server:

1. From **sysadm**, follow this path to delete the OS client.

   ```
   Client -> OS Client -> Delete
   ```

2. Specify the OS client hostname (for example, **junior**) and press Enter.
3. Quit **sysadm**.
4. From the shell, link the OS client's Internet address to the **/tftpboot** directory using the following commands. Use the data from the planning worksheet for the OS client's Internet Address in hexadecimal format. Be sure to use uppercase letters when specifying the Internet address. An example that shows the Internet address 80DE0356 follows.

```
# cd /tftpboot ↲
# ln -s /usr/stand/boot.aviion 80DE0356 ↲
# initrarp ↲
```

This link maintains OS client address entries that are resolved and used each time the OS client boots. The OS client (with a local / file system and swap space) fails to boot if this entry is not present in **/tftpboot** on the OS server.

What next?

You have added an OS client that has a local root and swap virtual disk. To add another such OS client, repeat the steps in this section "Adding an OS client that has a local root and swap virtual disk." If you want to add another type of client, go to the pertinent section in this chapter:

- Adding a diskless OS client
- Adding an OS client that has a local **swap** virtual disk
- Adding an X terminal client

If you want to add any local devices, such as a printer (manual *Installing and Configuring Printers on the DG / UX™ System*) or disk drive (this manual, Chapters 2 and 3), find the pertinent chapter using the task table in the Preface, "About this manual."

IMPORTANT: Because an OS client cannot load and set up a software package that resides in the **/usr** file system, the software packages must be located at the OS server.

Adding an OS client that has a local swap virtual disk

This type of OS client has an attached disk drive that can provide its own local **swap** virtual disk OS resources while receiving its **root** and **usr** virtual disk OS resources (/ and **/usr** file systems) from the OS server over a LAN.

Before you start

These procedures assume the setup of a diskless OS client. You will modify the diskless OS client configuration (the entire OS resides remotely on the OS server's attached disk drive) so that the OS client continues to get its / and **/usr** file systems from the OS server, but its swap resources will be put on the OS client's attached disk drive.

IMPORTANT: If you have not already created a diskless OS client, return to the section "Adding a diskless OS client" early in this chapter and complete its instructions; then return to this section for continued instructions.

Before you begin these procedures, make sure that:

- The AViiON computer to be used as an OS server is running the DG/UX system (its kernel has booted to a run level of 3) and is connected to a network.
- The computer system to be used as an OS client and its attached disk drive have been successfully powered up.
- The `SCM` prompt is displayed on the screen of the AViiON computer to be used as an OS client.
- You know the name of the disk drive attached to the OS client. The example used in this section is **sd(insc(0),3,0)**.

You will perform these steps:

- Build a temporary diskless OS client kernel.
- Boot the temporary kernel to a run level of 1.
- Build and boot an OS client kernel with local **swap** virtual disk resources.
- Set up packages and log in.

Building a temporary OS client kernel

Perform these procedures at the OS server:

1. Follow this path through **sysadm** to create a temporary system file.

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

Sysadm prompts

```
System configuration file name: [xxx]
```

2. Enter **temp**:

```
System configuration file name: [aviion] temp ↵
Build for this host or for OS client(s) of this
    host: [this host]
```

The name you type is appended to the supplied base filename **system** to yield the full name **system.temp**.

3. Enter **OS client**:

   ```
   Build for this host or for OS client(s) of this
   host: [this host] OS Client ↲
   Boot Device: [inen(0)]
   ```

4. Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0)**. Instead of **inen(0)**, you can specify a Data General network controller, named **dgen(*n*)**, where *n* is a number 0 through 5. For example,

   ```
   Boot Device: [inen(0)]  dgen(0) ↲
   Editor: [/usr/bin/vi]
   ```

5. Press Enter to accept the default editor, **/usr/bin/vi**:

   ```
   Editor: [/usr/bin/vi] ↲
   ```

6. Search for the section in the file labeled "Typical AViiON OS client device configuration." Remove the # from the line containing the disk drive name and any other I/O device attached to the OS client. The asterisk (*) is a pattern-matching metacharacter; it recognizes any SCSI ID. A sample section follows in Figure 7–9; it shows the comment indicator removed from the SCSI disk and network controller names.

```
##### Typical AViiON OS client device configuration

# Note that a system can have an lp() controller or a second duart()
# but not both.  Determine which situation applies.

#       kbd()                 # -- keyboard
#       grfx()                # -- graphics display
        sd(insc(),*)          # -- all SCSI disks on integrated SCSI adapter
#       st(insc(),*)          # -- all SCSI tapes on integrated SCSI adapter
        inen()                # -- integrated Ethernet controller
#       duart()               # -- integrated Duart terminal line controller
#       duart(1)              # -- second Duart (if present on system)
#       lp()                  # -- integrated printer controller (if present)
```

Figure 7–9 Sample AViiON OS Client Device Configuration

7. Save the file and quit the editor by typing **ZZ**. If asked about continuing, answer yes.

   Sysadm prompts

   ```
   Link the new kernel for which clients:[all]
   ```

8. Enter the OS client's hostname, used when you created a diskless OS client. For example, **junior**:

   ```
   Link the new kernel for which clients:[all] junior ↲
   Save the old kernel? [yes]
   ```

9. If you can afford the disk space, we recommend you accept the **yes** default response.

```
Save the old kernel? [yes] ↵
Continue with the build? [yes]
```

10. To continue the build process, press Enter:

```
Continue with the build? [yes] ↵
Configuring system...
Building kernel...
Successfully built dgux.temp
```

You have successfully built a temporary kernel named **dgux.temp**. Continue to the next section.

Booting the temporary kernel, creating an OS kernel with local swap resources, and setting up packages

Perform these procedures at the OS client.

1. Make sure that your system has been shut down. Your screen should display the SCM prompt. If it doesn't, shut down your system now using these commands:

```
# cd / ↵
# shutdown -g0 -y ↵
# halt ↵
SCM>
```

2. Boot the OS client's temporary kernel over the network to run level 1:

```
SCM> b inen(0) -1 ↵
```

Your screen displays a series of booting messages:

```
Booting inen( ) -1
Local Ethernet address is 08:00:1B:18:03
Local Internet address is 128.222.3.86 or 80DE0356 hex
...
```

A log-in prompt appears at the conclusion of the boot process. Log in as **sysadm**. The screen displays a shell prompt (#), indicating a successful login.

```
temp
DG/UX Operating System        Release n
Console Login: sysadm ↵
```

3. Set the TERM variable using the following Bourne shell commands:

```
# TERM=vt100 ↵
# export TERM ↵
```

4. Invoke **sysadm**:

   ```
   # sysadm ↲
   ```

5. If the attached disk is new, make sure it is soft formatted and registered. Use the **sysadm** physical disk software format, all soft formatting steps, explained in Chapter 9. The soft format automatically registers the disk. Go to step 8.
6. If the disk has been used for swap storage, this means it uses logical disk format. You must convert it to virtual disk format. To do this, use the **sysadm** sequence `Device -> Disk -> Physical -> Convert.`
7. If the disk has been used for data storage, make sure the disk has 50,000 free blocks. To determine free disk space, use the **sysadm** sequence `Device -> Disk -> Physical -> List` and select the normal viewing style.
8. Create the **swap** virtual disk with a length of 50,000 blocks. You do not need to create a file system this virtual disk.
9. Quit **sysadm**.
10. Using **vi**, edit the **/etc/fstab** file. Insert a pound sign (#) in the first column of the line that begin with "server–hostname," to produce the following line:

    ```
    # server–hostname:/srv/swap/client–hostname  swap  nfs sw x 0
    ```

    The preceding line specifies a remotely mounted swap space from the OS server.
11. Next, insert the following line to specify a locally mounted swap area.

    ```
    /dev/dsk/swap  swap_area  swap  sw 0 0
    ```

12. Type **ZZ** to save the file and quit the editor.
13. Follow this path through **sysadm** to create another system file:

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

    Sysadm prompts

    ```
    System configuration file name: [xxx]
    ```

14. Enter **local_swap**:

    ```
    System configuration file name: [xxx] local_swap ↲
    Build for this host or for OS client(s) of this
        host: [this host]
    ```

    The name you type is appended to the supplied base name **system** to yield the full name **system.local_swap**.
15. Type **this host** and press Enter:

    ```
    Build for this host or for OS client(s) of this
    host: [this host] this host ↲
    Editor: [/usr/bin/vi]
    ```

16. Press Enter to accept the default editor, **vi**:

    ```
    Editor: [/usr/bin/vi] ↲
    ```

17. Search for the section in the configuration file labeled "OS Client Configuration Variables." Find the SWAPDEVTYPE variable and change its assigned value from NETWORK_SWAP to **LOCAL_SWAP**.

    ```
    OS Client Configuration Variables

    NETBOOTDEV        "inen()"
    ROOTFSTYPE        NETWORK_ROOT
    SWAPDEVTYPE       LOCAL_SWAP
    ```

18. Save the file and quit the editor by typing **ZZ**.

    After several process messages appear, **sysadm** prompts

    ```
    Link the new kernel to /dgux? [yes]
    ```

19. Link the new kernel to **/dgux** by answering yes:

    ```
    Link the new kernel to /dgux? [yes] ↲
    Continue with the build? [yes]
    ```

20. Continue the build process by pressing Enter:

    ```
    Continue with the build? [yes] ↲
    ```

    You have successfully built a kernel named **/dgux.local_swap** that has been linked to **/dgux**.

21. Using **sysadm** `System -> Kernel -> Reboot`, verify the boot path and boot the new kernel. Confirm your desire to reboot.

    ```
    Boot path: [inen(0) -3] ↲
    All currently running processes will be killed.
    Are you sure you want to reboot the system? [yes] ↲
    ```

    The screen displays a series of booting messages during which you must confirm the current date and time. Booting leads automatically to the DG/UX package setup process.

    ```
    Booting inen( ) -i
    DG/UX System Release n Bootstrap
    Loading image ........
    .
    Linking short names for /dev device nodes .....
    Starting disk daemons ....
    Mounting local file systems .......
    Starting installation steps ....
    Set up packages [yes]
    ```

    IMPORTANT: Because it is located at the OS server, an OS client cannot load and set up a software package that resides in the **/usr** file system.

22. Respond to the **sysadm** package prompts as follows.

```
Set up package(s)? [yes] ↵
Package Name(s): [all] ↵
OK to perform operation? [yes] ↵
```

Perform package setup (explained in Chapter 6, section "Setting up packages"). Use the information from the OS Client Package Setup Worksheet that you used to set up the OS client as a diskless OS client.

After you complete package setup, **sysadm** prompts

```
Build kernel? [yes]
```

23. Answer **no**. (Otherwise, sysadm would build a custom kernel suitable for a diskless OS client.) For example,

```
Build kernel? [yes] no ↵
Reboot now? [yes]
```

24. Do not reboot:

```
Reboot now? [yes] no ↵

local_swap
DG/UX Operating System      Release n
Console Login:
```

25. At the login prompt, log in using the **sysadm** login name:

```
local_swap
DG/UX Operating System      Release n
Console Login: sysadm ↵
```

The screen displays a shell prompt (#), indicating a successful login.

26. Change the run level from **i** to **3**:

```
# init 3 ↵
```

A series of messages verifies transition to run level 3.

27. Type the following command to link the new kernel to **/dgux** so the correct kernel reboots each time you reboot your OS client.

```
# admkernel -o link local_swap ↵
```

Your OS client now uses a local swap space and a remote / and **/usr** file system.

What next?

You have added an OS client that has a local swap virtual disk. To add another such client, repeat the steps in this section "Adding an OS client that has a local swap virtual disk." If you want to add another type of client, go to the pertinent section in this chapter:

- Adding a diskless OS client
- Adding an OS client that has local **root** and **swap** virtual disks
- Adding an X terminal client

If you want to add any local devices, such as a printer (manual *Installing and Configuring Printers on the DG / UX™ System*) or disk drive, find the pertinent chapter using the task table in the Preface, "About this manual."

Adding an X terminal client

An X terminal client is a type of graphics terminal that provides X Window System graphics support. Lacking its own disk drive and operating system, the X terminal client depends on an OS server for its bootstrap.

In this section, you perform these steps:

- Complete the planning worksheet
- Add an X terminal client to the **hosts** database
- Add an X terminal client to the **ethers** database
- Attach an X terminal client to its bootstrap file

Network planning for X terminal clients

An X terminal client network planning worksheet appears after this section to help you prepare for installation. It lists network parameters for which you need to supply values. Completing the worksheet before you begin the installation procedures for clients will speed up the installation process.

If you have experience installing the DG/UX system or other operating systems, you may prefer to go directly to the X terminal client network planning worksheet.

The information necessary for completing the planning worksheet follows:

Host name	A unique name you assign to your X terminal hardware that can contain up to 31 alphabetic and numeric characters.

However, you are advised to keep the names short. Host names that relate to the use or location of the system are particularly helpful in networked environments where hosts may share file systems. Examples of hostnames are **fred**, **jamaica**, and **writer_doc**. Do not use the capitalized names **MY_HOST** or **PRIMARY**; the system reserves these names.

Internet address

The network system administrator provides the Internet address for the host being setup. An example of an Internet address is **128.223.2.1**. For more information on Internet addresses, see *Managing TCP/IP on the DG/UX™ System.*

Ethernet address

Host address that is unique to the particular hardware. This address is preset at the factory. It consists of six 2-digit (hexadecimal) fields separated by colons in the form *nn:nn:nn:nn:nn:nn*. Refer to your hardware documentation for information on determining your Ethernet address.

X terminal client network planning worksheet

Figure 7–10 is a worksheet in which you record the Internet address and Ethernet address for each X terminal client on your system.

X Terminal Client Hostname	Internet Address	Ethernet Address
Example: xterm1	*Example: 123.227.3.15* *(use periods)*	*Example: 08:00:1B:03:32:44* *(use colons)*

Figure 7–10 X Terminal Client Network Planning Worksheet

Adding an X terminal client to the OS server's hosts database (/etc/hosts)

The **hosts** database contains a list of Internet address and X terminal client hostname pairs to allow network communication between the OS server and the X terminals.

1. To add an entry to the **hosts** database, follow this path through **sysadm** at the OS server.

```
Networking -> TCP/IP -> Databases -> Hosts -> Add
```

The **sysadm** prompt depends on whether your host is an OS server that has also been declared the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. If your host is the NIS master, **sysadm** displays

```
Hosts database to use: [NIS (YP)]
```

If your server is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Hosts database will be used.` **Sysadm** will let you insert a new entry in the *local* **hosts** database for each OS client. Skip to step 3.

2. On the NIS master, press Enter to accept the default value **NIS**:

```
Hosts database to use: [NIS (YP)] ↵
```

3. **Sysadm** displays

```
Host Name:
```

4. Consult your OS Client Network Planning Worksheet that you completed and enter the hostname; for example,

```
Host Name: xterm1 ↵
Internet Address:
```

5. From your OS Client Network Planning Worksheet, enter the four-field (four octet) internet address. For example,

```
Internet Address: 123.228.2.15 ↵
Alias List:
```

6. You may specify an alternate name (alias) for your host. For example, if your department uses long hostnames, you may choose a shorter alias for convenience. For example,

```
Alias List: x1 ↵
OK to perform operation?  [yes]
```

7. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example,

```
OK to perform operation?  [yes] ↵
xterm1 has been added.
```

Sysadm has added the X terminal to the appropriate hosts file. Repeat the steps above for each X terminal client you want to add to the **hosts** database.

Adding an X terminal client to the OS server's Ethernet database (/etc/ethers)

The **ethers** database contains a list of Ethernet address and OS client hostname pairs to allow network communication between the

OS server and its X terminal clients. To add an entry to the **ethers** database, perform these steps at the OS server.

1. Follow this path through **sysadm**:

   ```
   Networking -> TCP/IP -> Databases -> Ethers -> Add
   ```

   The **sysadm** prompt depends on whether your host is an OS server that has also been declared as the NIS master. Declaring an OS server as the NIS master is explained in *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.* If your host is the NIS master, **sysadm** displays

   ```
   Ethers database to use: [NIS (YP)]
   ```

   If your server is not the NIS master, **sysadm** displays the message `This host is not the NIS master. Only the local Ethers database will be used.` **Sysadm** will let you insert a new entry in the *local* **ethers** database for each OS client. Skip to step 3.

2. For a networked environment, press Enter to accept the default value **NIS**:

   ```
   Hosts database to use: [NIS (YP)] ↲
   ```

3. **Sysadm** displays

   ```
   Host Name:
   ```

4. Consult the OS Client Network Planning Worksheet that you completed and enter the hostname; for example,

   ```
   Host Name: xterm1 ↲
   Internet Address:
   ```

5. From your OS Client Network Planning Worksheet, enter the Ethernet address, form *hh:hh:hh:hh:hh* where *h* is a hexadecimal number. For example,

   ```
   Ethernet Address: 08:00:1B:03:32:44 ↲
   OK to perform operation?  [yes]
   ```

6. Review your answers. If they are correct, press Enter to confirm; if not, enter **n** and correct the wrong entries. For example,

   ```
   OK to perform operation?  [yes] ↲
   xterm1 has been added.
   ```

   Sysadm has added the X terminal to the appropriate **ethers** file. Repeat this procedure to add as many X terminals as desired to the **ethers** database.

Attaching the X terminal client to its bootstrap file

To attach an X terminal client to its bootstrap file, follow these steps at the OS server.

1. Follow this path through **sysadm** at the OS server.

   ```
   Client -> X Terminal -> Add
   ```

   Sysadm prompts

   ```
   Client Host Name:
   ```

2. Specify the hostname of the X terminal client. For example,

   ```
   Client Host Name: xterm1 ↵
   Bootstrap File: [/usr/opt/X11/xtd/avx30boot]
   ```

3. **Sysadm** will create a link to the bootstrap file for this X terminal client. Press Enter to accept the default bootstrap file pathname. For example,

   ```
   Bootstrap File: [/usr/opt/X11/xtd/avx30boot] ↵
   OK to perform operation? [yes]
   ```

4. If you are happy with the answers, confirm with Enter:

   ```
   OK to perform operation? [yes] ↵
   Xterminal xterm1 has been added.
   ```

   You have added an X terminal client and attached it to the bootstrap file. The following actions take place when you turn on your X terminal client.

   - The X terminal issues an **rarp** request on the network. The host you have set up matches the **ethers** address that is part of the **rarp** request to the X terminal hostname in the **ethers** database.

   - The host matches the X terminal hostname to the Internet address in the **hosts** database. The host, in response to the **rarp** request, returns the Internet address to the X terminal client.

   - Finally, the X terminal client uses its Internet address to access the correct file in the **/tftpboot** and downloads its Xserver.

   After the X terminal client's Xserver loads, you are able to access the X terminal's configuration menus. For more information, refer to the AVX-30 Release Notice in the **/usr/opt/X11/release** directory.

   For information on adding a user account for the X terminal client user and assigning a password, refer to Chapter 4.

Where to go next

After adding network OS clients and/or X terminals, you might simply want to run the DG/UX system. You can find information on common DG/UX administrative tasks as described in the task table included in the Preface, "About this manual."

End of Chapter

8 Adding a secondary operating system release

A secondary release is a different release of DG/UX (or another vendor's UNIX system) installed on an OS server's disk for any clients that have special requirements. Secondary releases are designed to run on clients and primary releases to run on the server. But you can install and run the current release of DG/UX as a secondary release. The number of secondary releases allowed is restricted only by disk space and system performance.

The OS clients that use a secondary release need not employ the same hardware architecture as each other or as the OS server. Though DG/UX system software is the most common type of release, you can add operating systems intended for hardware other than AViiON systems. To install a foreign OS in the secondary release area, you must consult the documentation and release notice of the foreign release.

This chapter explains how to load and install DG/UX 5.4 Release 3.00 as a secondary release. Major sections proceed as follows.

- Installing a previous release of DG/UX as a secondary release
- Installing DG/UX 5.4 Release 3.00 as a secondary release
- Where to go next

IMPORTANT: The instructions in this chapter assume that you know how to install a primary release of the DG/UX system.

