Addendum to Installing and Managing the DG/UX™ System

086-000172-00

This addendum updates manual 093-701052-02. Please see "Updating Your Manual."

Ordering No. 086-000172

Printed in the United States of America
Revision 00, September 1990

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, MA 01580

Addendum to Installing and Managing the DG/UX™ System
086-000172-00

A vertical bar in the margin of a page indicates a technical change from 093-701052-02. The addendum number appears on all pages in this addendum.

Updating Your Manual

This addendum (086-000172-00) to *Installing and Managing the DG/UX™ System* (093-701052-02) includes new and revised information effective with DG/UX 4.31. It also includes minor corrections.

To update your copy of 093-701052-02, please do the following:

Remove	**Insert**
Title/Notice	Title/Notice
iii through xxv	iii through xxvi
old Chapter 2	new Chapter 2
old index	new index

Insert this sheet immediately behind the new notice page.

Installing and Managing the DG/UX™ System

093-701052-02

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

Ordering No. 093-701052

Printed in the United States of America
Revision 02, June 1990

NOTICE

Installing and Managing the DG/UX™ System
093-701052

Revision History:	Effective with:
Original Release – April 1989	DG/UX Release 4.10
Revision 01 – August 1989	DG/UX Release 4.10
Revision 02 – June 1990	DG/UX Release 4.30
Addendum 086-000172-00 – September 1990	DG/UX 4.31

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

Preface

This manual is for system administrators who are familiar with the UNIX® operating system. This manual presents a thorough discussion of the installation procedure for all hardware and software configurations; for example, standalone system, OS client, and OS server, plus a variety of software packages. This manual addresses the first-time installation and an update of the system. If you have a standalone system with disk and tape units and the System Software Package for AViiON Systems or the Operating System Package for AViiON Systems, you may prefer to use a different Data General document, *A Quick Guide to Installing the DG/UX™ System on a Stand-Alone AViiON™ Workstation*. It is described further in the "Related Manuals" section of the Preface. This manual also gives information on how to operate and manage the installed system.

The DG/UX system provides the **sysadm** utility to help you with installation and management tasks. The **sysadm** utility is a menu-based interface to a number of programs intended to simplify the responsibilities of a system administrator. Using **sysadm** and its attendant programs, you can do things like load and set up software packages, add user profiles, manage the network and printers, and build customized DG/UX kernels. This manual tells how you can use **sysadm** to help with these and other tasks.

Are You Experienced?

We assume that you are not new to the UNIX system. You don't need programming experience to use this manual, but you must know

- the general file system layout of the UNIX operating system,
- how to use UNIX commands and a text editor, and
- how to use a shell and work within the UNIX directory structure.

Manual Outline

Before you install anything, we recommend that you read Chapters 1 through 4. The remaining chapters are based on tasks and are best used as you do those tasks.

This manual is composed of the following chapters and appendixes:

Chapter 1

The section "Using the sysadm Utility" discusses how to manage a server or client OS with **sysadm**. How to use **sysadm** menus for managing the OS of any AViiON system whether it is a stand-alone, server, or client system.

The section "Managing Servers and Clients" discusses how to manage the server/client group (servnet) environment. Focuses on what tasks need to be done and how to do them; not dwelling on "who" does what task. Explains concepts and relationships of servers and clients. Shows the sequence of a diskless client booting an operating system from an OS server.

Chapter 2

Planning your DG/UX system and loading the DG/UX system software. Shows how the DG/UX file system has been organized to support the server/client environment. Setting up a stand-alone or server host. Setting up diskless clients on a server. Examples of installations on various configurations are given. Resource planning worksheets are provided to help you plan for your installation.

Chapter 3

Operating the DG/UX system. Startup and shutdown, recovering from system trouble (e.g., crashes, power outages). Explanation of **rc** scripts and run levels.

Chapter 4

Reconfiguring the system on a routine basis. Setting tuneable configuration parameters. Setting system date and time.

Chapter 5

Managing releases, loading software, listing information about releases.

Chapter 6

Managing OS and X terminal client systems. Setting defaults; adding and deleting OS and X terminal clients.

Chapter 7

Using the disk management utility **diskman** to format physical disks, create logical disks, create file systems, check file systems, display physical disk information, update the operating system, and do various other disk tasks.

Chapter 8

File system management. Shows how to mount, unmount, add, and delete file systems, and make backup tapes.

Chapter 9

File information. Finding and displaying information on files and file systems according to age, size, name, and block use.

Chapter 10	Setting up and managing terminals.
Chapter 11	Managing line printers and laser printers. Contains information about start/stop scheduler, set defaults, cancel, enable/disable, and add/delete printers.
Chapter 12	Setting up UUCP files and directories.
Chapter 13	Network terms and definitions. Basic network management functions: setting parameters for NFS® and TCP/IP, adding/deleting hosts and networks.
Chapter 14	Adding user accounts, creating aliases and groups. Explains how these are used and managed differently in stand-alone and servnet situations.
Chapter 15	Monitoring how system resources are being used with accounting programs. Printing summary reports on system use.
Appendix A	Device names and codes for servers and workstations.
Appendix B	Directories and files. Descriptions of the directories and files that are shipped with the DG/UX system.
Appendix C	Error messages for the line printer (LP) subsystem and UUCP.
Appendix D	The **fsck** file system check and repair program: invoking, options, arguments, output, and error conditions.
Appendix E	Using CDROM, magneto-optical, and diskette SCSI devices on the DG/UX system.
Appendix F	Loading the 4.30 release of the DG/UX system on an installed 4.20 system.
Appendix G	Loading the 4.30 release of the DG/UX system on a system that does not have a tape device.

Related Manuals

The manuals listed below contain information that relates to some aspect of installing and managing the DG/UX operating system. Each of these manuals is mentioned again, later in this manual.

Data General Software Manuals

A Quick Guide to Installing the DG/UX™ System on a Stand-Alone AViiON™ Workstation (069-000520)

Tells how to install the DG/UX system on a workstation that has its own tape and disk devices. Intended for users who are unfamiliar with systems like the DG/UX system, the manual provides a cookbook approach to installation on preloaded as well as non-preloaded Data General AViiON systems.

Managing NFS® and Its Facilities on the DG/UX™ System (093-701049)

Shows how to install, manage, and use the DG/UX ONC™/NFS product. Contains information on the Network File System (NFS), the Yellow Pages (YP), Remote Procedure Calls (RPC), and External Data Representation (XDR). (NFS is a U.S. registered trademark of Sun Microsystems, Inc. ONC is a trademark of Sun Microsystems, Inc.)