Installing a previous release of DG/UX as a secondary release

If you want to install a previous release of DG/UX as a secondary release, you can use the DG/UX software and documentation for the kernel that's running on the server. The server can be running DG/UX 5.4 Release 3.00 or a previous release such as 5.4 Release 2.10. Using the server's DG/UX system and **sysadm**, create a secondary release area, install the release there, and then build one or more kernels for clients who want the secondary release.

Generally, only a client can boot a DG/UX system that's in a secondary release area; the server cannot boot such a system even though it may have built the system.

Chapter 2 shows a previous release of DG/UX installed as a secondary release.

Installing DG/UX 5.4 Release 3.00 as a secondary release

You can install DG/UX 5.4 Release 3.00 as a secondary release on a server that's running any recent release of DG/UX (such as 5.4 Release 3.00 or 5.4 Release 2.10). If the server is running a previous release, for **sysadm** details you may want to use the manual *Customizing the DG / UX™ System* for that release.

Here is a summary of the tasks involved. Details on each task follow.

- Create logical or virtual disks and add file systems to hold the DG/UX 5.4 Release 3.00 software.
- Create the secondary release area and load it with DG/UX 5.4 Release 3.00 from the 5.4 Release 3.00 media.
- Provide OS clients with an installer kernel and add OS clients to the secondary release.
- Write-enable the **/usr** and **/usr/opt/X11** file systems.
- Boot the installer kernel, add file systems, and set up packages.
- Customize the OS client environment by creating a custom 5.4 Release 3.00 kernel for the clients that want to use it.

Creating logical or virtual disks and adding file systems for DG/UX 5.4 Release 3.00

If the server is running a release of DG/UX before 5.4 Release 3.00, you will create logical disks for the 5.4 Release 3.00 software. If the server is running a DG/UX 5.4 Release 3.00 or later, you will create virtual disks for the 5.4 Release 3.00 software. The OS clients will use the secondary release over the network and the disk format (logical or virtual) is not significant to them.

For the sake of example, this section assumes the server is running DG/UX 5.4 Release 2.01 and it assumes the following values:

Secondary release area name	**dgux_54R300**
OS server hostname	**server_dgux_54R201**
OS client host name	**client_dgux_54R300**

Perform these procedures at the OS server, using the server's standard operating system.

1. Make sure you have enough disk space before you start. Table 8–1 shows the default sizes and mount-point directories for the required and optional logical and virtual disks for DG/UX 5.4 Release 3.00. You need about 615,000 blocks to add just one OS client to the DG/UX 5.4 Release 3.00 secondary release.

   For more information on secondary release area sizes, refer to the section "Multiple release areas" in Chapter 2. If the server is running a previous release of DG/UX, then mentally substitute "logical disk" for "virtual disk," since you will be creating *logical* disks of the sizes given in Chapter 2.

Table 8–1 Logical and Virtual Disk Sizes and Mount Points for DG/UX 5.4 Release 3.00 as a Secondary Release

Logical or Virtual Disk Name	Mount Point Directory	Size in 512–Byte Blocks	For One OS Client
srv_swap*	/srv/swap	(50,000 * *number–of–clients*)	50,000
srv_dump	/srv/dump	33,000	33,000
root_dgux_54R300	/srv/release/dgux_54R300/root	(*number–of–clients* * (40,000 – kernel-size)) + kernel-size)	40,000
usr_dgux_54R300	/srv/release/dgux_54R300/usr	240,000	240,000
usr_opt_X11_dgux_54R300	/srv/release/dgux_54R300/usr/opt/X11	140,000	140,000
usr_opt_networker_54R300	/srv/release/dgux_54R300/usr/opt/networker	50,000	50,000
usr_opt_xdt_54R300	/srv/release/dgux_54R300/usr/opt/xdt	50,000	50,000
* If you already have OS clients attached to a primary release, the srv_swap disk has already been created and its file system mounted; you do not need to re-create srv_swap.			

2. Use **sysadm** to create the logical or virtual disks, with file systems on them, and sizes calculated from the table shown above.

3. Use **sysadm** to add file systems on the logical disks you just created `(File System -> Local Filesys -> Add)`. Adding a file system automatically adds its entry and mount point in the file system table (**/etc/fstab**).

Creating a secondary release area

This operation creates the directory structure to hold a secondary software release. To create a release, you supply a release area name as well as pathnames identifying the locations of the **/usr** file system, the swap area shared by OS clients, root areas, and any other shared software. This operation creates the release area structure, but the release area remains empty until you load it.

1. At the OS server, follow this path through **sysadm** to create a secondary release area.

   ```
   Software -> Release Area -> Create
   ```

   Sysadm prompts

   ```
   Release Area Name:
   ```

2. Type a unique name for the release you are adding. For example,

   ```
   Release Area Name: dgux_54R300 ↵
   Client Root Parent Directory: [/srv/release/dgux_54R300/root]
   ```

3. A default pathname for the OS client parent directory is the location for the root directories of the OS clients of this release. When you add an OS client to this release, the prototype host-dependent (/) directory structure is copied for the OS client. We suggest the default:

   ```
   Client Root Parent Directory: [/srv/release/dgux_54R300/root]↵
   /usr Directory: [/srv/release/dgux_54R300/usr]
   ```

4. Specify the pathname of the host-independent **/usr** directory for this release. Only one host-independent directory is created for each release because all OS clients of a release share the **/usr** directory. Again, we suggest the default:

   ```
   /usr Directory: [/srv/release/dgux_54R300/usr] ↵
   Share Directory: [/srv/share]
   ```

5. The **share** directory is any directory containing software the OS clients of this release share. If the directory does not exist, the operation creates it. For a view of the **/srv/release/dgux_54R300** file system, refer to Chapter 2, Figure 2–3. Take the default for the **share** directory:

   ```
   Share Directory: [/srv/share] ↵
   Swap Directory: [/srv/swap]
   ```

6. Again, take the default for the **swap** directory:

   ```
   Swap Directory: [/srv/swap] ↵
   OK to perform operation?  [yes]
   ```

7. Review your answers and, if they are correct, confirm by pressing Enter:

   ```
   OK to perform operation?  [yes]↵
   ```

   ```
   Release dgux_54R300 has been added.  You may now
   load software into this release area using the
   Package management operations.
   ```

   You have created a secondary release area.

Loading DG/UX 5.4 Release 3.00 into the secondary release area

Obtain the 5.4 Release 3.00 release tape before you start these procedures. Perform the procedures at the OS server, still using the server's **sysadm**.

Follow this path through **sysadm**:

```
Software -> Package -> Load
```

Load the package into the secondary release area. You will need to supply the correct release area name; for example, **dgux_54R300**.

Providing OS clients with an installer kernel

Before OS clients can be added to the secondary release, an installer (first-time) diskless kernel must be available for booting. The kernel that performs this role is just the 5.4 Release 3.00 stand-alone **sysadm** kernel, copied to the default diskless kernel filename (**dgux.diskless**) and booted with the **–D** option. At the OS server, exit from **sysadm** and enter the following shell commands:

```
# cd /srv/release/dgux_54R300 ↵
# mkdir root/_Kernels ↵
# cp usr/stand/sysadm root/_Kernels/dgux.diskless ↵
```

Adding OS clients to the secondary release

1. Refer to the section "Network planning for OS clients" in Chapter 7 and complete the planning worksheets.
2. Refer to the section "Adding OS clients to the OS server's network databases" in Chapter 7 for instructions on entering the OS clients in the **/etc/hosts** and **/etc/ethers** files.
3. Refer to the section "Adding an OS client to a release" in Chapter 7 for instructions on attaching the OS clients to the secondary release. Be sure to specify the correct release area, kernel pathname, and bootstrap file. Sample values follow.

Release area	**dgux_543**
Kernel pathname	**/srv/release/dgux_543/root/_Kernels/dgux.installer.diskless**
Bootstrap file	**/srv/release/dgux_543/usr/stand/boot.aviion**

Making the /usr and /usr/opt/X11 file systems exportable

At the OS server, perform these procedures for each OS client you want attached to the secondary release. Substitute the client's name for **client_dgux_54R300** in the commands we show. The procedures performed for the first OS client you add vary slightly from the procedures performed for the subsequent OS clients.

1. Modify the **/etc/exports** file to add the secondary release area's **usr** file system and make the **usr** file system readable and writable by entering the following command to **sysadm**:

   ```
   # admfilesystem –omodify –eP ”–root=client_dgux_54R300” \↵
     –p ”rw”  /srv/release/dgux_54R300/usr ↵
   ```

2. Modify the **/etc/exports** file to add the secondary release area's **usr/opt/X11** file system much as in step 1 by entering the following command to **sysadm**:

   ```
   # admfilesystem –omodify –eP ”–root=client_dgux_54R300” \↵
      /srv/release/dgux_54R300/usr/opt/X11 ↵
   ```

3. Export the secondary release's file systems by entering the following shell command:

   ```
   # exportfs –va ↵
   exported /srv/release/dgux_543/usr
   ```

Booting the installer kernel, adding file systems, and setting up the DG/UX 5.4 Release 3.00 packages

Perform these procedures at each OS client.

1. Boot the OS client's diskless installer kernel to a run level of **i** by booting from the server. For example,

   ```
   SCM> b inen( ) –D –i ↵   (Or other network controller name,
                             such as dgen(0))
   ```

2. Refer to the section "Network planning for OS clients" in Chapter 7 and complete the planning worksheets.

3. Set up the DG/UX 5.4 Release 3.00 software packages in the 5.4 Release 3.00 release area. In Chapter 7, in the appropriate section for your OS client, see "Setting up packages" for instructions on setting up packages. **Sysadm** will ask for a release area and you must specify the pathname you gave the DG/UX 5.4 Release 3.00 release area instead of the PRIMARY area. In the examples in this chapter, this pathname is shown as **/srv/release/dgux_54R300**.

4. Add the other file systems that are on the OS server by entering the following commands. Enter

```
# admfilesystem –oadd –p ”rw” –f \ ↵
server_dgux_54R300/srv/release/dgux_54R300/usr/opt/networker \ ↵
/usr/opt/networker ↵
```

5. Enter

```
# admfilesystem –oadd –p ”rw” –f \ ↵
server_dgux_54R300/srv/release/dgux_54R300/usr/opt/xdt \ ↵
/usr/opt/xdt ↵
```

6. Enter

```
# admfilesystem –omount  /usr/opt/networker  /usr/opt/xdt ↵
```

7. Build a custom kernel for the OS client and boot it as explained in Chapter 10.

8. Log in the DG/UX system as explained in Chapter 11.

   You have added an OS client to DG/UX 5.4 Release 3.00 in the secondary release area.

Customizing the OS client environment

After each OS client boots, continue customizing activities as desired; for example, by building one or more DG/UX 5.4 Release 3.00 kernels for them as explained in Chapter 10.

Where to go next

The following chapter explains how to add mass storage devices. Finding information on this and other common DG/UX administrative tasks is described in the task table included in the Preface, “About this manual.”

End of Chapter

9 Adding mass-storage devices

This chapter outlines the steps needed to format, configure, and use disk, diskette, and tape drives. Major sections proceed as follows.

- Standard and nonstandard devices
- Building and booting a kernel
- Soft formatting disk drives
- Using removable media devices
- Learning device filenames while DG/UX is running
- Where to go next

Before using the procedures explained in this chapter, complete hardware installation of the device you want to add. You can add the following mass-storage devices to your hardware configuration at any time.

- Winchester hard disk drive (in a system cabinet, disk-array storage system, or CSS/2 or CSS/3 subsystem)
- CD-ROM (Compact Disk-Read Only Memory)
- Multiple-read/write magneto-optical disk
- Diskette drive
- Tape drive

IMPORTANT: Before proceeding, make sure that all device power cables are connected. If a device is disconnected, power down the system before you connect the device, and then reboot the system.

Standard and nonstandard devices

To configure, format, and use a mass-storage device, you need the device's DG/UX device name. There are standard and nonstandard mass-storage devices.

A standard device is one you purchased from Data General whose device ID is preset through a jumper or DIP switch setting at the factory. Such a setting corresponds to a DG/UX device name. The file **/usr/etc/probedevtab** lists the standard mass-storage device names. Appendix B explains standard device name syntax.

Examples of standard DG/UX device names follow.

st(insc(0),5) SCSI tape device on the integrated SCSI controller [**insc(0)**] at SCSI ID 5.

sd(insc(0),6,0) SCSI disk device on the integrated SCSI controller [**insc(0)**], SCSI ID 6, at logical unit number 0.

sd(dgsc(0,7),0,1) Disk-array storage system SCSI disk device on the first Data General SCSI controller [**dgsc(0)**], SCSI ID 7 (7 is the default, so we could omit it, but we include it for clarity), SP A, at logical unit number 1.

sd(dgsc(0),1,4) Disk-array storage system SCSI disk device on the first Data General SCSI controller [**dgsc(0)**], controller SCSI ID of 7 omitted, SP B, at logical unit number 4.

sd(dgsc(vme(1),0),1,4) Disk-array storage system on the *second VME channel* [**vme(1)**], device on the Data General SCSI controller [**dgsc**... **(0)**], controller SCSI ID of 7 omitted, SP B, at logical unit number 4. Only a few AViiON computer models allow multiple VME channels. For all computers that use VME, **probedev** generates the device entry **vme(*)** that lets each kernel implicitly supports all VME channels.

sd(insc(0),3,6) SCSI diskette drive on the integrated SCSI controller [**insc(0)**], SCSI ID 3, logical unit 6 (seventh drive on SCSI ID 3).

sd(cisc(0),2,4) SCSI magneto-optical device on the Ciprico SCSI adapter [**cisc(0)**], SCSI ID 2, logical unit 4 (fifth drive on SCSI ID 2).

da(hada(0),6) Disk array (H.A.D.A.-type 30-disk subsystem), hada controller [**hada(0)**], logical unit number 6.

nvrd() Non-volative RAM (NVRAM) memory board.

Nonstandard I/O devices include

- Any I/O device you add to your configuration that does not have a standard device name
- Any standard I/O device you rejumper to a nonstandard setting
- Any I/O device purchased from a third-party vendor

An example of a nonstandard device name is **cird@28(FFFFF300,3)**, which is represented in the DG/UX long device name format. Refer to Appendix B for more information on nonstandard device names.

Building and booting a kernel

After connecting the device hardware, you must add the new device to your configuration. If you do not add the device, the operating system will not recognize it. Then you must boot the new kernel. The new system will then recognize the device and you can format it (if needed) and use it. See Chapter 10 for details on building and booting a kernel.

Soft formatting disk drives

This section explains the soft format procedure that lets disk media receive data. Only writable media can be soft formatted. A CD-ROM, for example, is not writable, so it cannot be formatted.

IMPORTANT: A nonvolatile RAM memory board (NVRAM board) must be soft formatted, just as a physical disk must be soft formatted, before you can create a virtual disk on it.

There are several aspects of soft formatting. You can perform all these in one **sysadm** sequence, or you can perform them individually. The easiest method is to perform them in one sequence. This sequence includes the option to perform the following operations:

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

Diskettes do not necessarily need to be software formatted. See the section “Using removable media devices” for more on using diskettes.

IMPORTANT: Soft formatting effectively erases any user data on a disk.

To soft format a physical disk or NVRAM memory board, follow these steps.

1. Make sure the drive or medium is write enabled (if this applies) and, for a diskette, that it is properly inserted 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(ncsc(0,7),0,2)
        2  sd(ncsc(0,7),3,0)
   ```

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

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

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

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

5. 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. Each other kind of disk, including a NVRAM memory board, must have a label so that the system can access it. In any case, installing a label does no harm, so take the default:

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

6. The operating system needs a virtual disk table to describe the virtual disks on a physical disk. This table establishes the virtual disk format. Tell **sysadm** to create one:

   ```
   Create Virtual Disk Table? [yes] ↵
   Establish Bad Block Mapping Facilities? [yes]
   ```

7. Bad block mapping lets DG/UX note bad blocks that it finds 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 necessarily 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] ↵
   Size of Bad Block Remap Area (blocks): [n]
   ```

8. 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 to 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] ↵
   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.

9. Relatively, the bootstrap consumes little space. For a diskette or NVRAM memory board, we suggest you answer **no**. For all other disks, answer **yes**, for example

   ```
   Install Bootstrap? [yes] ↵
   Disk label type [Generic SCSI]
   ```

10. The X window-based **sysadm** prints 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
        xxx.   Do you wish to continue? [yes]
    ```

11. 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:

    ```
        xxx.     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 xxx
    Physical disk xxx 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 3). Or to format another disk, repeat the steps in this section.

Using removable media devices

This section explains contains the requirements and uses for the following removable media device.

- Tape
- CD-ROM (Compact Disk–Read-Only Memory)
- Multiple-read/write magneto-optical disk
- Diskette

Using tapes

A tape requires no formatting. When you add a new tape drive to your hardware configuration, it must be recognized in the kernel. Refer to Chapter 10 for information on ensuring that the system file lists the newly added tape drive. The entry for a tape drive begins with **st**: for example, **st(ncsc(),4)**.

Using CD-ROM, magneto-optical, and diskettes

For CD-ROM, magneto-optical, and diskette drives, make sure you have installed the proper terminator on the last drive in a chain of SCSI disk drives. Also, excessive cable length could result in problems when trying to access the last device on the chain.

Each CD-ROM and Model 6880 diskette drive must have a unique SCSI ID. However, up to four magneto-optical or diskette drives (other than Model 6880) may be clustered on the same SCSI ID. If there are multiple drives on the same SCSI ID, then to access each drive you must include the LUN (logical unit number) in the DG/UX device name. For example, the following DG/UX device names represent three 5.25-inch diskette devices at SCSI ID 3:

```
sd(insc(0),3,0)
sd(insc(0),3,1)
sd(insc(0),3,2)
```

Using a CD-ROM disk drive

To add a file system for a CD-ROM, use the **sysadm** sequence `File systems -> Local Filesys -> Add.` The disk device must be registered before you can add the file system, and you must specify the device filename and a **cdrom** file type. To learn the drive filename, refer to “Learning device filenames while DG/UX is running” in this chapter. Chapter 3 gives instructions on registering the disk drive and adding a file system. For general information, see the **hfm** man page.

Before removing a compact disk from the drive, first unmount the file system. If you had originally registered the drive (as needed if the disk has a virtual disk on it), you also need to deregister the drive. Refer to the section “Removing media from drives” later in this chapter for details.

For DG/UX to recognize a drive, it must be specified in the kernel. For information on ensuring that the system file lists a newly added disk drive, see Chapter 10.

Using a magneto-optical disk drive

Decide how many file systems you want on the drive. If you want to create multiple file systems, you must create a virtual disk on the disk as explained earlier in this chapter.

If you want to create only one file system for the drive, you do not need to soft format the disk. However, if you do not soft format the disk, you will not get software-managed bad block mapping. To create just one file system, use the **sysadm** path `File system -> Local filesys -> Create`.

To learn the device filename, see “Learning device filenames while DG/UX is running” later in this chapter.

After creating the file system, you must add the file system with **sysadm**, as explained in Chapter 3. You must specify the device filename and **dg/ux** file type.

Before removing an optical disk from the device, first unmount the file system. If you had originally registered the drive (as you do with disks that contain named virtual disks), you also need to deregister the drive. Refer to a later section for instructions on removing a disk from a drive.

You may use a magneto-optical disk drive like a tape drive as well as a file system. You can write to the drive using the **tar** or **cpio** commands just as you can to tape drive.

You cannot use **sysadm** to create a backup on an optical disk. To make a backup on an optical disk, use the **dump2** command. If you do this, you are limited to one optical disk.

When you add a new magneto-optical device to your hardware configuration, you must ensure that the DG/UX kernel supports it. Chapter 10 explains how to ensure that the system kernel supports a disk drive.

Using a diskette drive

The 3.5-inch diskette drive supports 720-Kbyte and 1.44-Mbyte formats. The 5.25-inch diskette drive supports 360-Kbyte, 720-Kbyte, and 1.2–Mbyte formats. To create multiple file systems on a diskette, first use **sysadm** for formatting.

To prepare a diskette in MS-DOS format, you do not soft format it with **sysadm**; instead you create a file system with the **pc** option (**sysadm** sequence `File system -> Local filesys -> Create` and at the Mkfs options prompt, enter **pc**). To prepare a diskette in DG/UX format, you do not need to soft format the diskette; instead you can create a file system directly on the diskette. Since diskette capacity is relatively small, creating a virtual disk might consume too much disk space. Also, if you do not software format a diskette, you do not need to register it. For general information, see the **dfm** man page.