Programmer's Reference for the DG/UX™ System (093-701055 and 093-701056)

Alphabetical listing of manual pages for programming commands on the DG/UX system. This two-volume set includes information on system calls, file formats, subroutines, and libraries.

Setting Up and Managing TCP/IP on the DG/UX™ System (093-701051)

Explains how to prepare for the installation of Data General's TCP/IP (DG/UX) package on AViiON computer systems. Contains information on tailoring the software for your site, managing the system, and troubleshooting system problems.

System Manager's Reference for the DG/UX™ System (093-701050)

Contains an alphabetical listing of manual pages for commands relating to system administration or operation.

User's Reference for the DG/UX™ System (093-701054)

Contains an alphabetical listing of manual pages for commands relating to general system operation.

Using TCP/IP on the DG/UX™ System (093-701023)

Introduces Data General's implementation of the TCP/IP family of protocols and describes how to use the package.

Using the DG/UX™ Editors (069-701036)

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

Using the DG/UX™ Kernel Debugger (093-701075)

Explains how to use the DG/UX kernel debugger to analyze the state of the kernel's internal data structures and the state of the underlying hardware's registers and memory.

Using the DG/UX™ System (069-701035)

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

For a complete list of software manuals available for the DG/UX system and related products, see the Documentation Set near the end of this manual.

Data General Hardware Manuals

Setting Up and Starting AViiON™ 300 Series Stations (014-001801).

Describes how to unpack, check, and install system components and options. Explains how to power up, run diagnostics, and load the operating system. Includes operational, physical, electrical, and environmental specifications of the computer unit, monitor, keyboard, and mouse.

Setting Up and Starting AViiON™ 5000 Series Systems (014-001806).

Describes how to unpack, check, and install system components and connect options. Explains how to power up, run diagnostics, and load the operating system. Includes operational, physical, electrical, and environmental specifications of the computer unit.

Using the AViiON™ System Control Monitor (SCM) (014-001802).

Describes how to use the SCM commands and menus to debug programs, control program flow, and boot media on AViiON RISC-based hardware.

Readers, Please Note

Data General manuals use certain symbols and styles of type to indicate different meanings. You should familiarize yourself with the following conventions before reading the manual.

Convention	Meaning
boldface	In command lines and format lines: Indicates text (including punctuation) that you type verbatim from your keyboard. All DG/UX commands, path names, and names of files, directories, and manual pages also use this typeface.
`constant width/ monospace`	Represents a system response on your screen. Syntax lines also use this font.
italic	In 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 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 should not be confused with the boldface brackets shown below.
[ ]	In 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.
...	In format lines and syntax lines: Means you can repeat the preceding argument as many times as desired.
`$` and `%`	In command lines and other examples: Represent the system command prompt symbols used for the Bourne and C shells, respectively.
↵	In command lines and other examples: Represents the New Line key. Note that on some keyboards this key might be called Enter or Return instead of New Line.
< >	In command lines and other examples: Angle brackets distinguish a command sequence or a keystroke (such as **<Ctrl-D>**, **<Esc>**, and **<3dw>**) from surrounding text.

086-000172 updates
093-701052-02

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, and you are within the United States or Canada, contact the Data General Service Center by calling 1-800-DG-HELPS for toll-free telephone support. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

Free telephone assistance is available with your warranty and with most Data General service options. Lines are open from 8:30 a.m. to 8:30 p.m., Eastern Time, Monday through Friday.

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 1-800-877-4787 or 1-512-345-5316.

End of Preface

Contents

Chapter 1 — The Big Picture: Managing DG/UX™ Systems

Chapter 2 — Installing the DG/UX System

Chapter 3 — Operating the DG/UX System

Chapter 4 — System Configuration Management

Chapter 5 — Release Management

Chapter 6 — Client Management

Chapter 7 — Disk Management

Chapter 8 — File System Management

Chapter 9 — File Information

Chapter 10 — TTY Management

Chapter 11 — LP System Management

Chapter 12 — UUCP Management

Chapter 13 — Network Management

Chapter 14 — User Account Management

Chapter 15 — Accounting Management

Appendix A — Device Names and Codes

Appendix B — Directories and Files

Appendix C — LP and UUCP Error and Status Messages

Appendix D — File System Checking: fsck

Appendix E — Using SCSI Devices

Appendix F — Installing the DG/UX System on a Previous Release

Appendix G — Loading the Release on a Tapeless Stand-alone Workstation

Index

Documentation Set

Tables

Figures

Chapter 2
Installing the DG/UX System

You should be reading this chapter after you have finished the procedures in the *Setting Up and Starting* manual for your hardware. You should have successfully powered up your equipment. Do not answer any prompts on your screen at this point. You will need to gather information about your system and make configuration decisions before you are ready to answer any prompts.

If you need to change any hardware parameters (such as mouse, modem, or console baud rate or console language), please do so before starting the procedures in this manual. Refer back to your *Setting Up and Starting* manual for this information.

This chapter focuses on the initial installation (and optional update) of the DG/UX system.

It presents a thorough discussion of the installation procedure for all configurations; for example, standalone system, OS client, and OS server, each with variable types of software packages and a variable number and type of devices. If you have this explicit configuration — standalone system with disk and tape units and the System Software Package for AViiON Systems or the Operating System Package for AViiON Systems — you may prefer to use a different Data General document, *A Quick Guide to Installing the DG/UX™ System on a Standalone AViiON™ Workstation*. It provides step-by-step installation procedures for this particular configuration.

If you are installing the DG/UX™ software on a system that is already running a previous release (for example, if you're currently running major release 4.20 and you want to install major release 4.30), refer to Appendix F.

If you are installing the DG/UX system on a workstation that does not have its own tape device, refer to Appendix G.

Installation Roadmap

Figure 2-1 presents the installation roadmap followed by a summary of the installation phases and steps.

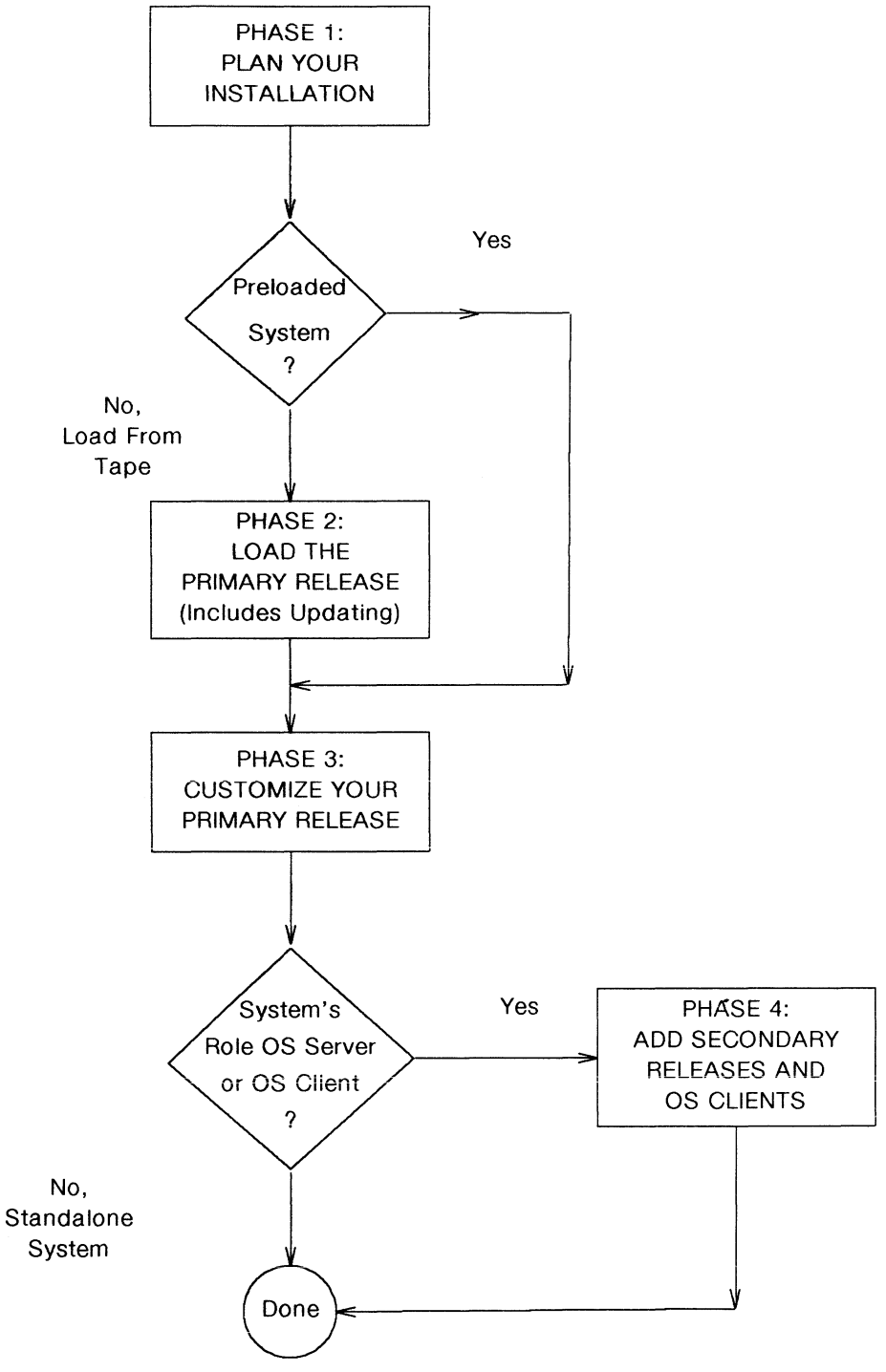


Figure 2-1 *DG/UX System Installation Roadmap*

Figure 2-2 shows the steps contained in each installation phase.

Phase 1: Planning the Installation

1. Determining How You Will Use Your System
2. Identifying Your System's Devices
3. Learning About Hosts, Software, Disks, and File Systems
4. Allocating Disk Space
5. Understanding the DG/UX directory tree
6. Assembling Network Information for Your System

Phase 2: Loading the Primary Release from Tape to Disk

7. Booting the Disk Management (diskman) Utility
8. Initializing Physical Disks With diskman
9. Creating System Logical Disks and File Systems
10. Loading DG/UX Files onto System Logical Disks
11. Updating the / and /usr File Systems

Phase 3: Customizing the Primary Release

12. Booting the Starter Kernel
13. Specifying Starter Devices
14. Creating Other Logical Disks and File Systems
15. Loading Software Packages With sysadm
16. Setting Up Software Packages With sysadm
17. Building a Custom Kernel
18. Setting Default Boot Characteristics
19. Starting System Administration

Phase 4: Adding OS Releases and Clients

20. Adding Secondary Releases
21. Building Kernels for Diskless Clients
22. Setting OS Client Defaults
23. Adding OS Clients
24. Booting and Setting Up an OS Client

Figure 2-2 *Outline of DG/UX System Installation*

Sample System Configuration Scenarios

Three configuration scenarios are given toward the end of the chapter (after Phase 4) as illustrations of how you can set up your system. The entire installation procedure is documented showing system/user dialogues for these scenarios:

- Standalone system
- OS server with two OS clients
- Standalone system supporting terminals

The example system may not be the same as your own system, so use these examples as only guides during your installation.

Using the Resource Planning Worksheets

Two resource planning worksheets are provided at the end of this chapter to help you prepare for installation. One worksheet is for the standalone system and OS server and the other is for an OS client. (Refer to Step 1, "Determining How You Will Use Your System" for definitions of these terms.) These worksheets list configuration questions for which you need to find answers (such as identifying your system's devices). Supplying these configuration answers during Phase 1 will speed up the installation process considerably.

If you have experience installing an operating system, you may prefer to skip over Phase 1 and go directly to the planning worksheets.

Phase 1: Planning the Installation

Planning the system involves:

- Determining How You Will Use Your System
- Identifying Your System's Devices
- Learning About Hosts, Software, Disks, and File Systems
- Allocating Disk Space
- Understanding the DG/UX Directory Tree
- Assembling Network Information for Your System

Step 1: Determining How You Will Use Your System

You can determine these things at this point:

- Role of your computer
- Type of DG/UX software package ordered
- Whether you will use the preloaded release on hard disk or you will load the software from tape

Role of Your Computer

How you install the DG/UX operating system depends on the role your computer plays. The roles are:

- Standalone system — any computer system that has its own disk containing the operating system that it uses, but which does not make an operating system image available to other systems. It may or may not be connected to a local area network; if it is not, it must have a tape device.
- OS Server system — any computer system that has its own disk containing a bootable operating system image and file system space that are provided to client systems on a local area network.
- OS Client system — any computer system that boots its operating system from an OS server system on a local area network.