Before you can use a diskette, you must insert it in a drive and add (mount) a file system for it (use **sysadm** sequence `File system -> Local filesys -> Add`).

Before removing a diskette from the drive, first unmount the file system. If you registered the device (as needed if the diskette that contains a virtual disk), you also need to deregister the device. Refer to a later section for instructions on removing a medium from a drive.

When you add a new diskette drive to your hardware configuration, it must be recognized in the kernel. Chapter 10 explains how to ensure that the system file lists the newly added diskette device.

Using a diskette as a file system

Decide how many file systems you want on the diskette. If you want to create only one file system on the drive, you do not need to soft format the diskette. However, if you do not soft format the diskette, you will not get software-managed bad block mapping. To create a single file system, use the **sysadm** sequence `File system -> Local filesys -> Create`.

To learn the device filename, see “Learning device filenames after the system is booted” later in this chapter.

After creating the file system, you must add the file system with **sysadm**, as explained in Chapter 3. You must specify the device filename and a **dos** file type.

Using the diskette as a tape

Instead of using the diskette drive as a file system, you may use it like a tape drive. Like a tape drive, you can write to the diskette using the **tar**, **cpio**, **dump**, **dump2**, or **dd** commands. Refer to the drive as **/dev/rpdsk** or **/dev/pdsk**. Before you can use the drive this way, you must unmount it (if mounted) and deregister it (if registered), as explained in following sections.

Because diskettes do not have tape marks (as found on magnetic tape), you cannot use multiple diskettes with the **tar**, **dump**, **dump2**, or **dd** commands. The **cpio** command, however, allows you to use multiple diskettes when archiving files.

You cannot use **sysadm** to dump to a diskette drive. You may dump to a diskette from the shell using the **dump2** command, but you are limited to one diskette.

If you boot the system while a diskette with a **tar** or **cpio** format file is in the diskette device, you may see an error relating to the physical file table. Ignore this error. If the diskette device is registered as a file system when you try to write to it as a file (with the **tar** or **cpio** command), the error message, `conflict on open` appears. Deregister the device.

Removing media from drives

Before you remove an optical disk, compact disk, or diskette from a drive, you should first make the file system on the drive inaccessible to users. You can send a broadcast message using the **wall**(1M) command to all logged in users. Then

- Unmount the file system
- Deregister the device (if you registered it)

Removing a disk or diskette that is mounted and/or registered results in an error message such as:

```
From System:
The file system on device device-name sealed,
            Status nnnnnnn   (nnnnnn is a status number.)
            Run fsck to restore
```

or an error message such as:

```
File system is no longer fault tolerant
```

If either kind of error occurs, you must re-insert the diskette and run the **fsck** command to recover. For more on the **fsck** command, see *Managing the DG/UX™ System*

Unmounting a file system

Unmounting a file system detaches it from the file system hierarchy, making it inaccessible to users. You need to know the file system on the device medium that you want to remove from the drive.

1. Follow this path through **sysadm**:

   ```
   File System -> Local Filesys -> Unmount
   ```

   Sysadm prompts for a file system name:

   ```
   File System(s) to Unmount:
   ```

2. Enter a file system name, or if sysadm is not displaying a list of names (as with ASCII **sysadm**), enter ? to display a list of mounted file system. For example,

   ```
   File System(s) to Unmount: /accounts ↵
   OK to perform operation? [yes]
   ```

3. If you're sure you want to unmount the file system, confirm with Enter:

   ```
   OK to perform operation? [yes] ↵

   /accounts is unmounted.
   ```

   If you receive the message `Device Busy`, this means you cannot unmount the file system. Perhaps a user's working directory is in the file system you are trying to unmount, or a user's path variable includes that file system. Or a user might be running a program or reading data from a file located in the file system you are trying to unmount. You can find out the identity of the user with the **fuser** command, as follows.

   fuser -u *file-system*

   where *file-system* has the form **/dev/dsk/***virtual-disk–name*; it is not the mount point.

   You can ask the user to exit from the file system. With no users on the file system, you should be able to unmount it. You can also use **fuser** to kill such processes, freeing the file system; see **fuser**(1M).

Deregistering a physical disk

Deregistering a physical disk closes it to the operating system; this lets you remove the disk media (if this applies) or run other programs (such as diagnostics) on the disk. Deregistering a physical disk makes any virtual disks on it inaccessible. Shutting down DG/UX automatically deregisters all physical disks; starting a DG/UX system automatically registers all disks specified in the DG/UX kernel.

To deregister a physical disk, follow this path through **sysadm**:

```
Device -> Disk -> Physical -> Deregister
```

Sysadm lists one or more of the disks that are registered. If not, enter **?** for a list of such disks. You must select one of the disks listed. For example,

```
Physical Disk(s): [sd(ncsc(0,7),0] ?↵

    11  sd(ncsc(0,7),0,0)
    12  sd(ncsc(0,7),1,0)
    13  sd(ncsc(0,7),3,0)
    14  sd(ncsc(1,7),2,0)

Enter ... 14↵        (Select disk sd(ncsc(1,7),2,0).)
```

After you select the device name of a disk you want deregistered, **sysadm** deregisters the disk. You can now remove the optical disk, CD-ROM, or diskette from the drive (if this applies).

Learning device filenames while DG/UX is running

At startup, the system generates a list of device filenames located in **/etc/devlinktab**. Use these names when referring to mass-storage devices for **sysadm** and command line operations. The **devlinktab** file lists the long name/short name pairs for each device in your configuration. For example, a SCSI tape drive has this DG/UX device name:

```
st(insc(0),4,0)
```

This is the short device name. It corresponds directly to an entry in the **devlinktab** file, as follows:

```
# directory      short      long
#
/dev/rmt         0          st(insc@7(FFF8A0007),4,0)
```

You use a device's short name when you need to write to it (like a tape) or add it as a file system. For more information on device naming, see Appendix B.

Where to go next

After learning about connecting and using disks and tapes, you will probably want to add them to the DG/UX kernel, so that you can software format them, create file systems on them (if applicable), and use them. The next chapter explains adding devices to the kernel.

End of Chapter

10 Building a DG/UX kernel

A DG/UX kernel is an executable program that provides operating system services to all other programs running on the system. The kernel runs directly on the hardware, managing access to I/O devices as well as user requests and application programs.

This chapter explains when and how to build a kernel. Major sections proceed as follows.

- Do you need to build a kernel?
- Choosing the method for building a kernel
- Building an auto-configured kernel
- Building a custom kernel
- Booting a kernel
- Where to go next

If you have OS clients that are not yet operational, go to Chapter 7 for instructions on building an OS client's first kernel.

Do you need to build a kernel?

You need to build and boot a kernel for computer systems that

- Have had new any I/O devices added
- Have had streams modules or socket protocols added
- Have had real or pseudo device drivers added
- Need a tunable variable changed

If you already have a tailored kernel, but believe you need to build and boot another kernel (for example, you have just added another SCSI disk drive), check the values set in the current kernel first. Use the **sysdef** command as shown after the next section. You may not need to build another kernel.

Finding out which kernel is in use

You may have multiple kernels resulting from several kernel building sessions. The default kernel is the one linked to the DG/UX system, file **/dgux**. You can find out which kernel is linked to **/dgux** by using the following shell command:

```
# ls -i /dgux* ↵
4051 /dgux   4049 /dgux.installer   4058 /dgux.gyramax
4051 /dgux.sport
```

The output shows that the inode numbers match for **/dgux** and **/dgux.sport**. (An inode is a data structure containing information about a file such as file type, size, date of creation, owner ID, and group ID.) Therefore, **dgux.sport** is the name of the system file linked to file **/dgux**, but this not necessarily the system running.

Checking the configuration variables set in the kernel

You can check the current values for configuration variables in the kernel. Checking these values may be helpful in determining whether or not you need to build a new kernel. For example, if you add a SCSI tape drive to your configuration, you must build a new kernel only if the kernel does not already include an entry for the drive.

The **sysdef** command displays values for the kernel you specify, or if you omit a kernel name, for the kernel that is linked to **/dgux**. For example,

```
# sysdef ↲
# Configured devices
#
vme()
kbd()
grfx()
syac(vme(0),0))
duart(0)
duart(1)
dgen(0)
wdt
sd(ncsc(0),0)
sd(ncsc(0),1)
sd(ncsc(0),3)
st(ncsc(0),4)
st(ncsc(0),5)
st(ncsc(0),6)

# Configuration variables
#
NODE "sport"
...
```

The list includes three SCSI tape drives on SCSI IDs 4, 5, and 6 on the first ncsc controller. If the new tape drive is connected to any of these SCSI IDs on this controller, building a new kernel is not required.

You can also peruse the system files on which kernels are based at this location: **/usr/src/uts/aviion/Build**. Scan your system file(s) using the **more**, **cat**, **grep**, or **view** commands.

Choosing the method for building a kernel

The **sysadm** program lets you build a kernel in one of two ways: with the Auto Configure or Build choice from the **sysadm** kernel menu.

IMPORTANT: DG/UX provides a sizing program called **probedev**, which **sysadm** uses to create a list of devices from which it creates system files and kernels. However, **probedev** can detect only those devices that are supported by the currently running DG/UX kernel. Therefore, when you create a kernel to support your current hardware, you must ensure that the underlying operating system supports all devices you could possibly have. The DG/UX installer kernel, **/dgux.installer**, and the stand-alone **sysadm**, **/usr/stand/sysadm,** do support all possible devices. Therefore, ensure that your own custom kernel that supports all your devices or the latest revision of **/dgux.installer** is running when you create a kernel.

Auto Configure and Build are similar in that they both run the **probedev** sizing program to build a list of devices. However, Build lets you edit the system file before building the kernel; Auto Configure does not.

The **sysadm** Kernel Menu offers the Auto Configure and Build methods for building a kernel. It also provides the Reboot option. Details follow.

Auto Configure Builds a kernel that includes default variable values and recognizes all attached standard I/O devices. A kernel built using this option is an "auto-configured" kernel. The auto configure option requires no user interaction; it builds the kernel and links it to **/dgux** automatically. You may choose this kernel-building option for a computer system that does *not*

- share a disk-array storage system with another host;
- connect to a nonstandard I/O device; or
- have packages (on attached disks) that require any special variable tuning.

Do not use **Auto Configure** to build a kernel for a host that shares a disk-array storage system with another host, or is customized with nonstandard devices or tunable parameters. In this case, using **Auto Configure** destroys the custom changes.

Build Builds a kernel that is the same as the one built with Auto Configure except that you can edit the system file, perhaps to customize the tunable variable values or add entries for nonstandard devices. A kernel built using this option is a *custom* kernel.

Reboot Shuts down the system completely (except for the hardware) and restarts the operating system, activating a new kernel.

Build an auto-configured or custom kernel according to your needs.

Building an auto-configured kernel

When using this option, you do not need to edit the system file. Do not use this option if any of the following is true:

- You are building a first-time kernel to support a disk-array storage system that connects to two hosts (any dual-host configuration).
- You are building a first-time kernel for an OS client at the OS server.
- You have nonstandard devices attached to your system.
- You need to tune any variables for a software package you may have installed on your system.
- You are building a kernel to add a device whose device driver is not part of the currently running kernel.
- You do not want the next kernel automatically linked to **/dgux** (**sysadm** links the kernel as part of the auto configure process).

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

```
System -> Kernel -> Auto Configure
System configuration file name: [xxx]
```

2. The system configuration filename distinguishes this system file (and derived kernel) from all other system files and kernels. The filename you type here serves as a base for all this system's filenames. The configuration filename will be **system**.*xxx*; the executable kernel filename will be **dgux**.*xxx*. If you want the new system file and kernel to replace existing files (which can save a lot of disk space), enter a name that will match the names of existing files. If you want to create new files, enter a name that will produce a unique name. For example,

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

If the file already exists, **sysadm** asks if you want to overwrite it:

```
Overwrite existing [system.xxx] [yes]:
```

If you know you want the new system to delete the old one, press Enter. Otherwise, enter **n** and respecify the name.

Sysadm prompts

```
[system.xxx]  Correct? [yes]
```

3. If the new filename is the one you want, press Enter; otherwise, enter **n** and respecify the name. For example,

```
[system.xxx] Correct? [yes] ↵
Operating system client? [no]
```

4. If your system is an OS client that will connect to an OS server, enter yes. Otherwise, press Enter for no. (As mentioned above, you should not use the Auto Configure option to build a kernel on an OS server for an OS client.) For example,

```
Operating system client? [no] ↵
Okay to perform operation? [yes]
```

5. To build the kernel, press Enter. To change an answer, enter **n** and respecify. After you confirm, **sysadm** displays

```
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 sensed by the current kernel. This is explained earlier under the IMPORTANT note, in section “Choosing the method for building the 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.***xxx* (*xxx* is the name you specified in step 2). The system configuration file resides in **/usr/src/uts/aviion/Build/system.***xxx*.

   The new kernel is not effective until you boot it. See the section “Booting a kernel” later in this chapter for booting instructions.

Building a custom kernel

Using this option, you can build a custom kernel for your computer system or build a kernel for an OS client from an OS server.

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

```
System -> Kernel -> Build
System configuration file name: [xxx]
```

2. The system configuration filename distinguishes this system file (and derived kernel) from all other system files and kernels. The filename you type here serves as a base for all this system’s files. The configuration filename will be **system.***xxx*; the executable

kernel filename will be **dgux**.*xxx*. If you want the new system file and kernel to replace existing files (which can save a lot of disk space), take the default or enter a different name that will match the names of existing files. If you want to create new files, enter a name that will produce a unique kernel filename.

If you are creating a kernel for your machine, you will probably want to accept the default. If you are creating a kernel for an OS client, or if for any other reason you want the new system to have a unique name, then you should provide a new name. For example,

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

Sysadm prompts

```
[system.xxx]  Correct? [yes]
```

3. If the new filename is the one you want, press Enter; otherwise, enter **n** and respecify the name. For example,

```
[system.xxx] Correct? [yes] ↵
Build for this host or for OS client(s) of this
   host: [this host]
```

4. If you are building for this host (which may be an OS server, OS client, or neither), take the default, `this host`, and skip to step 6. If you are building an OS client kernel from an OS server, enter **OS client** and continue here. For example, if you are building for an OS client on an OS server:

```
Build for this host or for OS client(s) of this
   host: [this host] OS client ↵
```

5. **Sysadm** prompts

```
Boot Device: [inen(0)]
```

Specify the network controller boot device that the client will use, or take the default if the boot device is **inen(0)**. Instead of an integrated network controller (**inen**), you can specify a Data General second generation network controller, named **dgen**(*n*), where *n* is a number 0 through 5. For example,

```
Boot Device: [inen(0)]  dgen(0) ↵
```

6. **Sysadm** prompts

```
Editor: [/usr/bin/vi] ↵
```

7. Specify an editor for editing the system file. The **vi** editor is the default (see Appendix A for a summary of **vi** commands). Or you can can specify another editor. For **vi**:

```
Editor: [/usr/bin/vi] ↵
```

Editing the system file for this host

The system file is several screens long. You will see that configuration variables that do not apply to your system are or commented out. A line that is commented out contains a # in the first column. Please read the file, and the following sections, before you edit the file.

A common source of many kernel build failures is the absence of a comment symbol (#) used to flag notes to be ignored. Be sure you comment out all text that is to be ignored. Check your spelling and verify the DG/UX device names.

Figure 10–1 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 10–1 Hardware Devices Excerpt from Configuration System File

Automatically configured hardware devices

Each attached standard device (specified using the DG/UX device format with an associated brief description) is automatically configured each time you build a custom kernel. The # symbol signifies a comment, which is ignored. For more information on the DG/UX device name format, refer to Appendix B.

To anticipate future additions of more disk drives to your hardware configuration, you can edit the system file and express the device name's SCSI ID field (final field) using the asterisk pattern-matching metacharacter. For example,

```
sd(ncsc(0),*)    ## All SCSI disks on Second-Generation SCSI adapter 0
```

The asterisk matches all disk drive SCSI IDs, eliminating the need to list each full disk drive name. Using the asterisk also eliminates the need to rebuild a new kernel each time you add a SCSI disk drive to the first controller.

IMPORTANT: Use the asterisk this way only when you want the operating system to support all SCSI disk units it finds on the SCSI disk controller. Do not use the asterisk in any configuration where two host systems have their own set of disks in a disk-array storage system (that is, in any dual-host configuration). If you use the asterisk in such a configuration, the first host booted will take control of all the disks in the storage system and the second host will not be able to use any. For more on this issue, see the 014-series storage-system manual supplied with the disk-array storage system.

Nonstandard hardware devices

A nonstandard device uses a driver that is not supplied by Data General (see “Standard and nonstandard devices” in Chapter 9). You are responsible for resetting the device’s jumper or DIP switch position to establish the desired device setting and for determining its device name in long form. Refer to the device’s hardware installation documentation for information on resetting a device’s jumper or DIP switch position, and Appendix B in this manual for information on DG/UX device naming conventions.

An example of a nonstandard device name in the system file follows.

```
# Hand-entered Nonstandard Devices

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

General configuration variables

NODE refers to the hostname of your computer. Your computer’s hostname, for example **sport**, is automatically supplied. It is used by the **uname** command to report the name and other attributes of the current system. Also, the **uucp** command uses the node name for performing file transfers on UNIX systems. It is also presented as part of the login banner message when you log in to your system. If you set up the TCP/IP package, the node name also corresponds to the hostname that you supply during TCP/IP setup. The node name is restricted to a maximum of 31 characters.

The master files located in the **/usr/etc/master.d** directory contain a complete list of general configuration tunable parameters. Do not change the settings in these files. You can override a given default by setting the parameter to the desired value in the system file. Read about the tunable parameters in *Managing the DG / UX*™ *System*. You are accepting the defaults by not changing any values assigned to variables in the system file.

A sample tuning variable is

```
      MAXUP 60
```

MAXUP specifies the maximum number of processes a non-superuser can have at any one time. The default MAXUP value is 50 processes. The preceding example overrides the default value.

Package variable tuning

The system file contains a series of concatenated package prototype system file fragments. If you have just loaded a new package on your system, check its release notice for information on possible variable tuning. If you need to tune a variable, either enter or modify it in the system file. Where you place the variable and value pair in the system file is unimportant. For example,

```
# These variables set values for message parameters and inter-
# process communication shared memory for fredware package.

MSGMAX      8192
MSGMNB      16384
SHMMAX      524288

# End of fredware variables.
```

These variables affect space for messages and shared memory for interprocess communication. MSGMAX specifies the number of bytes a message can contain; MSGMNB specifies the maximum number of bytes a message queue can contain; and SHMMAX specifies the maximum number of bytes in a segment. The package's release notice specifies certain variable value assignments. The example above shows kernel variables tuned for specific needs.

OS client configuration variables

If you are building a kernel for an OS client at the OS client, you must insert these OS client configuration variables and assigned values. You can do so anywhere in the system file. For an OS server building a kernel for an OS client, these variables are set automatically.

```
      # OS Client Configuration Variables

      NETBOOTDEV  ″inen( )″
      ROOTFSTYPE  NETWORK_ROOT
      SWAPDEVTYPE NETWORK_SWAP
```

NETBOOTDEV, ROOTFSTYPE, and SWAPDEVTYPE provide resources to the OS client over the network. Make sure there are no leading comment symbols (#) in the first columns of the lines containing these variables. Also, comment out the descriptive line labeled “OS Client Configuration Variables“ as shown above.

Tunable configuration variables for workstations

Kernels built for systems with a graphics device and those built for OS clients may need special tuning to improve system performance. The MAXSLICE variable is set automatically on these systems.

```
        MAXSLICE 50
```

MAXSLICE specifies the maximum time in milliseconds a user process can run before being suspended. After a process executes for its allocated time slice, that process is suspended. The default MAXSLICE value is 500 milliseconds (1/2 second). Reducing the MAXSLICE value can improve interactive response time for users running the X Window system. The preceding example overrides the default value.

8. .After you finish editing the system file, save the file and exit from the editor (**ZZ** command for **vi**).

Building the kernel for this host

You must let **sysadm** link the new kernel (here, named **dgux.sport**, for example) to **/dgux** before the new kernel can be booted with the default boot path. If you do not link the new kernel to **/dgux**, the existing kernel (if one exists) remains linked to **/dgux**, and you continue to use the “old” kernel.

```
Link the new kernel to /dgux? [yes] ↵
```

9. To link the new kernel to **/dgux**, accept the `yes` default. If you want to keep the current kernel linked to **/dgux**, enter **n**. For example

```
Link the new kernel to /dgux? [yes] ↵
```

Now, if you specified the name of an existing system earlier, sysadm asks if you want to save the existing system files. If you specified a new name, **sysadm** skips the following prompt.

```
Save the old kernel? [yes]
```

10. If you want to save the existing system files in addition to the new ones (at some cost in disk space), answer yes; to delete the existing system files and replace them with the new ones, enter **n**. For example,

```
Save the old kernel? [yes] n ↵
```

Sysadm prompts

```
Continue with the build?  [yes]
```

11. To continue, confirm:

```
Continue with the build?  [yes] ↵
Configuring system...
Building kernel...
Successfully built dgux.xxx
```

If you told it to link **/dgux**, **sysadm** also displays

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

If the build fails, remember that many build failures result from the absence of a comment symbol (#) in comment lines in the configuration file.

You have successfully built a custom kernel. The kernel file resides in **/dgux.***xxx* (*xxx* is the name you specified in step 2). The system configuration file resides in **/usr/src/uts/aviion/Build/system.***xxx*.

You must boot the new kernel for it to take effect.

Booting a kernel

When you boot a new kernel, all processes running on the system are killed; therefore you should not do this if users are logged on and/or applications are running.

IMPORTANT: Regardless of whether you build the kernel at the OS server or OS client, each OS client must boot its own kernel.

1. From the Kernel Build Menu, select the Reboot option. **Sysadm** displays the boot path, for example:

```
Boot path: [sd(ncsc(0),0,0)root:/dgux -3]
```

2. Verify the boot path, entering the name of the new system if it differs from the one displayed. Details on specifying a boot path appear in Chapter 11. Press Enter. **Sysadm** asks for confirmation:

```
All currently running processes will be killed.
Are you sure you want to reboot the system? [yes]
```

3. Confirm your desire to reboot by pressing Enter again.

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 5.4R3.00
Loading image ..........................

sport
Console login:
```

For details on booting and logging into the system, see Chapter 11.

Where to go next

After building and booting a kernel, you may want to simply run the system or perhaps look at some of the conceptual material in *Managing the DG/UX™ System*. The next chapter in the manual you are reading gives detailed booting information.

End of Chapter

11 Booting and logging in the DG/UX system

The primary reasons for booting a DG/UX system are to gain access to a new kernel that has just been built and to restart the system following a normal or abnormal shutdown. For information on failure detection and recovery, see *Managing the DG/UX™ System*.

Major sections in this chapter proceed as follows.

- Warning users before rebooting
- Booting from sysadm
- Booting from the SCM
- Booting from disk, tape, or network
- Boot messages
- Setting the default boot path
- Setting the auto reboot boot path
- DG/UX run levels
- Logging in the DG/UX system
- Shutting down the DG/UX system
- Where to go next

Regardless of the method used to boot the DG/UX system, the basic boot command is the same. It identifies a boot file located on a local device (disk or tape) or the local area network (LAN) device. See "Booting from disk, tape, or network" later for boot path syntax.

Warning users before rebooting

Rebooting your computer kills all running processes. If the OS server reboots, attached OS clients may continue to operate normally or they may reboot automatically. Rebooting an OS client, does not affect the OS server.

If multiple users are logged in to the system before you reboot, broadcast a message warning that you are about to reboot the system. See the **wall**(1M) manual page for information on sending a broadcast message to all users logged in the system. Make sure all users are logged out before proceeding. An example of such a warning follows.

```
# /etc/wall ↵
Five minutes until we reboot the system.  Please log off. ↵
<Ctrl-D>
```

When a system is rebooting, messages appear that describe the system initialization stages followed by a login prompt.

Booting from sysadm

Follow this path through **sysadm**:

```
System -> Kernel -> Reboot
```

In the following example, the kernel **/dgux** on the **root** virtual disk located on **sd(insc(0),0,0)** is booted automatically to a run level of 3.

```
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 default boot path is preset at the factory. You can reset it with the **dg_sysctl** command or the SCM Configuration Menu. You can override the default to establish the correct boot path for your system. Refer to "Specifying a boot path" for the exact syntax. For example, you could establish a boot path for an OS client of **inen**() **–3**, specifying a boot to run level 3 over the network. A warning appears that all currently running processes will be killed when you reboot. Press Enter to reboot the system.

Booting from the SCM

A system that has been shut down operates at the SCM level. An OS client will boot its kernel from the SCM. From the `SCM>` prompt, issue a boot command using this syntax:

`SCM>` **b** *boot-path*

where **b** stands for boot and *boot-path* identifies the boot file located on a boot device. For example

`SCM>` **b sd(insc(0),0,0)root:/dgux –3**

This command boots **/dgux** contained in the virtual disk **root**. The boot device is a SCSI disk attached to the first integrated SCSI adapter whose SCSI ID is 0. The system boots to a run level of 3.

Rather than typing this lengthy command each time you want to reboot the system, you can establish a shortcut using instructions given later.

Boot messages

When you boot your system, messages describe the progress of system initialization. The wording of the messages depends on the run level to which you are booting and whether you are booting an OS server, stand-alone system, or OS client. The example in Figure 11–1 shows typical server boot messages.

```
System -> Kernel -> Reboot

Boot path: [sd(ncsc(0,0,0)root:/dgux -3] <Enter>
All currently running processes will be killed.
Are you sure you want to reboot the system? [yes] <Enter>
Booting sd(ncsc(0,0,0)
DG/UX System Release 5.4R3.00 Bootstrap
Loading image
...............................................................
DG/UX System Release 5.4R3.00, Version generic
Using 32 Megabytes of physical memory
Found 2 processors
Configuring devices ...........................................
Registering disks ....
Using vdm(swap,2C7CC95E,373A2264,0) as swap disk.
No check necessary for vdm(root,2C99F585,373A2264,0).
Mounting vdm(root,2C99F585,373A2264,0) as root file system.
    Checking local file systems ...............
    Mounting local file systems .....
    Current date and time is Tue Nov  2 09:29:20 EST 1993 ...
    Checking system files ........
    Continuing system initialization ....
    Enabling automatically pushed STREAMS modules ........
    Linking short names for /dev device nodes ............

    Loading terminal controllers ....
    Starting disk daemons ....
    Mounting local file systems .....
    Checking for packages that have not been set up ....
    Starting miscellaneous daemons ...
    Starting streams error logging daemon ....
    Starting Logical Link Control Services ....
    Attaching TCP/IP network interfaces ...........
    Starting system logging daemon ....
    Starting DG/UX services daemon .............
    Starting NIS services as NIS client ......
    Starting NFS lock services .....
    Allowing remote systems to reclaim NFS locks ...............
    Starting batch services ....
    Starting line printer scheduler ....
    Saving ex(1) and vi(1) temporary files ....
    Starting NFS services .....
    Starting TCP/IP daemons .......
    Mounting NFS file systems .....
    Starting user count monitor ....
    Starting DG/UX administrative services .....
    Starting NetWorker daemons ....


    NOTE:  The run level change is complete.  See /etc/log/init.log
        for a verbose description of the system initialization process.
```

Figure 11–1 Typical Messages for OS Server Booting to Run Level 3

Figure 11–2 shows a typical boot command and booting messages at an OS client. In this example, **bob** is the name for the OS client and **steve** is the name for the OS server.

```
SCM> b inen() –3 <Enter>

Booting inen
Local Ethernet address is 08:00:1C:1F:03:77
Local Internet address is 128.222.14.32 or 80DE0E20 hex
Trying server at 128.222.14.31 or 80DE0E1F for TFTP transfer
00166464

DG/UX Bootstrap 5.4R3.00

Boot:  inen(0)
Local Ethernet address is 08:00:1C:1F:03:77
Local Internet address is 128.222.14.31
Broadcasting request for a boot server ...
Host name:  bob
Using steve:/srv/release/PRIMARY/root/bob as root
Loading /dgux ....................................
DG/UX System 5.4R3.00, Version generic
Using 16 Megabytes of physical memory
Found 1 processor(s)
Processor 1 is running
Configuring devices  ........................................


      Checking local file systems ...
      Current date and time is Thur Oct 22 14:30:24 EDT 1993
      Checking system files .....
      Enabling automatically pushed STREAMS modules ...
      Loading terminal controllers ...
      Starting disk daemons ...
      Mounting local file systems ...
      Checking for packages that have not been set up ...
      Starting miscellaneous daemons ...
      Starting Logical Link Control Services ...
      Attaching TCP/IP network interfaces ...
      Starting system logging daemon ...
      Starting miscellaneous daemons ...
      Starting NIS services as NIS client ...
      Starting NFS lock services ...

      NOTE:  Pausing for 15 seconds to allow remote systems to
             reclaim NFS locks.

      Starting batch services ....
      Starting line printer scheduler ....
      Starting ex(1) and vi(1) temporary files ....
      Starting NFS services .....
      Starting TCP/IP daemons .......
      Mounting NFS file systems ......

      NOTE:  See /etc/log/init.log for verbose description of
         the system initialization process.
```

Figure 11–2 Typical Messages for OS Client Booting to Run Level 3

Booting from disk, tape, or network

You can boot from a disk, tape drive, or local area network (LAN).

Disk or tape drive

IMPORTANT: AViiON systems can boot only from media formatted with a fixed-length record size of 512 bytes. Therefore, AViiON systems will not boot from QIC–320 or QIC–525 media written to in either 1024 bytes (1Kbyte) block size or in variable block size mode.

The syntax for booting a file located on a disk or tape drive follows:

DG / UX-device-name virtual-disk:*pathname* [*–run-level*]

IMPORTANT: Do not use spaces except between *pathname* and *–run-level.*

where:

DG / UX-device-name identifies the boot device. Refer to Appendix B for information on device naming. You can specify devices using the short or long form.

virtual-disk contains the executable kernel image.

pathname is the pathname of the executable kernel image on the virtual disk.

-run-level specifies the run level to which your system boots automatically. You should set your run level in the **/etc/inittab** file and use the boot path to override the pre-established default. Refer to the section "DG/UX run levels" for information on the run levels, and see *Managing the DG / UX*™ *System* for information on editing the **inittab** file.

Boot examples

b ↵

This boot command automatically boots the file identified in the autoboot path that you set through the SCM Configuration Menus.

sd(insc(),0)root:/dgux –3 ↵

This boot path specifies the **/dgux** kernel contained on the virtual disk **root**. The boot device is a SCSI disk attached to the first integrated SCSI adapter whose SCSI ID is 0. The system boots to a run level of 3, which enables multiuser mode.

st(insc(0),4,0) ↵

This boot path specifies the first virtual file (0) on a 150-megabyte QIC tape whose SCSI ID is 4.

Local area network

An OS client boots its kernel via a network. The command syntax is

network-controller-device OS-server Internet-address:pathname [*–run-level*]

IMPORTANT: Do not use spaces within any of the two fields.

where:

network-controller-device
is the name of the device used to connect the OS client to the local area network. Valid controller names have the form *controller-type* (*controller-number*) where *controller-type* can be **inen** (integrated Ethernet controller), or **dgen** (Data General second generation integrated Ethernet controller). The *controller-number* for controller type **inen** and **dgen** can be only **0**. Valid examples are **inen(0)** or **dgen(0)**.

OS-server-Internet-address
is the Internet address of the OS server. Consult your network system administrator for Internet addresses. An example of an Internet address is **128.223.2.1**.

pathname
identifies the kernel image to boot.

[*–run-level*]
specifies the run level to which your system boots automatically. You should set your run level in the **/etc/inittab** file and use the boot path to override the pre-established default. For information on editing the **inittab** file, see *Managing the DG / UX*™ *System*; for information on run levels, see section “DG/UX run levels,” later.

Local area network example

inen() 128.223.2.1:/srv/release/PRIMARY/root/bob/dgux ↵

The OS client boots the file located in **/srv/release/PRIMARY/root/bob/dgux** over the network controller **inen**( ) by way of the OS server’s Internet address. In this example, **bob** is the OS client hostname.

Setting the default boot path

You can establish a shortcut for booting your DG/UX system by setting a default boot path for the kernel. The shortcut allows you to type only the **b** command in lieu of a longhand boot command from the SCM prompt.

1. Type the **f** (format) command at the SCM prompt. The SCM Configuration Menu appears:

```
SCM> f  ↵

View or Change System Configuration

1. Change boot parameters
2. Change console parameters
3. Change mouse parameters
4. Change printer parameters
5. View memory configuration
6. Change testing parameters
7. Return to previous screen

Enter choice(s) ->
```

2. Type the option **1** followed by Enter to change boot parameters. The following menu appears.

```
Change Boot Parameters

1 Change system boot path
2 Change diagnostics boot path
3 Change data transfer mode [BLOCK]
4 Return to previous screen

Enter choice(s) ->
```

3. Type the option **1** again, followed by Enter. The system displays the current boot path. Enter **y** followed by Enter to change the path.

```
System boot path = [sd(insc(0),0,0)root:/dgux.installer]

Do you want to modify the boot path? [N] y ↵
```

4. Enter the new boot path at the prompt. The system them prompts for confirmation.

   Two boot disk examples follow. The first one is for a system having a SCSI system (boot) disk; it boots to a run level of 3 automatically.

   ```
   Enter new system boot path -> sd(insc(0),0,0)root:/dgux -3 ↵
   System boot path = [sd(insc(0),0,0)root:/dgux -3]
   Do you want to modify the system boot path? [no] ↵
   ```

   The next example is for a system having an ESDI system disk; it boots to a run level of 3 automatically.

   ```
   Enter new system boot path -> cied( )root:/dgux -3 ↵
   System boot path = [cied( )root:/dgux]
   Do you want to modify the system boot path? [no] ↵
   ```

   The system then offers to boot via the path you specified:

   ```
   Do you want to boot [N]
   ```

5. If you want to reboot immediately, answer **yes** to the next question. Otherwise, accept the **N** default, and return to the SCM prompt.

6. The Change Boot Parameters menu appears. Choose option 4.

   The View or Change System Configuration menu is appears. Choose option 7 to go to the SCM prompt.

   From now on you can type the letter **b** followed by Enter at the SCM prompt to boot the kernel.

Setting the auto reboot boot path

You can use the **dg_sysctl** shell command to define your system's booting behavior following a panic situation. You can establish whether or not the DG/UX system will automatically reboot and identify the mass-storage device to receive a system dump.

IMPORTANT: See *Managing the DG/UX™ System* for more details on actions to perform after a panic.

The syntax follows:

dg_sysctl [**–t**][**–r** *reboot-state*][**–b** *"boot-path"*][**–d** *autodump-state*] [**–f** *"dump-device"*]

where:

–t Makes a temporary change. The change does not persist following the system reboot. The default specifies a permanent change.

–r Sets the system's reboot behavior. If *reboot-state* is **auto**, the system automatically reboots after a panic. If *reboot-state* is **halt**, the system does not automatically reboot after a panic. The default is **halt**.

–b Sets the system's boot path. The boot path must conform to the SCM boot syntax. The *boot-path* is the path to use when the system is rebooted. Be sure to surround the boot path with double quotation marks (″ ″). Refer to "Setting the default boot path" for complete syntax.

–d Sets the system's *autodump* behavior. If *autodump-state* is **auto**, the system attempts to dump to *dump-device* after a panic. A tape must be present in the drive in the event of a dump. If *autodump-state* is **skip**, the system does not attempt to dump to *dump-device* after a panic. If *autodump-state* is **ask**, the system asks if you wish to take a system dump after a panic. The default is **ask**.

–f Sets the system's *dump-device* to be used during a panic. The default value for the DUMP variable is set in the **/usr/etc/master.d/dgux** file and can be reset in the system configuration file. For an AViiON 4000 OS server, for example, the default boot tape device is **st(insc(0),4,0)**. For an OS client, a dump is submitted over the network **inen()** to the OS server's dump device.

dump-device
is the name of the DG/UX device to which a panic dump is written.

Auto reboot path examples

```
# dg_sysctl ↲
```

With no arguments, this command reports the current values for the arguments, which include whether or not the boot path is permanently changed; whether the system will automatically autoboot following a panic or halt and query you about taking a dump; the boot path; and the dump device.

```
# dg_sysctl –r auto –b ”sd(insc(0),0,0)root:/dgux –3” –d auto –f ”st(insc(0),4)” ↲
```

This command enables auto-rebooting after a panic. It reboots the kernel located at the specified boot path to a run level of 3. The command also enables auto-dumping after a panic. The dump goes to **st(insc(0),4)**, which is a SCSI tape device with a SCSI ID of 4 that is attached to the first (0) integrated SCSI controller.

Make sure you have an appropriate tape inserted in the drive of the dump device. If the system panics, a dump is written to the specified device automatically.

DG/UX run levels

Table 11–1 lists the DG/UX run levels.

Table 11–1 DG/UX Run Levels

Run Level	Description
S or s	Single user mode. The system default file systems (**/swap**, **/**, and **/usr**) are mounted. No processes are running except those of the system administrator, who is logged in as **root**.
i	Installation mode. All local file systems are mounted and essential processes are running. The **installman** command is invoked to perform installation tasks. Refer to the **installman**(1M) manual page for more information.
1	Administrative mode. This mode is used to install and remove software, and to perform administrative tasks, such as checking file systems and doing backups. System processes are running and all file systems are mounted. You can login only at the console .
2	Multiuser mode. This is the mode with the most service for those who are not operating in a network environment and who are not running the DG/UX X Window System software, Release 4. All local file systems are mounted.
3	Multiuser mode. This is the mode required to run DG/UX X Window System software, Release 4. It is also the mode with remote file system sharing (NFS) and network services.
4	User–defined level. Used primarily for applications.
5	Stops the system and goes to the SCM. This state is equivalent to bringing the system to state **S** and issuing the **halt** command. Refer to the **halt**(1M) manual page for more information.
6	Stops the system and reboots the default boot path. This state is equivalent to bringing the system to state **S** and issuing the **reboot** command. Refer to the **reboot**(1M) manual page for more information.

OS clients and OS servers operate at run level 3, since network services are required to support OS clients.

Normally, you establish a run level from an autoboot path you entered through the SCM Menus or a **sysadm** menu selection. You can, however, change run levels from the shell using the **init**(1M) command:

init *run-level* ⤶

See the **init**(1M) manual page for more information on **init** command arguments.

Logging in the DG/UX system

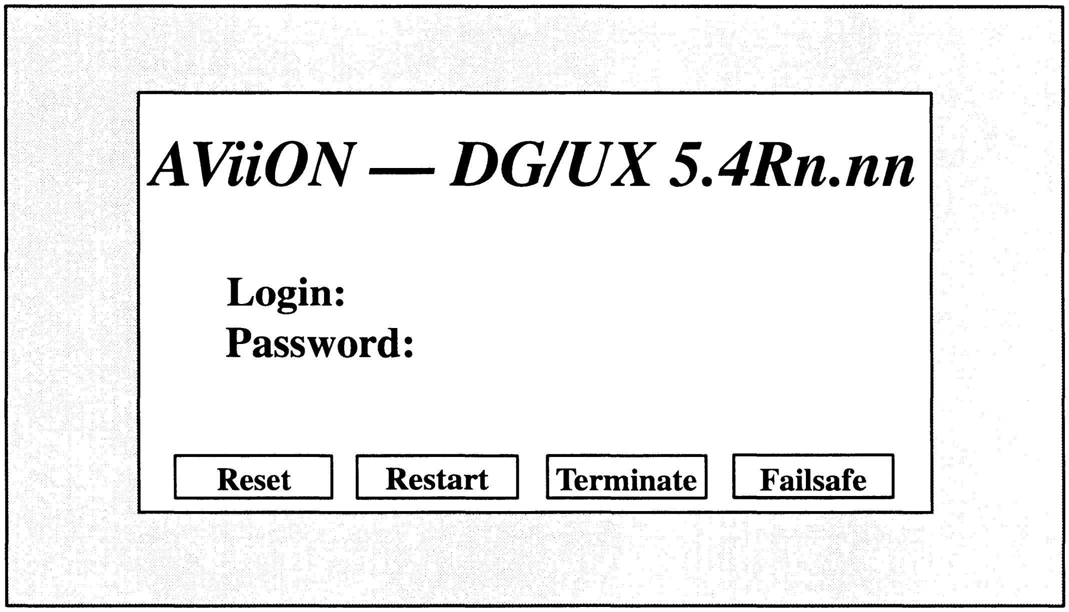
The login prompt appears after the DG/UX system has finished booting. The user login that you enter will depend on what you are doing. OS clients booting their first-time kernels need to set up software packages and mount file systems before their computers are operational. Therefore, OS clients should log in as **sysadm** to gain access to **sysadm**, which offers a menu-driven set of system administration procedures.