Select a resource planning worksheet according to your configuration. If you have an OS server-client configuration, complete both worksheets. If you have a standalone system, complete the worksheet designed for both the OS server and standalone system. Record your computer's role as an OS client, OS server, or standalone system on the appropriate worksheet at the end of the chapter.

Determining the Type of DG/UX Software Package You Have

Determine the type of DG/UX software package ordered to help you answer configuration questions later. The name of the software package will be written on the label attached to the release tape. Table 2-1 lists the package names and constituent products.

Table 2-1 DG/UX Packages and Constituent Products

Package for AViiON™ System		
System Software (Client-Server)	**Operating System**	**Network Computing**
DG/UX™	DG/UX™	
GNU-C	GNU-C	
DG/UX™ DTK	DG/UX™ DTK	
DG/UX™ X Windows		DG/UX™ X Windows
Looking Glass®		Looking Glass®
DG/UX™ TCP/IP		DG/UX™ TCP/IP
ONC™/NFS®		ONC™/NFS®
OSF/Motif™		OSF/Motif™

Just because you have received a given software package doesn't mean that you will be using all its products. For example, you may have the System Software package for a standalone system in which case you may not need the networking software products (DG/UX TCP/IP and ONC/NFS). So, find out what products you really want to install on your system.

Record your software package type (and the products that you will be using) and your DG/UX system release level (and update level, if applicable) on your system on the planning worksheet at the end of the chapter.

In addition, if you have any third party packages (such as a database manager or a desktop publishing system), record them on the planning worksheet at the end of the chapter.

Using System Software That is Preloaded or on the Release Tape

All new AViiON systems come with both preloaded DG/UX system software and a separate release tape containing the same DG/UX system software. A preloaded system means that at the factory the software has already been loaded onto the hard disk. This precludes a need to load from tape to hard disk. However, you can install either form of the software. You will need to load from tape if you receive an update release for your previously installed system (for example, you are currently running 4.30 and you want to update to 4.31).

To have a preloaded system does not mean that your system is already installed. Rather, the contents of the release tape have already been placed on the hard disk. You will still have to set up the information on disk and perform other tasks that customize the software for your particular configuration.

Record whether you are using a preloaded system or you are loading from tape on the planning worksheet at the end of the chapter.

Step 2: Identifying Your System's Devices

A device loosely refers to hardware; for example, disk and tape storage devices, communications controllers, and input/output devices. You will need to identify your system's devices. You can find out about your devices from these places:

- Power-up display
- System diagnostics
- Packing slip

The following sections explain the information in these three areas.

After you identify your system's devices, record them on your planning worksheet at the end of the chapter.

Viewing the Power-up Display

You can learn some information about your devices from your computer power-up display. Turn computer power off, then back on to rerun the power-up test. An example of such a display follows:

```
(c) Data General Corporation 1989, 1990
Model 400/4000 Series
Dual Processor
Color Graphics [8 bit], Z-Buffer Option
Firmware Revision 5.02
Keyboard Language is U.S. English
Local Ethernet address is 08:00:1B:7F:7F:07
Initializing [16 Megabytes]

Testing.....
     0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
Passed
```

where:

Model 400/4000	Type of AViiON workstation
Color Graphics [8 bit], Z-Buffer	Workstation is equipped with graphics
U.S. English	Workstation's console display language
08:00:1B:7F:7F:07	Unique address that identifies a specific host.

The power-up display in this example indicates the presence of color graphics. Color graphics is a feature of the AViiON 400 series workstation rather than an AViiON 4000 series workstation. Therefore, in this example, you can derive that you have an AViiON 400 series workstation.

Record your AViiON Series computer type and local ethernet address on the planning worksheet at the end of the chapter.

What you see on your screen next depends on whether or not your disk device has been preloaded with the DG/UX software at the factory,

If your system was not preloaded (you will load from release tape), you will see this message followed by the display of the SCM prompt.

```
Unable to load bootfile sd(insc(),0)root:/dgux.starter

SCM>
```

The message indicates that the system attempted to boot from the default disk device, assuming that your disk was preloaded with the DG/UX software. However, since your disk is not preloaded, the attempt to boot failed. You will be loading from tape at some point.

CAUTION: Do not respond to the SCM prompt and do not press any key at this time. Your screen will go blank eventually. Information on answering the SCM prompt is given in Phase 2.

If you have a system that has been preloaded at the factory, the system will automatically boot the DG/UX starter system, showing the following completed display for the example computer:

```
Booting sd(insc(),0)root:dgux.starter

DG/UX Bootstrap Release 4.30.01

================================================================

DG/UX System Release 4.31, Version DG/UX 00 (starter)
Using [16] Megabytes of physical memory
Found 2 processor(s)
Processor 0 running
Processor 1 running

                DG/UX Starter System
Enter the names of the devices you will use in Common Device
Specification Format, with one name per line.  Enter just newline
when done.
Examples:  sd(insc(),0)  st(insc(),4)  cird()  st(cisc(),4)

Include duart() for servers and kbd() and grfx() for workstations.

Device Name?
```

Terms given in the preceding display are defined as follows.

Booting	Process of loading the DG/UX starter operating system from disk to computer memory.
Bootstrap release	The release level of the DG/UX bootstrap (program that loads the DG/UX system).
DG/UX starter system	The operating system that you will use as a base when customizing your system. Your system does not become customized until you identify your devices (such as disks, tape devices, parallel printers, and communications board) in a system file.
Common device specification	A naming convention used to identify the devices on your system.

Record the DG/UX system revision level (and update level, if applicable) on the planning worksheet at the end of the chapter.

Using the System Diagnostics

You received the AViiON System Diagnostics software (system diagnostics) with your computer system either preloaded on disk or on tape. The system diagnostics provide menu-based utilities to test any hardware in the AViiON product line. With one subtest (Acceptance test), you can verify your system's devices. A successful test displays the names of your system's devices.

Refer to *Using AViiON™ System Diagnostics* for information on running diagnostics.

Checking the Packing Slip

Refer to the packing slip to find out the model numbers of the equipment you ordered. Refer to the *Setting Up and Starting* manual for information on inventorying your equipment.

After you have identified your equipment by model number, you will need to do two things:

- For a disk or tape device, identify its model number. The device's size (or capacity) is determined by the model number.
- Derive the device specification from each device's model number.

The following sections explain how to determine this information.

Determining the Capacity of a Disk Device

You will need to know the capacity (or size) of your disk unit when you plan your installation. You can determine the size of a disk unit by its model number. Check your packing slip and your hardware manuals for information on device model numbers.