Those who do not need to perform continued setup activities, can log in with their user logins to begin real work. Make sure that you have been assigned a user login and password (see Chapter 4 for details).

The type of login prompt you see will depend on whether you are using a graphics monitor or an alphanumeric display terminal. Choose the appropriate section for continued procedures.

Logging in with a graphics monitor

If your system console is a graphics monitor and you have installed the X Window System package, you will see the following login box displayed on the screen (see Figure 11–3).



AViiON — DG/UX 5.4Rn.nn

Login:
Password:

Figure 11–3 Login Screen for a Graphics Monitor

The four buttons along the bottom of the screen are aids to controlling the X server, which is the program that controls the display of information.

Reset Resets the X server. The X server is not terminated.

Restart Terminates the X server, then restarts it.

Terminate Terminates the X server, giving control to a single VT100 terminal emulation screen.

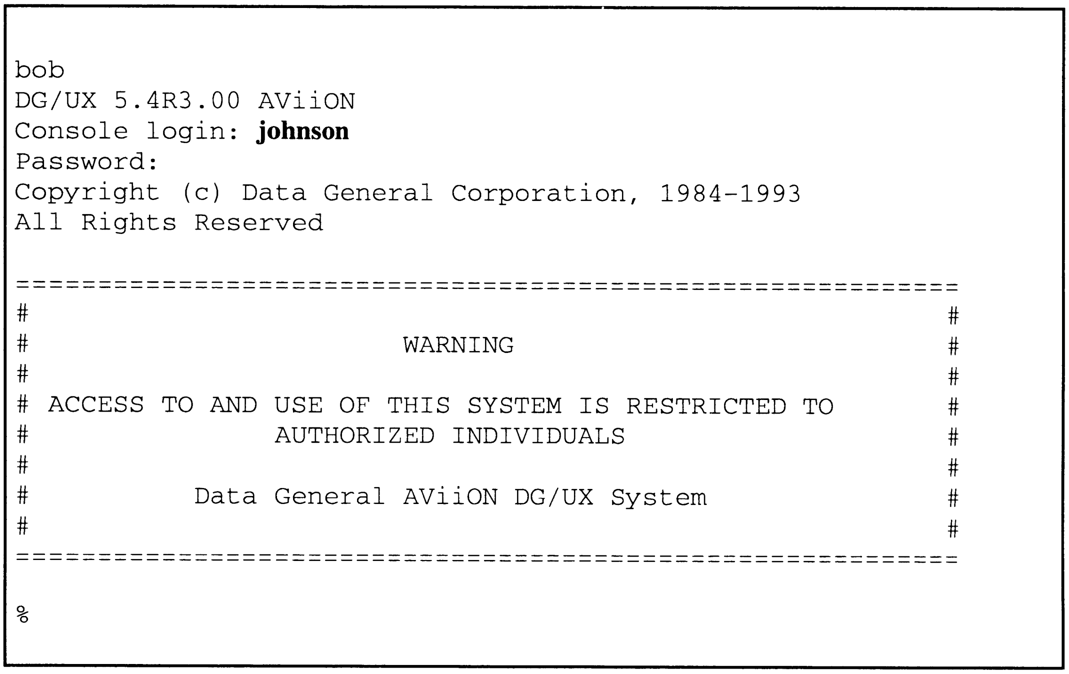
Failsafe Is a toggle button limiting startup to a single Xterm window.

IMPORTANT: If you choose to exit the X Window environment and continue the procedures in a single VT100 terminal screen environment, move the cursor to the "Terminate" button and click. To return to the X Window environment, enter **xdm** at the login prompt. The X Window environment will be restored.

A single window and a login prompt appear.

Logging in with an alphanumeric display terminal

Figure 11–4 shows the login display for an alphanumeric terminal.



```
bob
DG/UX 5.4R3.00 AViiON
Console login: johnson
Password:
Copyright (c) Data General Corporation, 1984-1993
All Rights Reserved

==========================================================
#                                                        #
#                        WARNING                         #
#                                                        #
# ACCESS TO AND USE OF THIS SYSTEM IS RESTRICTED TO      #
#               AUTHORIZED INDIVIDUALS                   #
#                                                        #
#          Data General AViiON DG/UX System              #
#                                                        #
==========================================================

%
```

Figure 11–4 Alphanumeric Display Terminal as a System Console

In this example, **bob** is the hostname that was specified during the TCP/IP setup. The login banner appears, followed by the % C-shell prompt.

Shutting down the DG/UX system

If multiple users are logged in to the system, before shutdown broadcast a message warning that you are about shut down. See the **wall**(1M) manual page for information on sending a broadcast message to all users logged in the system. An example of such a warning follows.

```
# /etc/wall ↵
Five minutes until system shutdown. Please log off. ↵
<Ctrl-D>
```

When you want to shut down, from the system console type these shell commands:

```
# cd / ↵
# shutdown -g30 -y ↵
```

The first command changes the current directory to / (root). The second command specifies a grace period of 30 seconds before shutdown begins. Also, it includes an affirmative response (yes) to start the shutdown. Otherwise, a confirmation request appears, requiring a response. The following messages appear:

```
Shutdown started.      Wed June 16 11:08:57 DST 1993

Shutdown is complete.
```

Your system has been shut down. Type

```
# halt ↵
```

Control goes to the SCM. When the SCM prompt appears, the DG/UX system has shut down.

Where to go next

This chapter ends the body of this manual. After learning about boot paths and logging in, you may want simply to run the DG/UX system, or perhaps look at some of the conceptual material in *Managing the DG / UX™ System.*

If there are OS clients that will boot for the first time, you need to set up software packages and mount file systems before their computers can do useful work. Chapter 6 describes package setup procedures. Chapter 3 describes procedures to add local and remote file systems.

For information on common system administrator tasks, use the task table shown in the Preface, "About this manual."

End of Chapter

A Basic DG/UX system concepts and commands

This appendix offers a few tips to get you on your feet when customizing your system. It discusses the following:

- The shell environment
- Directory structure and file system navigation
- Viewing and editing files
- Using manual pages

For a complete discussion of the DG/UX system's user environment, including complete coverage of the shells, user commands, and editors, see *Using the DG / UX™ System* and *Using the DG / UX™ Editors*.

The shell environment

The shell is the program that provides the basic command-line interface on the DG/UX system. When you see a prompt such as #, %, or $ on your screen, you know you are working in a shell.

There are three shells available on the DG/UX system, but for the purposes of this discussion, you do not need to know the differences among them. It is important to know, however, that the prompt you see on the screen changes when you become the superuser. For a normal user, the shell prompt is % or $; for the superuser, the shell prompt is #.

Directory structure and file system navigation

The DG/UX system directory structure is like an inverted tree in that branches grow from a central root. The root is the directory /, called the root directory. The root directory contains files (such as your kernel, **/dgux**) and directories (such as **/tmp**).

Any directory may contain files and subdirectories. The pathname of a file or directory represents the hierarchy of directory names that leads from the root directory to the given file or directory. For example, the pathname of the **cat** command is **/usr/bin/cat**, indicating that the **cat** program (a file) resides in the **bin** directory, which resides in the **usr** directory, which resides in the / (root) directory. You may specify any file or directory uniquely by using its entire pathname.

Every process running on the system, including your shell, has a current directory. The significance of a current directory is that it allows the process to run (or execute) a file or directory without a full pathname, using a name relative to its current directory. Commands to run a file or directory specified using relative path-naming are successful only if the current directory is specified in your search path (a period (.) — signifying the home directory — must be in the path). See *Using the DG/UX™ System* for information on setting the PATH variable.

To change your current directory, use the **cd** command:

```
# cd / ↵
```

The preceding command puts you in the root directory. You could have also changed to the root directory from **/admin** by using this form of the **cd** command:

```
# cd .. ↵
```

The **. .** (dot-dot) notation refers to the next highest directory, regardless of your current position in the file system structure. To return to the **/admin** directory from the root directory, use this command:

```
# cd admin ↵
```

To change from the **/admin** directory to the **/usr** directory, you could use a **cd** command specifying an absolute pathname, like this:

```
# cd /usr ↵
```

Or you could use a **cd** command specifying a relative pathname, like this:

```
# cd ../usr ↵
```

To return to your home directory (where you were when you first logged into the system), use the **cd** command without an argument:

```
# cd ↵
```

To see your current directory, use the **pwd** command:

```
# pwd ↵
```

Once in a directory, list the directory's contents with the **ls** command. The **ls** command has a number of options, of which two commonly used ones are

-a This option lists all files in a directory. Normally, the **ls** command lists all files except those beginning with . (period). Files starting with dots are typically personal configuration files, such as those in your home directory that initialize your environment. The **.profile** file in the **/admin** directory is an example of such a file.

-l This option produces a long listing. By default, the **ls** command lists only the names of files and directories. The long listing includes several attributes of the listed file or directory.

An example of running the **ls** command follows:

```
# ls -l chap1 ⤶
-rw-r--r--   1 smith   doc  22319 Sep 25 10:32 chap1
```

Each part of the output line is explained as follows:

`-rw-r--r--` The mode is a series of hyphens or letters indicating the type of file and the permissions. If the first character is **d**, the file is a directory. If the first character is -, the file is a regular file. This example shows a regular file. There are letters other than **d**.

The next nine letters indicate permissions: the first three are the owner permissions; the next three are the group permissions; and the last three are permissions for other users. The letter **r** represents read access; **w** represents write access; and **x** represents execute access for executable files (such as programs) and search access for directories. The symbol- indicates that the permission is denied. The **rwx** - letters are a subset of the full set of permissions. See the **ls**(1) manual page for a complete list. You change the permissions of a file with the **chmod** command.

`1` The link count is the number of physical pointers linked to that file. Ignore the link count for now.

`smith` This field indicates the username of the file's owner. When you create a file or directory, the system attaches your name to the file as the owner. You change the ownership of a file with the **chown** command.

`doc` This field indicates the name of a user group for which you may assign special permissions. Group permissions appear in the mode of the file or directory. When you create a file or directory, the system attaches your group to the file. You change the group of a file with the **chgrp** command.

`22319` This is the size of the file in bytes. For directories,this value represents the size of the directory data itself, not the contents of the directory.

`Sep 5 10:32` The date and time stamp tells when the file was last modified. If the file has not been modified since it was created, the date and time stamp indicate creation date and time.

`chap1` This is the name of the file or directory.

See *Using the DG / UX™ System* for more information on the **ls** command.

Viewing and editing files

There are several commands that let you view the contents of a file. For short files, such as the **sysadm** profile's shell initialization file, **.profile**, use the **cat** command:

cat /admin/.profile ↵

The **cat** command prints the entire file to the screen without stopping.

For longer files, such as the DG/UX system release notice, use the **more** command:

more /usr/release/dgux_5.4.3.rn ↵

The **more** command prints text to your display one screen at a time, allowing you to advance to the next screen by pressing the space bar. To exit **more** before you reach the end of the file, press **q**. To list other commands while using **more**, press **h** or ?.

The **more** command is good for files that contain special attributes such as highlighting. The **more** command does not, however, allow you to navigate through a file as easily as the **view** command,which you invoke like this:

# view /usr/release/dgux_5.4.3.rn ↵

The **view** command allows you to move through a file much like an editor because, in fact, **view** is the same as the DG/UX editor **vi** except that **view** opens the file in a read-only mode.

Table A–1 shows a few example commands that you can use to move through a file in **view** and **vi**.

Table A–1 Basic vi Navigation Commands

Command	Description
<Ctrl–F>	Move forward one screen.
<Ctrl–B>	Move backward one screen.
j	Move forward one line.
25j	Move forward 25 lines.
k	Move backward one line.
0	Go to the beginning of the line.
$	Go to the end of the line.
l	Move forward one character.
h	Move backward one character.
w	Move forward one word.
b	Move backward one word.
/string	Search forward for string.
?string	Search backward for string.
G	Move to the end of the file.
:1	Move to the first line of the file.
<Esc>	Return to command mode.
:q	Quit (exit) the file.
:q!	Quit (exit) the file without saving changes.

As demonstrated in the **25j** command, you can specify numeric arguments before a command to repeat it a number of times. The searching commands, / and ?, and the last-line mode commands, which start with **:**, move the cursor temporarily to the bottom of the screen. Pressing Enter executes the command and returns the cursor to the text area of the screen.

The **vi** editor operates in two modes: command mode and input mode. When you first invoke **vi**, it is in command mode. In command mode, you can navigate through the file as shown in Table A–1 and issue editing commands. Some editing commands perform functions like deleting or copying a character, word, line, and so on. Others place you in input mode where you can type text into the file. To exit from input mode and return to command mode, press **<Esc>**.

Table A–2 shows some example **vi** commands for changing a file.

Table A–2 Basic vi Editing Commands

Keystroke	Description
i	Enter input mode, inserting text at current cursor position.
I	Enter input mode, inserting text at the beginning of the line.
A	Enter input mode, appending text to the end of the line.
o	Enter input mode,opening a line below the current line and moving text in it.
O	Enter input mode, opening a line before the current line and entering text in it.
dd	Delete the current line.
dw	Delete to the end of the current word.
x	Delete the character under the cursor.
yy	Copy ("yank") the current line.
p	Insert ("put") the most recently yanked or deleted line below the current line.
P	Insert ("put") the most recently yanked or deleted line before the current line.
<Esc>	Exit input mode and return to command mode.
:w	Write changes to the file.
:w file2	Write changes to a file called **file2**.
:wq	Write changes to the file and exit ("quit") the file.
:q!	Exit ("quit") the file without saving changes.

For more information, see *Using the DG/UX™ System Editors.*

Using manual pages

A manual page is an on-line document that contains a technical description of all attributes for each command, system call, or special file, collectively referred to as utilities, in the DG/UX system.

This manual makes frequent references to particular man pages that you can consult for more information. Use the **man** command to access these on-line man pages. The next section discusses how to display manual pages.

Displaying a manual page (man)

A man (short for manual) page is a well-known convention used by the developers of the UNIX system to document their system within the system. Each command is identified by a number in parenthesis that signifies the type of entity being documented. Table A–3 lists the types of entries.

Table A–3 Man Page Types

Number	Class
0	Table of contents and permuted keyword–in–context index.
1	Commands and application programs
2	System calls
3	Subroutines and libraries
4	File format
5	Miscellaneous
6	Communications protocols
7	System Special files
8	System maintenance procedures

With the **man** command, you can access each on-line manual page, which is appended with an identifying parenthetical number such as **man**(1) and **ls**(1). For example, to see the man page for the **man** command, you would type the following.

```
$ man man ↵
```

Many man pages are too long to fit on one screen so they scroll quickly up and off the screen. To insert a pause between screens, you can use the **more** command along with the **man** command to control the output display. For example:

```
$ man man | more -f ↵
```

IMPORTANT: The vertical bar is referred to as a pipe. For more information on pipes, see the manual *Using the DG/UX™ System.*

At the bottom of the first screen, you will see this message displayed:

```
-- More --
```

Press the space bar to display the next screen of the man page. At the top of each man page is the command name, such as **man**(1). Each man page is formatted to contain the following categories of information, where appropriate.

NAME Names and briefly describes the utility.

SYNOPSIS Gives utility syntax; the command name is presented in boldface type, and its options are presented in regular type; optional arguments are enclosed in brackets. An ellipsis (...) following an argument indicates you can repeat it.

DESCRIPTION Gives a more detailed account of the utility and its arguments.

EXAMPLE	Illustrates the use of a command or concept.
FILES	Lists any system files used by the utility.
DIAGNOSTICS	Gives possible error messages and recovery actions.
SEE ALSO	Lists any related utilities.
BUGS/CAVEATS	Informs you of any known programming and design bugs or limitations.

To print a man page, use this command format:

man –Tlp [*single–digit*] *manpage-name* **| lp –S iso–88591 -d** *printer-name*

where:

–Tlp specifies a printer as a terminal type.

[*single–digit*] *manpage-name* specifies an optional digit (**0–9**) corresponding to the desired type of man page. See Table A–3 for man page types.

–lp is the print command.

–S iso–88591 specifies the character set in which the set of man pages are formatted.

–d *printer–name* identifies the printer's name.

An example follows:

```
$ man –Tlp 1 more | lp –S iso–88591 –d laser ↵
```

This command prints the **more** man page to a printer named **laser**.

End of Appendix

B DG/UX I/O device names

Each DG/UX system device has a unique name which is constructed using the DG/UX device naming format. This appendix describes the following topics.

- Standard and nonstandard I/O devices
- SCSI disk and tape device names
- Network controller device names
- SMD and ESDI disk drive names
- Non–volatile error correcting memory device names
- Synchronous line controller names
- Asynchronous line controller names
- Other I/O device names (keyboard, graphics display, parallel line printer)
- Device nodes

You need to know I/O device names under these circumstances:

- To specify tape and disk drives for booting DG/UX and other software.
- To specify tape and disk drives for configuring a custom DG/UX kernel.
- To recognize and specify disk and tape drives when formatting disks.
- To recognize I/O devices that are configured into the kernel and listed in the system file.
- To specify nonstandard I/O devices to be probed and configured.
- To recognize device nodes that are created in the **/dev** directory after the kernel is booted.

Two forms of the DG/UX device-naming format are available for specifying devices: short and long. Usually, you will use the short form to specify I/O devices. However, each short form is internally represented by its long form. Nonstandard I/O devices and device nodes are represented exclusively in the long form. Syntax for both forms is given in later sections in this appendix.

Standard and nonstandard I/O devices

A standard I/O device is one you purchased from Data General for use with AViiON computers. Its device ID has been preset through a jumper or DIP switch setting at the factory. A special file located at **/usr/etc/probedevtab** lists names for all possible standard devices. The standard I/O device settings allow you to refer to them using a short form of the device name, which is just the I/O device name and ID (or unit number).

An I/O device can be nonstandard if you add a device to your configuration for which there are no remaining standard device names. Or you might rejumper a standard I/O 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. This setting duplicates the standard address of an existing device controller in the configuration. Therefore, you would have to rejumper the existing device controller to a nonstandard address in long format. Refer to *Programming in the DG / UX™ Kernel Environment* for information on determining the controller addresses.

In addition, you may try to add nonsupported I/O devices to a configuration. A nonsupported I/O 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 *Writing a Standard Device Driver for the DG / UX™ System* 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.

Remaining sections of this appendix list the classes of standard devices and explain the syntax of their DG/UX device names.

SCSI disk and tape device names

This section explains SCSI disk and tape drive and controller names, including disk-array storage system drive and controller names. For examples of the disk and tape drive names, see the next section, "Disk and tape controller device names."

Disk and tape drive names

The format for SCSI disk and tape drives follows.

device (*controller-name*,[[*vme-controller*] *controller-SCSI–ID*], *device-SCSI-ID*, *LUN*) where:

device is as a two-letter mnemonic that 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 a CSS, PHU, or CLARiiON disk-array storage system)

st SCSI tape (on a CSS, PHU, or CLARiiON tape-array storage system)

controller-name specifies the associated controller, such as **ncsc** or **dgsc**. See the next 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-SCSI–ID is necessary only if you connect two AViiON computers to a single drive in a dual-initiator configuration. See *Managing the DG/UX™ System* for details about this configuration. Table B–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 not part of a CLARiiON disk-array storage system or for a **st** tape, this is the SCSI ID number on the disk or tape drive which is jumpered at the factory. Table B–1 lists the default SCSI IDs.