Table 2-2 lists the devices' model numbers and corresponding sizes (in MBytes and blocks).

Table 2-2 Disk Device Model Numbers and Corresponding Capacities

Disk Model Number	Approx Size (MBytes)	Approx Size (Blocks)
5070DR Optical Disk	2458	5033164
5070S Optical Disk	900	1843200
6442 ESDI (full-height)	327	669696
6491 SCSI (full-height)	322	659456
6539 SCSI (half-height)	179	366592
6541 SMD	1066	2183168
6542 SMD	2132	4366336
6554 SCSI (full-height)	662	1355776
6555 ESDI (full-height)	648	1327104
6627 CD ROM	590	1208329
6627 CD ROM	650	1331200
6661 ESDI (half-height)	330	675840
6662 SCSI (half-height)	322	659456
6685 SCSI (full-height)	1040	2129920
6740 SCSI (full-height)	1040	2129920

Note the disk device model number(s) and size(s) on the planning worksheet at the end of the chapter.

Specifying Devices

Each device must have a unique identifier so that the operating system can access it. Such an identifier is called a device specification. There are two types of device specifications: extended and simple. Examples of each type follow:

Extended	Simple
`cied@18(FFFFEE00,1)`	`cied(0,0)`

You must use the extended form if your configuration contains more than 2 SCSI adapter boards, 2 ESDI or SMD controller boards, 4 asynchronous terminal controller boards, or 2 synchronous terminal controller boards. (Consult your system administrator or the packing slip to find out what your configuration includes.) Refer to Appendix A, "Device Names and Codes," for information on using the extended form.

If your configuration contains the same number (or less) of boards that were previously mentioned, you can use the simple form, which is covered in this section.

You can supply simple device specifications for these basic devices:

- I/O devices
- Network devices
- SCSI tape and disk devices
- ESDI and SMD disk devices

Table 2-3 defines each device.

Table 2-3 Device Names

Device Name	Description
	I/O Devices
kbd	Keyboard.
grfx	Graphics monitor.
duart	Dual universal asynchronous receiver transmitter; a serial communications device.
syac, **sdcp**	Systech terminal controller boards: **syac** is asynchronous; **sdcp** is synchronous.
	Network Devices
hken, **inen**	Intelligent LAN (local area network) controllers. These provide a communications interface used by TCP/IP. The **hken**() device drives the Hawk LAN controller available for servers and some workstations. The **inen**() device drives the integrated LAN controller found on workstations and some servers.
	SCSI Tape and Disk Devices
sd	SCSI disk.
st	SCSI tape.
	ESDI and SMD Disk Devices
cied	Ciprico ESDI controller; can also be specified as **cird**() in the system configuration file (discussed later).
cimd	Ciprico SMD controller; can also be specified as **cird**() in the system configuration file (discussed later).
cird	Specifies all **cied**() and **cimd**() devices.

The syntax you use to specify a device is determined by the particular device type.

I/O and network devices:

dev-name ([*controller-num*])

SCSI tape or disk device:

dev-name (*adapter-type* ([*adapter-num*]) *unit-id*)

ESDI or SMD disk drive:

dev-name ([*controller-num*,] *unit-num*)

where:

dev-name = Specific device name (listed in Table 2-3).

controller-num = **0** (first) or **1** (second) board. For asynchronous terminal controller boards only, **0** (first), **1** (second), **2** (third) or **3** (fourth) board.

adapter-type = **insc** (Integrated SCSI adapter) or **cisc** (Ciprico SCSI adapter).

adapter-num = **0** (first) or **1** (second) board.

unit-id = unit identifier of disk or tape. For SCSI devices, the *unit-id* corresponds to the SCSI identifier and SCSI logical number of the device. Valid unit identifiers are **0**, **1**, **2**, **3**, **4**, **5**, and **6**. Alternatively, you can use an asterisk (*) to select all devices of a particular type.

unit-num = unit number of the device. Valid unit numbers are **0**, **1**, **2**, and **3**.

Examples:

Keyboard	**kbd()**
Second position on RS-232 interface	**duart(1)**
Integrated LAN	**inen()**
Tape device at SCSI id 4 on first integrated SCSI adapter	**st(insc(),4)**
Disk device at SCSI id 1 on second Ciprico SCSI adapter	**sd(cisc(1),1)**
First ESDI disk on first ESDI controller	**cied(0,0)** or **cied()**
Second SMD disk device on first SMD controller	**cimd(0,1)**
All SCSI disk devices on first integrated SCSI adapter	**sd(insc(),*)**

NOTE: If the value for *controller-num* or *adapter-num* is 0, you can express the value as null; for example, **inen()**. Do not space between the parentheses.

Step 3: Learning About Hosts, Software, Disks, and File Systems

If you have experience in system installations, you may skip over this step and proceed to Step 4, "Allocating Disk Space." Otherwise, you may want to familiarize yourself with this terminology.

host

A host refers to the AViiON series hardware you are using and the software running on the hardware. You assign a name to a host to uniquely identify it. When naming your system, select a host name that relates to the use or location of your system; for example, owner, group, project, or department. Mnemonic names are particularly helpful in networked environments where hosts may share file systems. Do not use the capitalized names **MY_HOST** or **PRIMARY**; these names are reserved by the system.

release

A release is a complete operating system. You will have at least one release, the DG/UX 4.30 system, which is the primary release. If you intend to support secondary releases for OS clients that will not run the DG/UX 4.30 system, you need to assign names to the releases. A release name must conform to DG/UX file naming conventions described in *Using the DG/UX™ System*. It is good practice to use names that identify the release specifically, as in this formula:

architecture_os_version

For example, you might call a DG/UX 4.20 system release **88k_dgux_420** because it is based on the Motorola 88000 chip architecture and the DG/UX 4.20 system.

package

A package is a set of files and/or programs that comprises a software application or other utility, such as the X Window System™, known as X11.