For a CLARiiON disk-array storage system, this field indicates the SP (storage-control processor) SCSI ID. The disk-array storage system can have up to two SPs. The SP SCSI IDs are user selectable, 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 6-23 (hexadecimal).

Table B–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	4
Second cartridge tape	5
First controller or SCSI-2 adapter channel	7
Second controller or SCSI-2 adapter channel	6

These default SCSI ID number assignments are conventions only; any SCSI drive can be rejumpered as SCSI ID 0 through 6.

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 next section.

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 B–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 B–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 B–2 lists the standard SCSI adapter/controller device names alphabetically by device name. Devices with values outside the ranges shown are nonstandard.

Table B–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(FFFFE300)
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)

continues

Table B–2 SCSI Controller and 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)

```
                 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 on SCSI ID 0. Since the disk is at LUN 0, the LUN is not required, although we show it 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.

Example 2: SCSI disk on insc controller (long form)

```
                      sd(insc@7(FFF8A000),0,0)
                      |    |   | |          | |
                      |    |   | |          | |
          device _    |    |   | |          | |__  logical-unit-number
                           |   | |          |      (LUN)
                           |   | |          |
   controller-type ___     |   | |          |__ SCSI-ID-number of disk
                               | |
       device-code _____       | |_____controller-address
```

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)

```
                      sd(insc(0),3,0)
                      |    |   | | |__diskette drive logical-unit
                      |    |   | | |  number (LUN)
                      |    |   | |
          device ____ |    |   | |__SCSI-ID-number of disk
                           |   |
   controller-type_____    |   |__controller-number
```

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)

```
                      sd(dgsc(0),0,2)
                      | |    | | |__ disk drive logical unit (LUN) number
                      | |    | |       (hex)
                      | |    | |
          device __ | |    | |__ SCSI-ID-number of SP (storage-control
                        |    |       processor)
                        |    |
   controller-type___   |    |___SCSI-2 adapter controller-number
```

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)

```
                        sd(dgsc(vme(1),(0),0,2))
                        |   |        |   |  |  |  disk drive logical unit
                        |   |        |   |  |  |__(LUN) number (hex)
                        |   |        |   |  |
             device____ |   |        |   |  |__ SCSI-ID-number of SP
                            |        |   |         (storage-control processor)
                            |        |   |
       controller-type_____ |        |   |__SCSI-2 adapter
                                     |        controller-number
                                     |
                                     |__VME channel number
```

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)

```
                        da(hada(1),12)
                        |   |    |  |
                        |   |    |  |
            device ____ |   |    |  |__ disk drive logical unit number (LUN)
                            |    |      (hex)
                            |    |
       controller-type_____ |    |____  controller-number
```

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)

```
                        sd(ncsc(0,7),0,0)
                        |  |    |  | | |
                        |  |    |  | | |
            device_____ |  |    |  | | |___logical-unit-number (LUN)
                           |    |  | |
         controller-type___|    |  | |____SCSI-ID-number of disk drive
                                |  |
          controller-number___  |  |___controller SCSI-ID number
```

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.

Network controller device names

The format for the network controller devices follows:

```
device [@device-code] ( [vme-controller] [controller-address [, secondary-address
        [, alternate-address ] ] ] )
```

where:

device is a mnemonic that specifies the network controller device. The valid values are

cien	CMC VME Ethernet intelligent controller
dgen	Data General second generation integrated Ethernet controller
hken	Interphase VME Ethernet controller
inen	Integrated Ethernet controller
pefn	Interphase VME Fiber Distributed Data Interface (FDDI) controller
vitr	VME token ring controller

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it is 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. If you omit this, **vme(0),** is implied. For example, the device name **pefn(vme(0),1)** is functionally the same as **pefn(1)**. This item does not apply to the integrated controllers **dgen** and **inen**.

controller-address Valid controller addresses follow:

cien	Controller's VME A32 address
dgen	Not used
hken	Interphase VME Ethernet controller
inen	Not used
pefn	Controller's VME A16 address
vitr	Controller's VME A32 address

Table B–3 lists the valid controller numbers supported (short form), the device code, and controller address (long form).

secondary-address is different for each device; they follow:

cien	Not used
dgen	Not used
hken	Controller's VME A32 address
inen	Not used
pefn	Not used
vitr	Alternate token ring address that overrides board-set default

Table B–3 lists the valid controller numbers supported (short form), device code, and controller address (long form). Controllers whose values fall outside this range are considered nonstandard.

alternate-address is different for each device; they follow:

cien	Alternate Ethernet address that overrides board-set default
dgen	Alternate Ethernet address that overrides board-set default
hken	Alternate Ethernet address that overrides board-set default
inen	Alternate Ethernet address that overrides board-set default
pefn	Not used
vitr	IBM product ID (18 hex digits) that overrides board-set default

Table B–3 Network Controller Names

Controller Number (Short Form)	**Controller Address (Long Form)**
CMC Ethernet Controller (cien)	
cien(0)	cien@48(E1800000)
cien(1)	cien@49(E1900000)
cien(2)	cien@4A(E1A00000)
cien(3)	cien@4B(E1B00000)
cien(4)	cien@4C(E1C00000)
cien(5)	cien@4D(E1D00000)
cien(6)	cien@4E(E1E00000)
cien(7)	cien@4F(E1F00000)
Data General Second Generation Integrated Ethernet Controller (dgen)	
dgen(0)	Not used.
dgen(1)	Not used.
dgen(2)	Not used.
dgen(3)	Not used.
dgen(4)	Not used.
dgen(5)	Not used.
Interphase VME Ethernet Controller (hken)	
hken(0)*	hken@15(FFFF4000,E1000000)
hken(1)*	hken@16(FFFF5000,E1080000)
hken(2)	hken@10(FFFF4200,E1100000)
hken(3)	hken@11(FFFF4400,E1180000)
hken(4)	hken@12(FFFF4600,E1200000)
hken(5)	hken@13(FFFF4800,E1280000)
hken(6)	hken@14(FFFF4A00,E1300000)
hken(7)	hken@17(FFFF4C00,E1380000)

* For computers limited to 1 Gbyte or less of physical memory, these are
hken(0)
hken(1)
	
	hken@15(FFFF4000,55900000)
	hken@16(FFFF5000,55980000)

Integrated Ethernet Controller (inen)	
inen(0)	Not used.
Interphase VME FDDI Controller (pefn)	
pefn(0)	pefn@78(FFFF2000)

Continued

Table B–3 Network Controller Names

Controller Number (Short Form)	Controller Address (Long Form)
Interphase VME FDDI Controller (pefn)	
pefn(0)	pefn@78(FFFF2000)
pefn(1)	pefn@79(FFFF2200)
pefn(2)	pefn@7A(FFFF2400)
pefn(3)	pefn@7B(FFFF2600)
pefn(4)	pefn@7C(FFFF2800)
pefn(5)	pefn@7D(FFFF2A00)
pefn(6)	pefn@7E(FFFF2C00)
pefn(7)	pefn@7F(FFFF2E00)
VME Token Ring Controller (vitr)	
vitr(0)*	vitr@40(E4000000)
vitr(1)*	vitr@41(E4002000)
vitr(2)	vitr@42(E4004000)
vitr(3)	vitr@43(E4006000)
vitr(4)	vitr@44(E4008000)
vitr(5)	vitr@45(E400A000)
vitr(6)	vitr@46(E400C000)
vitr(7)	vitr@47(E400E000)

* For computers limited to 1 Gbyte or less of physical memory, these are

vitr(0)	vitr@40(61000000)
vitr(1)	vitr@41(61002000)

Controllers whose values fall outside this range are considered nonstandard.

Examples 1 and 3 are short and long forms of the same device.

Example 1: VME token ring controller (short form)

```
              vitr()
                   |
                   |
   controller __  |
```

This device name identifies the first VME token ring controller.

Example 2: VME token ring controller on second vme controller channel (short form)

```
                  vitr(vme(1),0)
                     |______  |
                     |  |     |
       controller __ |  |     |__ controller number
                        |
                        |__vme-controller
```

This device name identifies the first the first VME token ring controller on the second VME channel.

Example 2: VME token ring controller (long form)

```
                 vitr@40(E4000000)
                    |    |    |
                    |    |    |
   controller   __  |    |    |
                         |    |
   device-code ______    |    |____  A32 address
```

This device name identifies a VME token ring controller whose device code is 40 and A32-address is E4000000.

Other examples

hken()	First () Interphase VME Ethernet LAN.
pefn(1)	Second (1) Interphase FDDI controller.

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 B–4 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 B–4 SMD and ESDI Disk Drive Controller Names

Controller Number (Short Form)	**Controller Address (Long Form)**
Ciprico ESDI Controller (cied)	
cied(0)	cied@18(FFFFEE00)
cied(1)	cied@19(FFFFF100)
cied(2)	cied@1A(FFFFFB00)
cied(3)	cied@1B(FFFFFD00)
Ciprico SMD Controller (cimd)	
cimd(0)	cimd@18(FFFFEE00)
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)

```
                 cied (0, 0)
                 |     | |
                 |     | |
controller-type  |     | |_____logical-unit-number
                       |
                       |____ controller-number
```

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)

```
                 cied@18(FFFFEE00),0)
                 |     | |          |
                 |     | |          |
controller-type  |     | |          |____unit-number
                       | |
     device-code_______| |
                         |__ VME A16 address
```

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 B–5 lists the valid device short forms and long forms.

Table B–5 Valid Device Forms for NVRAM Memory

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 following 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 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
```

If the configured device **nvrd(0)** were mirrored, the mirrored board would have the following values:

Device Code	**81**
Control Register Address	**FFFFFF10**
Memory Address	**E5000000**

Synchronous line controller names

The format for the synchronous line controllers follows.

device ([vme–controller] controller-number)
or
device@device–code ([vme–controller] VME A32 address)

where:

device is a mnemonic that specifies the synchronous line controller. The valid values are

iscd	Integrated synchronous chip controller
ssid	VME VSC synchronous controller
vsxb	VME VSC/3i synchronous controller

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 **vsxb(vme(0),1)** is functionally the same as **vsxb(1)**. This does not apply to the integrated controller **iscd**.

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it represents the interrupt vector.

controller-number identifies the controller the device is attached to.

VME A32 address identifies the address of the controller that the device is attached to.

Table B–6 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form).

Asynchronous line controller names

The format for asynchronous line controllers is as follows:

device[@device-code]([vme–controller] controller-number, line number)

where:

device is a mnemonic that specifies the asynchronous line controller. The valid values are

duart Dual-line asynchronous terminal controller

syac Systech VME asynchronous terminal controller

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it 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. If you omit this, **vme(0),** is implied. For example, the device name **syac(vme(0),1)** is functionally the same as **syac(1)**. This does not apply to the integrated controller **duart**.

controller-number identifies the controller the device is attached to.

VME A32 address identifies the address of the controller that the device is attached to.

IMPORTANT: For the **duart** device, it refers to the address of its memory-mapped control registers (when using the long form). See Table B–6.

line-number identifies the particular line attached to the controller, starting at 1. For **duart**, either 0 or 1.

Table B–6 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form). Controllers whose values fall outside this range are considered nonstandard.

Table B–6 Synchronous and Asynchronous Line Controller Device Names

Controller Number (Short Form)	Controller Address (Long Form)
Integrated Synchronous Chip Controllers	
iscd()	Not used.
VME VSC Synchronous Controller (ssid)	
ssid(0)	ssid@50(55B00000)
ssid(1)	ssid@51(55B10000)
ssid(2)	ssid@52(55B20000)
ssid(3)	ssid@53(55B30000)
ssid(4)	ssid@54(55B40000)
ssid(5)	ssid@55(55B50000)
ssid(6)	ssid@56(55B60000)
ssid(7)	ssid@57(55B70000)
ssid(8)	ssid@58(E2080000)
ssid(9)	ssid@59(E2090000)
ssid(A)	ssid@5A(E20A0000)
ssid(B)	ssid@5B(E20B0000)
ssid(C)	ssid@5C(E20C0000)
ssid(D)	ssid@5D(E20D0000)
ssid(E)	ssid@5E(E20E0000)
ssid(F)	ssid@5F(E20F0000)
VME VSC/3i Synchronous Controller (vsxb)	
vsxb(0)	vsxb@90(E3400000)
vsxb(1)	vsxb@91(E3410000)
vsxb(2)	vsxb@92(E3420000)
vsxb(3)	vsxb@93(E3430000)
vsxb(4)	vsxb@94(E3440000)
vsxb(5)	vsxb@95(E3450000)
vsxb(6)	vsxb@96(E3460000)
vsxb(7)	vsxb@97(E3470000)
vsxb(8)	vsxb@98(E3480000)
vsxb(9)	vsxb@99(E3490000)
vsxb(A)	vsxb@9A(E34A0000)
vsxb(B)	vsxb@9B(E34B0000)
vsxb(C)	vsxb@9C(E34C0000)
vsxb(D)	vsxb@9D(E34D0000)
vsxb(E)	vsxb@9E(E34E0000)
vsxb(F)	vsxb@9F(E34F0000)

Continued

Controller Number (Short Form)	Controller Address (Long Form)
Systech VME Asynchronous Terminal Controller (syac)	
syac(0)*	syac@60(E3000000)
syac(1)*	syac@61(E3020000)
syac(2)*	syac@62(E3040000)
syac(3)*	syac@63(E3060000)
syac(4)*	syac@64(E3080000)
syac(5)	syac@65(E30A0000)
syac(6)	syac@66(E30C0000)
syac(7)	syac@67(E30E0000)
syac(8)	syac@68(E3100000)
syac(9)	syac@69(E3120000)
syac(A)	syac@6A(E3140000)
syac(B)	syac@6B(E3160000)
syac(C)	syac@6C(E3180000)
syac(D)	syac@6D(E31A0000)
syac(E)	syac@6E(E31C0000)
syac(F)	syac@6F(E31E0000)
* For computers limited to 1 Gbyte or less of physical memory, these are	
syac(0)	syac@60(60000000)
syac(1)	syac@61(60020000)
syac(2)	syac@60(60040000)
syac(3)	syac@61(60060000)
syac(4)	syac@61(60080000)
Dual–Line Asynchronous Terminal Controller (duart)	
duart(0)	duart@4(FFF82000)
duart(1)** ***	duart@10(FFF82040)
duart(2)** ***	duart@17(FF782000)
duart(3)** ***	duart@18(FF782040)
** Not all computers support this duart. *** For AViiON 300-series and 400-series systems, duart(1) is duart@10(FFF82C00). For AViiON 530 systems, duart(1) is duart@10(FFF83220).	

Examples 1 and 2 are short and long forms of the same device.

Example 1: Asynchronous terminal controller (short form)

```
                 syac (0,4)
                   |    | |
                   |    | |
controller type__ |    | | __line number
                        |
                        |____  controller-number
```

This device name identifies the first (0) **syac** controller on the fourth (4) line.

Example 2: Asynchronous terminal controller (long form)

```
                  syac@60(60000000)
                   |    |   |
                   |    |   |
controller type__  |    |   |
                        |   |
 device code_________   |   |
                            |__ VME A32 address
```

This device name identifies the **syac** controller whose device code is 60 and whose VME A32 address is 60000000.

Example 3: VME VSC synchronous interface controller (long form)

```
                   ssid@50(55B00000)
                   |      |  |
                   |      |  |
 controller type _ |      |  |
                          |  |
 device code____________  |  |
                             |__ VME A32 address
```

This device name identifies the **ssid** controller whose device code is 50, and VME-A32 address is 55B00000.

Example 4: VME VSC/3i synchronous controller (long form)

```
                  vsxb@90(E3400000)
                   |    |   |
                   |    |   |
controller type__  |    |   |
                        |   |
 device code_________   |   |
                            |__ VME A32 address
```

This device name identifies the **vsxb** controller whose device code is 90, and VME-A32 address is E3400000.

Examples 5 and 6 are short and long forms of the same device.

Example 5: Duart (short form)

```
                  duart(0,0)
                   |    | |
                   |    | |
controller type__  |    | |__ line number
                        |
                        |____ controller-number
```

Example 6: Duart (long form)

```
            duart@4(FFF82000,0)
              |   | |        |
device name__ |   | |        |
                  | |        |___ line-number
                  | |
                  | |___Memory-mapped control register address
                  |
                  |___device code
```

This device name identifies the first (0) device attached to the **duart** whose device code is 4 and memory-mapped control register address is FFF82000.

This device name identifies the first (0) device attached to the first (0) **duart**. This device could also be specified as **duart(**).

Other I/O device names

The format for the other I/O devices follows:

device [@*device-code*]

where:

device is a mnemonic that specifies the particular device. The valid values are

kbd	Keyboard
grfx	Graphics display
lp	Parallel line printer

device-code has two meanings. For integrated devices, it is an internal representation; for VME devices, it represents the interrupt vector.

Table B–7 lists the valid controller numbers supported (short form) and the corresponding device code and controller address (long form). I/O devices with values outside this range are nonstandard.

Table B–7 Other I/O Device Names

Controller Name (Short Form)	Controller Address (Long Form)
Graphic Display, Keyboard, and Line Printer (grfx, kbd,lp)	
grfx(0)	Not used
kbd(0)	Not used
lp(0)	Not used

Device nodes

The following sections describe device nodes for physical and virtual disks, tapes, terminals, and all other devices supported by the DG/UX system. Device nodes are automatically created in the **/dev** directory each time you boot the kernel.

A device node is a special file that provides a handle to a particular device so that programs can access it. Device nodes come in two forms:

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

See the **mknod**(1M) manual page.

Physical disk nodes

Each physical disk 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),2,0)	Block node for **sd(insc(0),2,0**
/dev/rpdsk/sd(insc@7(FFF8A000),2,0)	Character node for **sd(insc(0),2,0)**

Virtual and logical disk nodes

Virtual and logical disks are accessible 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/rdsk/comm

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:

```
                                               [{l}]  [{c}]
/dev/rmt/tape@device-code(address,SCSI-ID,LUN) [{m}]  [{u}]  [n]
                                               [{h}]
```

Where: **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.

c or **u** mean use compression mode and use un compressed 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)

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

Terminal nodes

Terminal line devices have nodes with names of the form **/dev/tty***number*, where *number* is a decimal number of at least two digits starting at 00. The assignment of names to your system's terminal lines depends on the order that the terminal line controller devices are configured by the kernel, which in turn depends on the order those controller devices are listed in the system file. Terminal nodes are assigned in a first-come, first-served order, starting at **/dev/tty00** and working up. If you use the standard kernel auto configuration mechanism, all standard **duart** controllers on the system are listed in order, followed by all standard **syac** controllers on the system.

Only one asynchronous line is available on some **duart** devices. On stand-alone computer systems, this is because one channel of the first **duart** is used to provide access to the mouse device, which has the special device node **/dev/mouse**. Other systems also reserve one **duart** channel for special use, in this case as the system console (**/dev/syscon**).

A **syac** controller causes from 16 to 255 device nodes to be created, depending on how many lines are on the controller. If a cluster controller box is used, all 255 lines are created, even though only a few of them (perhaps 8 or 16) are actually in use.

End of Appendix

C Using Data General terminals on the DG/UX system

This appendix presents guidelines for making your Data General terminals operational in a DG/UX environment. It covers the following topics:

- Recommended terminal settings
- Selecting character set and emulation mode
- Setting the line discipline

The capabilities of your terminals are based on a number of factors, some of which you can control. Given your terminal and keyboard type, you can set the TERM environment variable using the DG/UX shell and select a terminal hardware emulation mode through the terminal's firmware setup menu to support a specific terminal behavior and desired character set. Definitions of the TERM variable, terminal emulation mode, and character set follow.

TERM variable

The operating behavior of your terminal is determined by associated characteristics located in a compiled database for terminal and printer device capabilities **/usr/lib/terminfo/?/*** where ? stands for the first character of the name, and * stands for the device name. For example, **/usr/lib/terminfo/d/d215** is the **terminfo** entry for Data General's DASHER D215 terminal and terminals that behave like it. You select the appropriate terminal behavior by assigning the **terminfo** entry to the TERM variable.