Examples of other packages are **nfs** (Network File System), **tcpip** (Transport Communications Protocol/Internet Protocol), and **yp** (Yellow Pages).

local software

For your site-specific shell scripts, programs, and so on, we suggest that you create a directory **/local** for programs and files that may change from system to system within your environment. We suggest that you use the directory **/usr/local** for site-specific programs and files that do not change from system to system. You should put these file systems on their own logical disks—not on the root logical disk.

block

A disk block is a unit of measure for the capacity of a disk or portion of a disk. It is equivalent to 512 bytes.

physical disk

A physical disk is the actual recording medium used for storing information. Disk space on the disk's surface is measured in

blocks. The amount of available disk space you have depends on the disk type. Table 2-2 lists disk model numbers and corresponding available disk space. You name a physical disk unit with a common device specification. Before you use a physical disk, you first divide it into logical disks.

logical disk

Abbreviated as LDU (logical disk unit), a logical disk is a fixed portion of the physical disk space. (Logical disks are sometimes called partitions.) You name a logical disk using the DG/UX file naming conventions described in *Using the DG/UX™ System*. A logical disk name must be unique within a system. The physical disk should be divided into exclusive logical disks that are reserved for specific data. For example, on a host named **revenue**, you might create a logical disk named **sales**. A logical disk can span different locations on a single physical disk or it can span several different physical locations on multiple physical disks. Each different physical location comprising a logical disk is called a piece. You use a logical disk as a virtual physical disk. Logical disks are measured in blocks.

Once you have divided the physical disk into logical disks, you effectively use logical disks as file systems that are restricted to that allocated space. There is one exception, however; a swap logical disk does not contain a file system. Information on a swap logical disk is given in Step 4.

Preloaded disks already contain logical disks that are reserved for the DG/UX operating system. If you load from tape, however, you will have to create these explicit logical disks that are used by the operating system. In addition, you will create logical disks for a variety of other uses.

file system

A file system is the organization of the space on a logical disk into a hierarchy of files. Only one file system can be created per logical disk. All the space allocated for the logical disk is made available to the files of the file system except for a small amount used by the operating system. For example, on a host named **revenue**, the logical disk named **sales** will contain a hierarchy of files you create that relate to sales accounts. Regardless of the spanning of logical disk pieces across physical disks, a file system will effectively operate as a logical unit. Figure 2-3 for the correspondence between physical disks, logical disks, and file systems.

mount point

A mount point is the directory location at which you place a file system. For example, for a file system created for user accounts on a logical disk named **sales**, you might mount it beneath root — **/sales**.

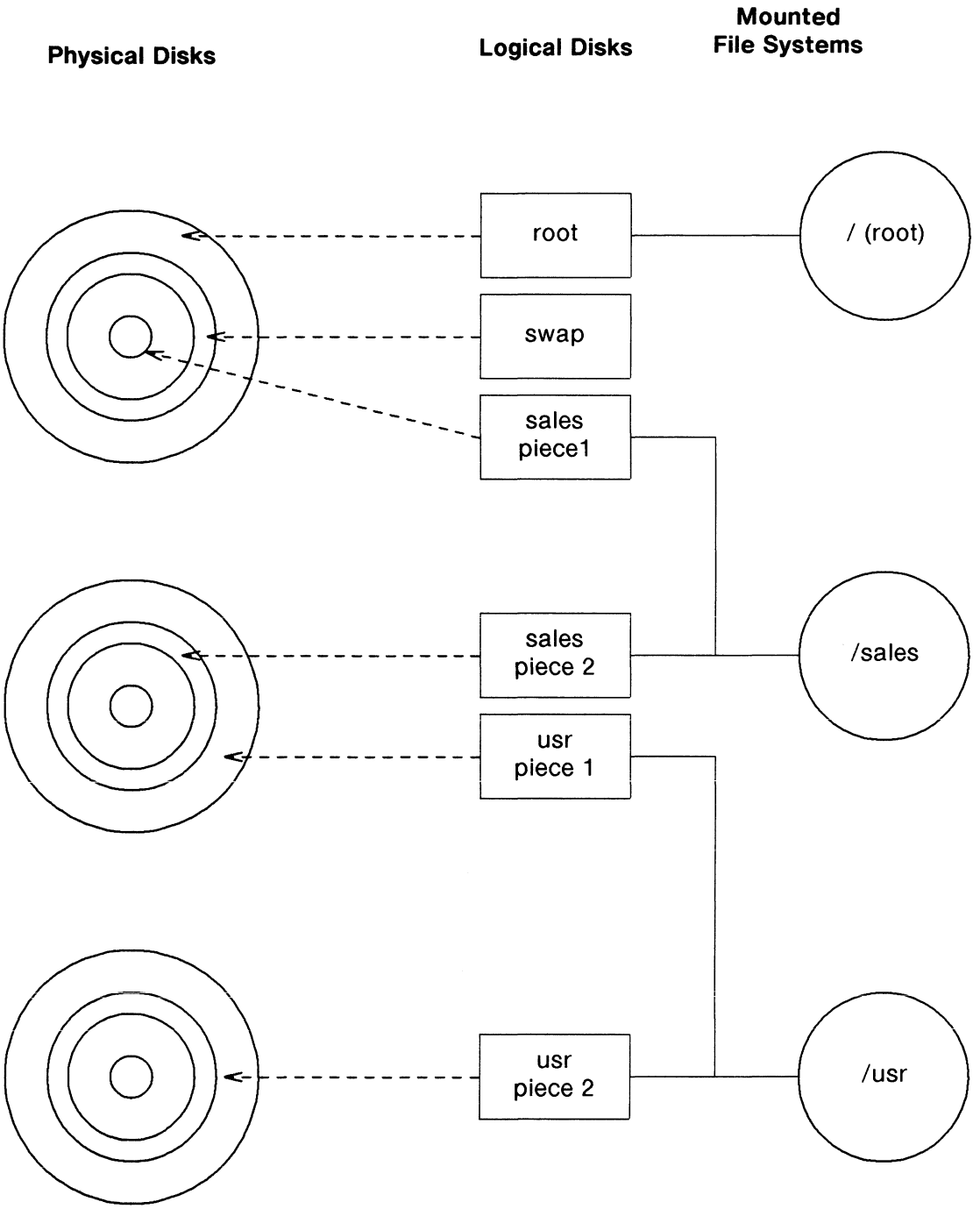


Figure 2-3 *Correspondence Among Physical Disks, Logical Disks, and File Systems*

Step 4: Allocating Disk Space

In Step 1, you identified the role of your computer; either a standalone system, an OS server, or an OS client. Also, you identified your DG/UX software package type and other optional third party packages. In Step 2, you identified your devices and you determined the physical size (maximum capacity) of your disk devices. You supplied these answers on the resource planning worksheet(s) at the end of the chapter.

Next, you need to answer these questions (and to record the answers on the planning worksheet).

- Besides my DG/UX software package and third party packages, what other software do I have? Or what other data will be stored on the disk(s)? (Read about packages in the next section within this step.)
- If my system is an OS server, how many OS clients are there? Will multiple software releases be supported?
- What logical disks are needed for my software?
- What size should each logical disk be?
- To improve performance, should I consider breaking logical disks into pieces across physical disks? If so, what size should each piece be and on what physical disk(s)?
- Where will the logical disks be mounted?

Read the following sections for help in answering these questions.

Logical Disk Types

There are several logical disk types, some of which are already preloaded and others that you have to create.

- System; may be preloaded
- Software packages; you create
- User home directories; you create
- Miscellaneous; you create
- OS Client; you create

NOTE: If your system is preloaded, the X Windows (X11) package will be included.

You name a logical disk using a combination of alphabetic characters, numbers, the period (.), the comma (,), and the underscore (_). You can use as many as 31 characters in a logical disk name.

The following sections address the different logical disks that you may want to create.

System Logical Disks

On systems preloaded at the factory, the system logical disks are created automatically for you. If you are loading from tape, you will have to create the system logical disks. They are:

- **root**
- **usr**
- **swap**
- if you have the X Windows package, **usr_opt_X11.**

Table 2-4 shows the default sizes and mount point directories for the **root**, **swap**, **usr**, and **usr_opt_X11** logical disks. Logical disks on preloaded disks reflect these default sizes (in blocks) and mount points.

Table 2-4 Default Sizes and Mount Points for DG/UX System Logical Disks

Logical Disk Unit	Size in 512-Byte Blocks	Mount Point Directory
root	40,000	**/**
usr	160,000	**/usr**
swap	50,000*	–
usr_opt_X11	105,000**	**/usr/opt/X11**

* 32,768 on preloaded 179MBytes disks.

** Only if you have the X11 package; includes the Looking Glass (X11.lg) package.

The default **usr** logical disk size will hold the DG/UX operating system, TCP/IP (10,000 blocks), and ONC/NFS (20,000 blocks). If you never intend to load TCP/IP and ONC/NFS, you can reduce the size of your **usr** logical disk. The **swap** logical disk does not need a corresponding file system, so has no mount point directory.

CAUTION: Do not place any site-specific or third party software on system logical disks. These disks are reserved exclusively for the system software. If you update your operating system at some future point, you will lose all files that you might have placed in the system logical disk space.

root Logical Disk

The root logical disk, whose file system is named **/**, is reserved for system-level programs and facilities. The **/** directory will be the mount point for subdirectories that will contain such things as data and configuration files, system commands, temporary system files, device nodes, spool files, and symbolic links to other subdirectories. If your system is not preloaded, you will create and load the appropriate file systems into the root logical disk in Phase 2.

Record the desired **root** logical disk size on the planning worksheet at the end of the chapter.

usr Logical Disk

The **usr** logical disk, whose file system is named **/usr**, is reserved again for system-level programs, facilities, and software packages. The **usr** directory will be the mount point for subdirectories that will contain such things as database and configuration files, administrator commands, site specific files, standalone utilities and bootstraps, and user commands. If you have a nonpreloaded system, you will load the appropriate file systems into the **usr** logical disk during Phase 2. Of particular note is the loading of software packages (such as X windows, known as **X11**) into a **/usr** subdirectory named **opt**.

Record the desired **usr** logical disk size on the planning worksheet at the end of the chapter.

Swap Space

Swap space is the temporary storage location of an active page from process on a logical 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. Swap logical disks differ from others in that they never need to be file systems. Each system needs at least one swap logical disk.

The amount of swap space that you will actually need depends on the amount of physical memory in your machine, the nature and number of the applications you intend to run, and the number of users on the system. If your programs will allocate large portions of memory, you may need more swap area. Insufficient swap area can result in the termination of running processes and errors such as "out of paging area" at the system console. If you encounter such an error, you will need to create more swap space. You can also create multiple swap logical disks to supplement existing swap logical space. See Chapter 8, "File System Management," for more information on adding more swap space.

The typical user can operate comfortably with swap space that is 1.5 times the size of the machine's physical memory. For a machine with 16MBytes of memory, this rule implies a swap space of 24MBytes. Given that a megabyte is 1,048,576 bytes, and a block is 512 bytes, convert 24MBytes to blocks like this:

```
(1.5 * (physical-memory * bytes-per-MBytes) )/ bytes-per-block = blocks
          in-Mbytes

(1.5 *  (16              *          1048576) )/ 512             = 49152
```

(or, more simply, 50,000)

As a result, you would specify 50000 blocks for the size of swap.

Record the desired **swap** logical disk size on the planning worksheet at the end of the chapter.

Applications Packages

For applications packages, you can use one logical disk to contain all packages, or you can create a separate logical disk for each package. To save disk space, consider making a logical disk for each package. By tailoring the size of a logical disk to fit its software package, you save disk space.

The DG/UX system loads some packages into directories in **/usr/opt**. Once installed, each package has its own directory there. If you choose to create logical disks to hold application packages, you should mount them on appropriately-named directories under **/usr/opt**. The X11 package, for example, resides in **/usr/opt/X11**. Before loading X11, you should create a logical disk for it, named **usr_opt_X11** and 105,000 blocks in size, and mount it on the directory **/usr/opt/X11**. The release notices for your applications packages should tell you what you need to know:

- How much disk space the package needs, and
- The directory where the loaded package will reside.

After creating the needed logical disks and mounting them in the desired locations, you are ready to load the packages. Later instructions lead you through the necessary steps.

There are three packages that do not reside under **/usr/opt**; they are TCP/IP, NFS, and YP. These packages load into other parts of the **/usr** and **/** (root) file systems.

Record the desired application package logical disk sizes on the planning worksheet at the end of the chapter.

Home Directories

A home directory is useful for containing each individual's work on the system. You should create one logical disk (named **accounts** as an example) to accommodate all users' home directories.

A user's home directory will require a variable amount of space. Space concerns will depend on these questions: will a user produce (and accumulate) programs? academic papers? database reports? data files? In environments where most activity

occurs in a work directory rather than a home directory, a user may not need much home directory space. Remember to account for any OS clients' home directories as well as directories for local users.

As an example, users are working on documentation projects of varying sizes which are all located in one central project directory. Each user's home directory is needed primarily for containing subdirectories to save mail, write memos, collect product specifications, and produce scratch files.

You might calculate the size of a single logical disk for multiple user accounts as follows:

```
blocks_per_user * number_users = logical_disk_size

    40,000      *      5       =     200,000
```

Phase 3, Step 19 contains instructions for adding user accounts.