Terminal emulation mode

A terminal emulator is a program that simulates the operation of a particular terminal model. An emulation mode can be defined by a formal standard or by a de facto industry standard (that is, a specific implementation of a terminal). An example of the former is the American National Standards Institute (ANSI) X3.64-1979 document, implemented as the ANSI mode of operation on older Data General terminals. An example of the latter is the command set of the Digital Equipment Corporation VT100 terminal, implemented as the VT100 emulation mode on modern DG terminals. For example, if you have a DASHER D215 terminal that you set to VT100 mode, its keys will be redefined to behave as the keys on a DEC VT100 terminal.

Supported emulation modes follow:

- VT: a set of terminal behaviors built into VT100 terminal series, which are considered Data General's modern terminals.

- DG-UNIX: a Data General proprietary mode designed for UNIX compatibility. It is a superset of the DG mode (terminal behaviors built into older Data General terminals that run on Data General proprietary systems). The DG/UX system does not support DG mode.
- ANSI: a set of terminal behaviors built into Data General's older terminals such as the D211.

Character set

Terminals must be encoded with a character set to manipulate data. A variety of standards is available; your terminals must conform to one. They follow:

- ISO 8859-1 (Western European language conformance).
- VT Multinational (International conformance designed for the Data General VT terminal series); subset of ISO 8859-1.
- DGI (proprietary Data General International).
- U.S. ASCII; a subset of ISO 8859-1, VT Multinational, and DGI.

Your terminals are probably already set up to behave according to one of these standards. However, you can set the desired character set for some terminals.

Recommended terminal settings

Try to set your terminals to operate in VT or ANSI emulation mode in a DG/UX environment. These modes are most compatible with the features a UNIX system expects a terminal to offer.

The following tables detail configurations of Data General terminals, keyboards, emulation modes, and TERM variable settings. For each configuration, it specifies the supported character set and function keys. The legend following the table defines abbreviated terms.

Table C–1 focuses on the “older” terminals.

Table C–1 Older Terminal Configuration Information

Terminal Type	7–Bit TERM Variable	7–Bit Character Set	8–Bit TERM Variable	8–Bit Character Set
D210	d210	USASCII	n/a	DGI
D211	d211–7b	USASCII	d211	DGI
D214	d214	USASCII	n/a	DGI
D215	d215–7b	USASCII	d215	DGI
D220	d220–7b	USASCII	d220	DGI
D410	d410–7b	USASCII	d410	DGI
D411	d411–7b	USASCII	d411	DGI
D460	d460–7b	USASCII	d461	DGI
D461	d461–7b	USASCII	d461	DGI
D470C	d470c–7b	USASCII	d470c	DGI
D555	d555–7b	USASCII	d555	DGI
D577	d577–7b	USASCII	d577	DGI
D578	d578–7b	USASCII	d578	DGI

Your TERM variable selection will depend on whether your terminal is configured for 7-bit or 8-bit communications. Each of these terminal types is used with a 6246 keyboard type and runs in ANSI emulation mode. Consult your terminal's hardware documentation for this information.

All function keys used with and without the Shift, Ctrl, and Ctrl-Shift control keys will work properly.

Table C–2 focuses on the "newer" terminals.

Table C–2 Newer Terminal Configuration Information

Configuration Elements				Resulting Features	
Terminal Type	**Keyboard Type**	**Emulation Mode**	**TERM Variable**	**Internat'l Character Set**	**Function Keys Supported**
D1400i	6488	wyse 60	d1400i	None	F1–F9
	6488	wyse 60	wyse60	None	F1–F9
	6488	wyse 50+	wyse50	None	F1–F9
	6488	vt100	vt100	VTM	F1–F4, Num Keypad (,456789)
	6488	vt220–7	vt220	VTM	F1–F12
	6488	vt220–8	vt220	VTM	F1–F12
D216 D216E	6348	vt100	vt100	None	C1–C4, Num Keypad (,456789)
	6348	vt100	d216	None	C1–C4
D216+ D216E+	6348	vt100	vt100	None	C1–C4, Num Keypad (,456789)
	6488	vt100	vt100	None	F1–F4, Num Keypad (,456789)
	6348	vt100	d216	None	C1–C4
	6488	vt100	d216	None	F1–F4
	6488	vt100	d216	DGI	F1–F4
	6348	dg–unix	d216+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D217	6348	vt100	vt100–fk	None	C1–C4, F1–F12
	6488	vt100	vt100–fk	None	F1–F12
	6348	vt100	vt100	None	C1–C4, Num Keypad (,456789)
	6488	vt100	vt100	None	F1–F4, Num Keypad (,456789)
	6348	vt100	d217	None	C1–C4, Num Keypad (,456789)
	6488	vt100	d217	None	F1–F4, Num Keypad (,456789)
	6348	dg–unix	d217–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d217–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift

Continued

Table C–2 Newer Terminal Configuration Information

Configuration Elements				Resulting Features	
Terminal Type	**Keyboard Type**	**Emulation Mode**	**TERM Variable**	**Internat'l Character Set**	**Function Keys Supported**
D230C	6384	ANSI	d220–7b	None	F1–F15, Shift, Ctrl, Ctrl–Shift
	6348	ANSI	d220	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6348	ANSI	d230c	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D412	6348	vt220	vt220	VTM	F1–F18
	6348	vt220	d412	VTM	F1–F15
D412+	*6348	vt320	vt220	VTM/ISO	F1–F18
	*6488	vt320	vt220	VTM/ISO	F1–F12
	*6348	vt320	vt412	VTM/ISO	F1–F15
	6348	dg–unix	412+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d412+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D413	6348	vt320	vt220	ISO	F1–F18
	6488	vt320	vt220	ISO	F1–F12
	6348	dg–unix	d413–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d413–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D462	6348	vt220	vt220	VTM	F1–F18
	6348	vt220	d462	VTM	F1–F15
D462+	*6348	vt320	vt220	VTM/ISO	F1–F18
	*6488	vt320	vt220	VTM/ISO	F1–F15
	*6348	vt320	d462	VTM/ISO	F1–F15
	6348	dg–unix	d462+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d462+	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
D463	6348	vt320	vt220	ISO	F1–F18
	6488	vt320	vt220	ISO	F1–F12
	6348	vt320	d463	ISO	F1–F15, Shift, Ctrl, Ctrl–Shift
	6348	dg–unix	d463–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift
	6488	dg–unix	d463–unix	DGI	F1–F15, Shift, Ctrl, Ctrl–Shift

LEGEND

None	No international character set support is provided; however, the U.S. ASCII character set is supported.
6348	CEO–style 107–key keyboard.
6246	Older version of CEO–style 107–key keyboard.
6488	101–key keyboard (PC/AT layout; 102 keys for international.

DGI	Proprietary DG International.
VTM	VT Multinational.
ISO	ISO 8859–1 (requires setup sequence on D412+ and D462+).
*	For D412+ or D462+ terminals, must be a firmware Revision 3 or higher to support correctly in vt320 mode.
Num Keyboard (,456789)	Includes these keys only: , 4 5 6 7 8 9
Shift, Ctrl, Ctrl–Shift	Keys used in conjunction with (press and hold while pressing) a function key F1–F15.
F1–F4	Function keys F1, F2, F3, and F4 only.
vt100/vt220/vt320 emulation	DEC VT100, VT220, and VT320 terminal emulation will cause a mismatch between a keyboard's key labels and their corresponding functions. Use the adhesive labels provided with your hardware documentation for correct key representation. Only the function keys operate correctly in VT100, VT220, or VT320 emulation modes. The Shift and Ctrl keys are not supported. One exception, however, is the D1400i terminal which can operate in any of these modes and correctly matches function key labels to corresponding functions.
C1–C4	Four keys labeled C1, C2, C3, and C4 on the keypad.

Selecting character set and emulation mode

You can explicitly select your terminal's character set for only the D217, D413, and D463 terminal models. Most older terminals are set to only one character set. See your hardware documentation for more information.

Select the desired terminal emulation mode by invoking the terminal's configuration menu or by setting the DIP switches on the rear of the terminal, depending on the model of terminal.

Using the D216 as an example, press Shift-N/C to invoke the terminal's port menu so that you can select the VT100 operating mode.

Setting the line discipline

Line discipline is the term that applies to the options that interpret terminal characteristics.

Terminal options that are probably most important to you are the definitions of the line-editing and control keys.

Table C–3 lists the most frequently used keys with their common settings.

Table C–3 Commonly Used Line-Editing and Control Keys

Function	Definition	Common Setting
intr	Terminates a process.	Ctrl–C
quit	Terminates a process but dumps an image of memory that can be examined later.	Ctrl–|
erase	Removes a character and closes up the space.	Ctrl–H
kill	Deletes a line.	Ctrl–U
eof	End–of–file (logs you out); also terminates interactive input for commands such as **mailx** (an electronic mail facility that enables the exchange of messages) and **cat** (a command that allows you to send text to a file via the terminal).	Ctrl–D
start	Resumes scrolling of text on the screen.	Ctrl–Q
stop	Stops scrolling of text on the screen.	Ctrl–S
susp	Suspends temporarily an active job.	Ctrl–Z
werase	Removes a word and closes up the space.	Ctrl–W

With the **stty** (set terminal type) command, you can find out the current settings for the line discipline. Typical output for the **stty everything** shell command follows:

```
% stty everything
speed 9600 baud; line = 1;
rows = 50; columns = 80; ypixels = 754; xpixels = 739;
intr = ^c; quit = ^|; erase = ^h; kill = ^u;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = ^y;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl cread -clocal -loblk -parext
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
  ixon -ixany -ixoff -imaxbel
isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop echoctl -echoprt echoke -defecho -flusho -pendin iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
%
```

The caret (^) represents the Control key.

If you have an older Data General terminal operating in ANSI emulation mode, you **must** change the line discipline to enable tab expansion and disable new line expansion to the carriage return/new line function. Type this command:

```
% stty nl tab3 ↵
```

To change any key settings, use this command format:

stty *key-name character*

When you redefine keys, use the caret symbol (^) to represent the Control key. Also, you must precede it (and any other special character that is used as a metacharacter by the shell), with a backslash (\) or you must surround the entire key definition with a pair of single or double closed quotation marks. (' ' or " "). A metacharacter is a symbol that stands for something other than itself, such as the caret (^) which in the shell represents the first character in the line. Examples follow:

```
% stty intr \^C ↵
% stty erase \^\? ↵
% stty eof \^D ↵
```

Type the **stty** command again to see the change in the line discipline.

When selecting a new key definition, make sure that the key is not already being used. Also, if you use the **editread** facility, those keys should not conflict with keys set through the **stty** command. Refer to *Using the DG/UX™ Editors* for more detailed information on **editread**.

Once you make changes to your line discipline from the shell, they will remain in effect only while you are currently logged in. You will have to reset them each time you log in. If you prefer, you can put them in a setup file (such as your **.login**) so that they will be set automatically each time you log in.

See the **stty**(1) manual page for more information on setting line discipline.

Setting the TERM variable

Set the **TERM** environment variable. Using the D216 terminal in a C shell environment as an example, you would use the following shell commands:

```
% setenv TERM vt100 ↵
% tput init ↵
```

The **tput** command sets up such terminal capabilities as cursor movement, function key sequences, and screen highlighting (reverse video).

IMPORTANT: If you are having problems with terminal control, check to make sure that the **TERM** variable is set correctly for your terminal. Also try executing the command **tput reset** to reset your terminal to its normal operating mode.

End of Appendix

D Worksheets

This appendix contains copies of worksheets you filled 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
- Remote file systems
- User accounts
- OS clients
- X terminal client networks
- Secondary releases
- Tty line controllers and tty lines

If you think you may need additional copies, copy the blank worksheet(s) before completing them.

Virtual disk worksheets

Figures D–1 and D–2 are virtual disk planning worksheets, one as the first page, the next as an additional page (if you need an additional page).

Virtual Disk Layout Worksheet (Page 1)

Virtual Disk or Mirror Name	Mount Point Directory	Drive Name ______________ ______ Mbytes		Drive Name ______________ ______ Mbytes		Drive Name ______________ ______ Mbytes	
		Piece or Image	Size in Blocks	Piece or Image	Size in Blocks	Piece or Image	Size in Blocks
swap	NA	1					
root	/	1					
usr	/usr	1					
usr_opt_X11							
usr_opt _networker							
var_opt _networker							
usr_opt_xdt							
Total Used							
Total Capacity							
Free Space							

Figure D–1 Virtual Disk Planning Worksheet, Page 1

Virtual Disk Layout Worksheet (Page of)

Virtual Disk or Image Name	Mount Point Directory	Drive Name ________ ______ Mbytes		Drive Name ________ ______ Mbytes		Drive Name ________ ______ Mbytes	
		Piece or Image	Size in Blocks	Piece or Image	Size in Blocks	Piece or Image	Size in Blocks
Total Used							
Total Capacity							
Free Space							

Figure D–2 Virtual Disk Planning Worksheet, Second or Subsequent Page

Remote file system worksheet

Figure D–3 shows a remote file system planning worksheet.

Remote File System Planning Worksheet

Remote Hostname	Remote Mount Point	Local Mount Point
Sample: cobra	*/pdd/sam/image*	*/udd/elixer/image*

Figure D–3 Remote File System Planning Worksheet

User accounts worksheet

Figure D–4 shows a worksheet for planning user accounts. You will probably need additional (blank) copies; if so, copy the original blank worksheet before completing it.

User Account Planning Worksheet

User Name (Login Name)	Group Name	Shell

Figure D–4 User Account Planning Worksheet

OS client worksheets

This section provides worksheets for recording information for OS clients and X terminals:

- Figure D–5 and D–6 are OS Client Network Planning Worksheets.
- Figure D–7 is a Local Root and Swap Virtual Disks Configuration Planning Worksheet
- Figure D–8 is an X Terminal Client Network Planning Worksheet.

Client Network Planning Worksheet (Part 1 of 2)

OS Client Hostname	Internet Address	Ethernet Address
Example: junior	*123.227.3.14* *(use periods)*	*08:00:1B:03:45:11* *(use colons)*

Figure D–5 Client Network Planning Worksheet (Part 1 of 2)

Client Network Planning Worksheet (Part 2 of 2)

ONC and TCP/IP Parameters	Example Value	Actual Value
NIS domain name	my_domain	
Do you subnet?	yes	
Network mask	0xffffff00	

Figure D–6 Client Network Planning Worksheet (Part 2 of 2)

Local Root and Swap Virtual Disk Configuration Planning Worksheet

Parameter Type	Example Value	Actual Value
OS client hostname	junior	
OS client Internet address	128.222.3.86	
OS client Ethernet address	08:00:1B:18:03:11	
OS client disk drive name	sd(insc(0),0,0)	
OS server hostname	boss	
OS server Internet address	128.222.3.120	
Temporary kernel name	dgux.temp	
NIs domain name	my_domain	
Do you subnet?	yes	
Network mask	0xffffff00	
OS client's Internet address (hexadecimal equivalent*)	80DE0356	
*The screen of the computer used as the OS client shows the Internet address in hexadecimal format the first time it boots its temporary kernel to a run level of 1. You can record its hexadecimal address at that point for later use. See Chapter 7, the section "Loading the root (/) file system on the OS client's local disk" for the procedure that produces that screen message.		

Figure D–7 Local Root and Swap Virtual Disks Configuration Planning Worksheet

X Terminal Client Network Planning Worksheet

X Terminal Client Hostname	**Internet Address**	**Ethernet Address**
Example: junior	*123.227.3.14* *(use periods)*	*00:00:A7:00:04:9F* *(use colons)*

Figure D–8 X Terminal Client Network Planning Worksheet

Secondary release worksheets

This section provides four worksheets to help you plan a secondary release of the DG/UX system. If you plan to add DG/UX 5.4 Release 2.01/2.10 as a secondary release, use the first pair of worksheets, Figures D–9 and D–10, together. If you are planning to add DG/UX 5.4 Release 3.00 as a secondary release, use the second pair of worksheets, Figures D–11 and D–12. When you install a secondary release, you create logical or virtual disks, depending on the DG/UX revision the server is running.

For each set of worksheets, use the first worksheet to calculate disk sizes; use the second worksheet for disk information.

DG/UX 5.4 Release 2.01/2.10 Virtual or Logical Disk Planning Worksheet (Part 1 of 2)

Virtual or Logical Disk Name	Basis for Calculating Disk Size	Virtual or Logical Disk Size
srv_swap	(50,000 * *number–of–clients*) +10%	
root_dgux_542	(40,000 * *number–of–clients*)	
usr_dgux_542	240,000	
usr_opt_X11_dgux_542	105,000	
usr_opt_aview_dgux_542	8,000	

Figure D–9 DG/UX 5.4 Release 2.01/2.10 Virtual or Logical Disk Planning Worksheet (Part 1 of 2)

DG/UX 5.4 Release 2.01/2.10 Virtual or Logical Disk Planning Worksheet (Part 2 of 2)

		Drive Name ____________ ____ Mbytes		Drive Name ____________ ____ Mbytes		Drive Name ____________ ____ Mbytes	
Virtual or Logical Disk Name	**Mount Point Directory**	**Piece #**	**Blocks**	**Piece #**	**Blocks**	**Piece #**	**Blocks**
srv_swap	/srv/swap	1					
root_dgux_542	/srv/release/dgux_542_root	1					
usr_dgux_542_usr	/srv/release/dgux_542/usr	1					
usr_opt_X11_dgux_542	/srv/release/dgux_542/usr/ opt/X11	1					
usr_opt_aview_dgux_542	/srv/release/dgux_542/usr/ opt/aview	1					
Total Used							
Total Capacity							
Free Space							

Figure D–10 DG/UX 5.4 Release 2.01/210 Virtual or Logical Disk Planning Worksheet (Part 2 of 2)

DG/UX 5.4 Release 3.00 Virtual or Logical Disk Planning Worksheet — Part 1 of 2

Virtual or Logical or Disk Name	Basis for Calculating Disk Size	Virtual or Logical Disk Size
srv_swap	(50,000 * *number–of–clients*)	
root_dgux_54R300	(*number–of–clients* * (40,000 – kernel-size)) + kernel-size	
usr_dgux_54R300	240,000	
usr_opt_X11_dgux_54R300	140,000	
usr_opt_aview_dgux_54R300	10,000	

Figure D–11 DG/UX 5.4 Release 3.00 Virtual or Logical Disk Planning Worksheet (Part 1 of 2)

DG/UX 5.4 Release 3.00 Virtual or Logical Disk Planning Worksheet — Part 2 of 2

		Drive Name ________ ____ Mbytes		Drive Name ________ ____ Mbytes		Drive Name ________ ____ Mbytes	
Virtual or Logical Disk Name	**Mount Point Directory**	**Piece or Image**	**Blocks**	**Piece or Image**	**Blocks**	**Piece or Image**	**Blocks**
srv_swap	/srv/swap	1					
root_dgux_54R300	/srv/release/dgux_54R300_root	1					
usr_dgux_54R300	/srv/release/dgux_54R300/usr	1					
usr_opt_X11_dgux_54R300	/srv/release/dgux_54/usr/opt/X11	1					
usr_opt_aview_dgux_54R300	/srv/release/dgux_54R300/usr/opt/aview	1					
Total Used							
Total Capacity							
Free Space							

Figure D–12 DG/UX 5.4 Release 3.00 Virtual or Logical Disk Planning Worksheet (Part 2 of 2)

Tty controller and tty line worksheets

This section contains the following tty controller and tty line worksheets.

- Figure D–13 is terminal line controllers worksheet.
- Figure D–14 is a RS-232/422 ports on computer unit worksheet
- Figure D–15 is a VAC/16 controller worksheet.
- Figure D–16 is a VDA host adapter worksheet
- Figure D–17 is a tty lines worksheet

Terminal Line Controllers Worksheet

Board No.	Device Name	Configuration File Position	Board or Port Type	Cluster Controllers			
				Address	No. Lines	Address	No. Lines
	duart()	1					
	duart(1)						
0	syac()	2		01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
1	syac(1)	3		01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
2	syac(2)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
3	syac(3)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	
4	syac(4)			01		09	
				02		0A	
				03		0B	
				04		0C	
				05		0D	
				06		0E	
				07		0F	
				08		10	

Figure D–13 Terminal Line Controllers Worksheet

RS-232/422 Ports On Computer Unit Worksheet

Port type:		**Device name:** *duart()*
tty Line	**Device Type**	**Description**
Port type:		**Device name:** *duart(1)*
tty Line	**Device Type**	**Description**

Figure D–14 RS-232/422 Ports On Computer Unit Worksheet

VAC/16 Controller Worksheet

Board no: *0*			**Device name:** *syac()*			**Range of tty lines:** *01 –16*	
Port No.	**tty Line**	**Device Type**	**Description**	**Port No.**	**tty Line**	**Device Type**	**Description**
0				8			
1				9			
2				10			
3				11			
4				12			
5				13			
6				14			
7				15			

Figure D–15 VAC/16 Controller Worksheet

VDA Host Adapter Worksheet

Sheet _____ of _____

			Board type:	Board no:	Device name		Range of tty lines:		
Cluster Address	Port No.	tty Line	Device Type	Description	Cluster Address	Port No.	tty Line	Device Type	Description
	0					0			
	1					1			
	2					2			
	3					3			
	4					4			
	5					5			
	6					6			
	7					7			
	8					8			
	9					9			
	10					10			
	11					11			
	12					12			
	13					13			
	14					14			
	15					15			

Figure D–16 VDA Host Adapter Worksheet

tty Lines Worksheet — Sheet ___ of ___

tty Line	Device Type	Lineset or Model	TERM Variable or Printer Name	tty Line	Device Type	Lineset or Model	TERM Variable or Printer Name

Figure D–17 tty Lines Worksheet

End of Appendix

E Localizing your DG/UX system

Localizing your DG/UX system means tailoring it for a specific language or locale. This appendix describes how to localize your DG/UX™ system using such internationalization features as ISO and 8-bit code sets. Locale databases let you set up an environment for a particular language, country, or area of a country.

DG/UX system internationalization is based on the UNIX® System V Release 4 Multi-National Language Supplement (MNLS).

The appendix describes the following topics:

- Using non-ASCII code sets
- Accessing and creating locale databases
- Taking advantage of library changes

Using non-ASCII code sets

This section describes how to use non-ASCII code sets such as Latin 1 (Latin Alphabet No. 1, defined by ISO standard 8859–1). The section describes the following tasks:

- Converting between code sets
- Printing non-ASCII characters
- Displaying non-ASCII characters
- Converting keyboard input

Converting between code sets

A code set is a collection of characters (visual glyphs) and a way of representing the characters with numeric values; U.S. ASCII (ISO 646) is a well known code set. A code set used by an application program when processing data is called an internal code set. A code set used by a terminal or printer is an external code set.

The DG/UX system supports the following internal code sets:

ISO 8859–1 (Latin 1)
ISO 8859–2 (Latin 2)
ISO 8859–3 (Latin 3)
ISO 8859–4 (Latin 4)
ISO 8859–5 (Latin/Cyrillic)
ISO 8859–7 (Latin/Greek)

- PC 437 (United States)
- PC 850 (Multilingual)
- PC 860 (Portugal)
- PC 863 (Canada-French)
- PC 865 (Norway)

External code sets supported by the DG/UX system include the internal code sets and the following:

- ISO 646 and ISO 646 national variants for the following:
  - Denmark
  - France
  - Great Britain
  - Germany
  - Italy
  - Norway
  - Portugal
  - Spain
  - Sweden
  - Yugoslavia
- DEC Multilingual
- Roman 8
- EBCDIC
- Data General International (DGI)

The DG/UX system provides the most complete support for Latin 1. Character terminals, keyboards, printers, and workstations that support Latin 1 can be used with the DG/UX system. With the exception of U.S. ASCII, each of these code sets requires 8 bits per character. DG/UX commands and libraries properly handle 8-bit characters.

The DG/UX system provides limited support for the DGI and DEC Multilingual code sets. DG/UX utilities transform keyboard input from these code sets to Latin 1. On output to a CRT, DG/UX utilities transform Latin 1 into either DGI or DEC Multilingual. For example, to use a DGI terminal on the DG/UX system, run the script shown in the **codeset_conv**(5) man page.

To convert a file from one code set to another, use the **iconv** program. For example, to convert an ISO 8859–1 file named **latin1** to an ASCII file named **ascii** using the best fit available for each character, enter this command:

```
iconv –f 88591 –t ASCII –m b latin1 > ascii
```

For more information about **iconv**, see the **iconv**(1) man page.

Printing non-ASCII characters

The DG/UX system supports the printing of non-ASCII characters. If you have a line printer that handles the characters you are printing, you can simply use the **lp** command. If you are using a PostScript® printer, you must use the **–S** option on **lp**. For example, to print a Latin 1 file named **europel** on a printer whose queue is named **pslaser**, use the following command:

# lp –d pslaser –S iso–88591 europel↵

For more information, see the **lp**(1) and **postprint**(1) man pages.

Displaying non-ASCII characters

To display Latin 1 characters on an AViiON workstation, simply run **mterm** or **xterm** to create a window. To display characters from the Data General International character set, use **mterm –d216** or **mterm –d410**.

To display Latin 1 characters on a terminal, you need a terminal that supports Latin 1. Data General terminal models d217, d413, and d463, or later models in those series support Latin 1 when the terminal is in ISO mode. For more information, see the manual for your terminal.

Converting keyboard input

The DG/UX system lets you enter non-ASCII characters from an ASCII keyboard. A STREAMS-based **tty** driver and a line discipline module (**att_kbd**) handles single-byte and multibyte character I/O.

In an **mterm** window, you can enter Latin 1 characters from an ASCII keyboard using **mterm**'s built-in compose keys. To view the compose-key sequences, position your mouse cursor on View in the border of the **mterm** window and click the left mouse button. To enter a non-ASCII character, press the Pause key and then type the two-key compose sequence.

At a terminal or in a window, you can enable compose keys by running the following script, in which the ` character is an accent grave.

```
stty_settings=`stty -g`
strchg -p
strchg -p
strchg -h att_kbd
strchg -h ldterm
strchg -h ttcompat
stty $stty_settings
# stty -g bug workaround; set stty values as appropriate:
stty line 1 \
intr \^c erase \^\? kill \^u swtch \^g susp \^n dsusp \^n werase \
\^t -brkint -istrip ixany ixoff echoe echoctl echoke iexten tab0
kbdload /usr/lib/kbd/88591.cpz
kbdset -a 88591.cpz
```

After you have run the script, you can enter all Latin 1 characters from an ASCII keyboard. The compose keys also let you enter certain ASCII characters such as tilde (~) that may not be on all keyboards. To use a compose key sequence, type Ctrl-T followed by the two-character sequence listed in the **cpz**(4M) man page.

Accessing and creating locale databases

This section describes how to access and create localization databases in **/usr/lib/locale**. These databases contain native language, cultural, and regional data.

Accessing locale databases

A locale contains localization data for a particular language, territory, and code set. Each locale has categories that contain information for a specific area of localization. A locale contains the categories described in Table E–1, listed by the name of the environment variable used to control them. For specific information about each environment variable, refer to **environ**(5).

Table E–1 Locale Environment Variables

Variable	Description
LC_COLLATE	Collating Sequence. Data defining collating sequences to be used for a specific single-byte code set.
LC_CTYPE	Character Classification and Conversion. Data defining character classes for the code set used by the locale.
LC_MONETARY	Currency Representation. Data defining currency format based on locale conventions.
LC_NUMERIC	Numeric Representation. Data defining numeric format based on locale conventions.
LC_TIME	Date and Time Format. Data defining date and time display format and language.
LC_MESSAGES	USL-style Message Catalogs. Determines the native-language used to display system error messages stored in USL-style message catalogs. Not recommended for applications that are intended to be portable (see the IMPORTANT note below).
LANG	Default. LANG is used as the default locale if the corresponding environment variable for a particular category is unset. Also, used to locate X/Open-style message catalogs and is the portable method for locating USL-style message catalogs. The system-wide default value for LANG can be changed with the **sysadm**(1M) command.
NLSPATH	X/Open-style Message Catalogs. Defines location of native-language system error message catalogs in X/Open format. Uses the value of LANG to determine which locale catalogs are used. The system-wide default value for NLSPATH can be changed with the **sysadm**(1M) command.

DG/UX categorizes native language, code set, cultural and region-specific data by locale, where a locale name has the form *language*[*_territory*[*.codeset*]]. ISO 8859–1 is the default code set if you omit *.codeset* in the locale name. You can customize your system to use data from the locales listed in Table E–2.

Table E–2 Predefined DG/UX Locales

Locale Name	Language	Country	Code Set	Alternate Name
C	English	United States	ASCII	C–locale
da	Danish	Denmark	ISO 8859–1	da_DK
da_DK.850			PC 850	
da_DK.865			PC 865	
nl	Dutch	Netherlands	ISO 8859–1	nl_NL
nl_NL.437			PC 437	
nl_NL.850			PC 850	
nl_BE	Dutch	Belgium	ISO 8859–1	
nl_BE.437			PC 437	
nl_BE.850			PC 850	
en	English	United Kingdom	ISO 8859–1	en_GB
en_GB.646			ISO 646	
en_GB.437			PC 437	
en_GB.850			PC 850	
en_AU	English	Australia	ISO 8859–1	
en_AU.646			ISO 646	
en_AU.437			PC 437	
en_AU.850			PC 850	
en_CA	English	Canada	ISO 8859–1	
en_CA.646			ISO 646	
en_CA.437			PC 437	
en_CA.850			PC 850	
en_US	English	United States	ISO 8859–1	
en_US.646			ISO 646	
en_US.437			PC 437	
en_US.850			PC 850	
fi	Finnish	Finland	ISO 8859–1	fi_FI
fi_FI.437			PC 437	
fi_FI.850			PC 850	
fr	French	France	ISO 8859–1	fr_FR
fr_FR.437			PC 437	
fr_FR.850			PC 850	
fr_BE	French	Belgium	ISO 8859–1	
fr_BE.437			PC 437	
fr_BE.850			PC 850	

Continued

Table E–2 Predefined DG/UX Locales

Locale Name	Language	Country	Code Set	Alternate Name
fr_CA	French	Canada	ISO 8859–1	
fr_CA.437			PC 437	
fr_CA.850			PC 850	
fr_CH	French	Switzerland	ISO 8859–1	
fr_CH.437			PC 437	
fr_CH.850			PC 850	
de	German	Germany	ISO 8859–1	de_DE
de_DE.437			PC 437	
de_DE.850			PC 850	
de_AT	German	Austria	ISO 8859–1	
de_AT.437			PC 437	
de_AT.850			PC 850	
de_CH	German	Switzerland	ISO 8859–1	
de_CH.437			PC 437	
de_CH.850			PC 850	
el	Greek	Greece	ISO 8859–7	el_GR
is	Icelandic	Iceland	ISO 8859–1	is_IS
is_IS.850			PC 850	
it	Italian	Italy	ISO 8859–1	it_IT
it_IT.437			PC 437	
it_IT.850			PC 850	
it_CH	Italian	Switzerland	ISO 8859–1	
it_CH.437			PC 437	
it_CH.850			PC 850	
no	Norwegian	Norway	ISO 8859–1	no_NO
no_NO.850			PC 850	
no_NO.865			PC 865	
pl	Polish	Poland	ISO 8859–2	pl_PL
pt	Portuguese	Portugal	ISO 8859–1	pt_PT
pt_PT.850			PC 850	
pt_PT.860			PC 860	
ru	Russian	Russia (CIS)	ISO 8859–5	ru_SU
sh	SerboCroatian	Yugoslavia	ISO 8859–2	sh_YU
es	Spanish	Spain	ISO 8859–1	es_ES
es_ES.437			PC 437	
es_ES.850			PC 850	
sv	Swedish	Sweden	ISO 8859–1	sv_SE
sv_SE.437			PC 437	
sv_SE.850			PC 850	

Continued

Table E–2 Predefined DG/UX Locales

Locale Name	Language	Country	Code Set	Alternate Name
tr	Turkish	Turkey	ISO 8859–3	
sk	Czech	Czechoslovakia	ISO 8859–2	

To use locale data from a single locale for all categories, set the environment variable, LANG, to the desired locale name. T his is convenient when one locale can meet most of your data requirements. After setting LANG to a locale name, you can override this setting on an individual category basis by setting the individual environment variable corresponding to each category to a different locale name.

The language your system uses to display system messages is set using environment variables which are based on two types of message catalogs, X/Open and USL-style. These catalogs contain system and application messages separate from executable programs. The values of NLSPATH and LANG are used to locate X/Open message catalogs. The value of LANG is used to locate USL message catalogs, although you can override the value of LANG by setting LC_MESSAGES.

IMPORTANT: Use of LC_MESSAGES is not recommended in applications that are intended to be portable. For further details about X/Open and USL message facilities, refer to *Porting and Developing Applications for the DG / UX System*.

The following command sets all the locale categories to use data from the locale **fr**, with the exception of LC_MONETARY, which uses data from the locale **fr_CH**:

```
LANG=fr; LC_MONETARY=fr_CH; export LANG LC_MONETARY
```

Creating locale databases

The default location of locale data is **/usr/lib/locale**. The default location of X/Open-style message catalogs is **/usr/lib/nls/msg**. The **locale** directory and the **msg** directory each contains locale directories for specific locales.

To create a new locale, create a directory in **/usr/lib/locale** and create files in that directory. Table E–3 lists the programs you can use to create the database files.

Table E–3 Programs for Creating Locale Databases

Filename	Command	Description of File Information
LC_COLLATE	`colltbl colltbl.src`	Collating sequence
LC_CTYPE	`wchrtbl chrtbl.src`	Character classification and conversion
LC_MESSAGES/*	`mkmsgs` *catalog*	USL-style message catalogs in USL format
	`gencat -a` *catalog*	X/Open-style message catalogs in USL format
LC_MONETARY	`montbl montbl.src`	Currency representation
LC_NUMERIC	`chrtbl chrtbl.src`	Numeric representation
LC_TIME	`cp time.src LC_TIME`	Date and time format
***.cat**	`gencat` *catalog*	X/Open-style message catalogs in X/Open format

LC_MESSAGES contains a file named **Xopen_info**, which contains data not available in **LC_COLLATE**, **LC_CTYPE**, **LC_MONETARY**, and **LC_TIME**. The data in **Xopen_info** comprises the following:

- Default time format
- Default date format
- Default format for date and time
- An equivalent for "yes"
- An equivalent for "no"

While the locale databases specify which language to use for system error messages, the DG/UX system contains message catalogs in only one language, English. The DG/UX system provides several USL-format message catalogs in the **/usr/lib/locale/C/LC_MESSAGES** directory and a few X/Open-format message catalogs in the **/usr/lib/nls/msg/C** directory. Each USL-style catalog begins with **dg** or **ux**. Each X/Open-style catalog ends in **.cat**.

The DG/UX system has utilities for creating and accessing other message catalogs (see Table E–3), but it does not contain the catalogs themselves. Currently, catalogs of messages in French, German, and Spanish are available in Data General's Western European System Messages Release 1.0.

Example: creating local message catalogs

To create a set of local message catalogs, follow these steps:

1. Copy to a temporary directory all files in **/usr/lib/locale/C/LC_MESSAGES** that begin with **dg** or **ux**.

2. Copy to a temporary directory all files in **/usr/lib/nls/msg/C** that end with **.cat**.

3. Decompile the USL-format catalogs using **mkmsgs**. For example:

   ```
   # mkmsgs –d dgcore > dgcore.txt ↵
   # mkmsgs –d uxcore.abi > uxcore.abi.txt ↵
   # mkmsgs –d uxlibc > uxlibc.txt ↵
   # mkmsgs –d uxlp > uxlp.txt ↵
   # mkmsgs –d uxsyserr > uxsyserr.txt ↵
   ```

4. Decompile the X/Open-format catalogs using **gencat**. For example:

   ```
   # gencat –d catexstr.cat > catexstr.msg ↵
   # gencat –d exterr.cat > exterr.msg ↵
   ```

5. Edit the decompiled source files to change the text from English to the appropriate language. For example:

   ```
   # vi dgcore.txt uxcore.abi.txt uxlibc.txt uxlp.txt uxsyserr.txt ↵
   # vi catexstr.msg exterr.msg ↵
   ```

   When editing **dgcore.txt**, be sure to avoid deleting blank lines, as this will result in the wrong message being displayed later. Unlike the files whose names begin with **ux**, **dgcore** is an X/Open-style message catalog, even though it is in USL format; i.e., **dgcore** was created by **gencat –a** not **mkmsgs**. **gencat** cannot decompile files created via **gencat –a**.

6. Recompile the message catalogs. For example:

   ```
   # mkmsgs dgcore.txt dgcore ↵
   # mkmsgs uxcore.abi.txt uxcore.abi ↵
   # mkmsgs uxlibc.txt uxlibc ↵
   # mkmsgs uxsyserr.txt uxsyserr ↵
   # gencat exterr.cat exterr.msg ↵
   ```

7. Install the USL-format catalogs in *locale*/**LC_MESSAGES** and the X/Open-style catalogs in **msg/***locale*, where *locale* is the name of the locale for which the messages have been translated.

   To activate a set of messages for a particular locale, set the environment variable LANG or LC_MESSAGES to the name of the locale. You can make this the system default by setting it in **/etc/login.csh** and **/etc/profile**. An individual user can make it his or her default by setting it in **$HOME/.login** or **$HOME/.profile**. A user can also set it for a single shell session.

Taking advantage of library changes

The DG/UX system provides a number of library routines (C functions) to support internationalization. These routines are derived from MNLS. The following figure lists the man pages that describe these library routines. For more information about these library routines, see the relevant man page.

addsev(3C)	Define additional severities.
getopt(3C)	Get option letter from argument vector.
gettxt(3C)	Retrieve a text string.
lfmt(3C)	Display error message in standard format and pass to monitor and logger.
pfmt(3C)	Display error message in standard format.
setcat(3C)	Define default catalog.
setlabel(3C)	Define the label for pfmt() and lfmt().
vlfmt(3C)	Display error message in standard format and pass to monitor and logger.
vpfmt(3C)	Display error message in standard format and pass to monitor and logger.
getwc(3W)	Get wchar_t character from a stream.
getwidth(3W)	Get information of supplementary code sets.
getws(3W)	Get a wchar_t string from a stream.
mbchar(3W)	Multibyte character conversion.
mbstring(3W)	Multibyte string conversion.
printf(3W)	Print formatted output.
putwc(3W)	Put wchar_t character on a stream.
putws(3W)	Put a wchar_t string on a stream.
scanf(3W)	Convert formatted input.
ungetwc(3W)	Push wchar_t character back into input stream.
vprintf(3W)	Print formatted output of a variable argument list
wconv(3W)	Translate characters.
wctype(3W)	Classify ASCII and supplementary code set.
widec(3W)	Multibyte character I/O routines.
wstring(3W)	Wchar_t string operations and type transformation.
curses(3X)	CRT screen handling and optimization package.
curs_addwch(3X)	Add a wchar_t character to a curses window.
curs_addwchstr(3X)	Add string of wchar_t characters to a curses.
curs_addwstr(3X)	Add a string of wchar_t characters to a curses window.
curs_getstr(3X)	Get character strings from curses terminal keyboard.
curs_getwch(3X)	Get (or push back) wchar_t characters from curses terminal keyboard.
curs_getwstr(3X)	Get wchar_t character strings from curses terminal keyboard.
curs_inswch(3X)	Insert wchar_t character before cursor in curses window.
curs_instr(3X)	Get a string of characters from a curses window.
curs_inwch(3X)	Get a wchar_t character from a curses window.
curs_inwchstr(3X)	Get a string of wchar_t characters from a curses window.
curs_inwstr(3X)	Get a string of wchar_t characters from a curses window.
curs_pad(3X)	Create and display curses pads.
curs_printw(3X)	Print formatted output in curses windows.
curs_scanw(3X)	Convert formatted input from a curses window.

End of Appendix

Index

Within the index, a bold page number indicates a primary reference. A range of page numbers indicates that the reference spans those pages.

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:
COMPANY NAME ____________________
ATTN: ____________________
ADDRESS ____________________
CITY ____________________
STATE ____________________ ZIP __________

SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME ____________________
ATTN: ____________________
ADDRESS (NO PO BOXES) ____________________
CITY ____________________
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.

Customizing the
DG/UX™ System

093–701101–04

Cut here and insert in binder spine pocket