Record the desired home directories' logical disk sizes on the planning worksheet at the end of the chapter.

Miscellaneous

You will have to create miscellaneous logical disks whose sizes depend entirely on your plans for your system. Examples of customized logical disks are:

- Work directories
- Local tools packages
- Temporary file directories

Record the desired logical disk sizes on the planning worksheet at the end of the chapter.

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 areas would be too large for a single physical disk unit, or if you suspect that disk I/O performance could become a bottleneck during multiuser access, you may consider breaking up large work areas and distributing them across multiple physical disks.

Local Tools Packages

Local tools packages are another candidate for a customized logical disk. An appropriately-named tools directory is easily recognizable and accessible to users on your network. 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

User programs use temporary space when they start and as they execute. Large program compilations, heavy network traffic, and large database I/O activities access require temporary file space.

To segregate temporary file space from the **root** directory, you may want to create an explicit logical disk for temporary file space and mount it on the **/var/tmp** or **/tmp** directories. By default, 40,000 blocks are allocated to the **root** logical disk for its file system. After the **root** file system is loaded, 12MBytes will remain as free space which can be used for **/var** and **/tmp**. All subdirectories of **/var** will use the same space. In addition, you may want to create separate logical disks for the **mail** and **news** subdirectories of **/var**.

OS Client Logical Disks

NOTE: This section applies to OS client-server configurations only. If you do not have this configuration, skip over it, and proceed to Step 5.

You need to be concerned about OS client logical disks only if your system will be an OS server. You will create them (whether or not your system is preloaded) for each of the following:

- **/srv** directory
- OS Client root space
- OS Client swap space
- OS Client dump space
- Each secondary release; optional if secondary releases are supported

You create one of each of these logical disks on the OS server.

Record the desired OS-client logical disk sizes on the planning worksheet at the end of the chapter.

/srv directory

The **/srv** logical disk holds file systems for OS clients such as root directories and swap space. In Phase 4, you will add an OS client with the command, **sysadm addclient**. You will mount its file systems in the **/srv** directory tree with the command **sysadm addfsys**.

You should create a single logical disk named **/srv** with a size of 5000 blocks.

OS Client Root Space

The OS client root space is a single logical disk that contains all the root directories for all clients. The root directory will contain subdirectories that correspond to each client. In Phase 4, you will add an OS client with the command **sysadm addclient**. You will mount its file systems in the **/srv** directory tree with the command **sysadm addfsys**.

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.

You would calculate the size of the logical disk like this:

```
blocks_per_client * number_clients  = logical_disk_size

    40,000        *        5        =     191,692.80
```

```
(or 192,000)
```

The command that you use to build kernels for OS clients allows you to link all OS clients to the same kernel image. The benefit is a savings in disk space. For OS clients to share a kernel, however, their root directories (and the directory containing the kernel) must all be on the same logical disk. For this reason, you may not want to distribute OS client root directories over different logical disks. Sharing kernels can, however, result in weakened security because any user can access and change the kernel image. See Step 21 for more information on kernel sharing.

OS Client Swap Space

Refer to the discussion of swap space in the preceding section on "System Logical Disks." Whatever you decide for the swap needs of your OS clients, you should create a single logical disk for all clients' swap areas. You need to create a file system on the client swap logical disk so that clients can mount the file system across the network. Mount the client swap file system at **/srv/swap**.

Once you have decided how much swap space each of your OS clients needs, add another 17% of this total area (for file system overhead) to arrive at the size for the total client swap logical disk. If, for instance, you wanted to provide 50,000 blocks of swap space for five clients, you would calculate the size of the logical disk like this:

```
blocks_per_client * number_clients * 1.17 = logical_disk_size

    32,768        *        5       * 1.17 =     191,692.80
```

Later, when you add an OS client with the command **sysadm addclient**, the swap space you allocated when creating the **/srv/swap** logical disk will be apportioned to the OS client.

OS Client Dump Space

You need to allocate space for OS client system dumps. When a system panics or hangs, you have the option of taking a system dump, which means writing the contents of the system's memory to a file or tape so that Data General system engineers can diagnose the problem.

On a system with a tape device, you configure your kernel so that it dumps to tape. On an OS client without a tape device, you configure the system so it dumps to your network controller (that is, over the net to a file on the server). In Phase 3, Step 17, "Building a Custom Kernel," you will set the system DUMP parameter to the appropriate value for your system.

The **/etc/bootparams** entry that the **addclient** command creates for an OS client determines where the client's system dump goes. Typically, the file is **/srv/dump/***client_name*, where *client_name* is the client's host name.

You need to decide how much space the **/srv/dump** directory needs. This directory does not need to be its own logical disk or file system, but, due to the size of a system memory dump, you need to make sure the **/srv/dump** directory has room to grow as needed. If you don't create a **dump** logical disk, ensure that **/srv** is sufficiently large to accommodate a dump.

The maximum amount of space you would need for **/srv/dump** is equal to the total size of physical memory on all of your OS client systems. In practice, however, you need less space than this for two reasons:

- OS Clients will not need to make systems dumps all at the same time.
- You will not keep system dump files online for very long. It is probably sufficient to have enough space for one or two system dumps.

If each OS client has no more than 16 megabytes (16,777,216 bytes or 32,768 blocks) of physical memory, then 33,000 blocks is sufficient to hold one system dump.

You would calculate the size of the logical disk like this:

```
blocks_per_client + overhead  = logical_disk_size

    33,000        +   17%     =   38,600
```

Secondary Releases

To support secondary releases, you need to create not only the necessary secondary root logical disks, but also any other file systems that the secondary release requires. If you wanted to retain the 4.20 release of the DG/UX system as a secondary release, for example, you would have to create logical disks for the 4.20 **usr** and **root** disks. You would, of course, want to give the logical disks names like **usr_420** and **root_420** to distinguish them from the primary (DG/UX 4.30) release logical disks. The **root_420** logical disk would have to be large enough to contain separate root space for each OS client of the secondary release. You do not need to create secondary swap areas for OS clients of secondary releases.

For specific information on foreign releases (releases other than the DG/UX system), see the foreign system's documentation.

Step 5: Understanding the DG/UX Directory Tree

This section shows an overview of the DG/UX directory tree. In addition, it shows the **root** and **usr** logical disks (that you will create in Step 9 and load with the appropriate file systems in Phase 2, Step 10). If your computer is an OS client-server, you will also create the **srv** logical disk and load with the appropriate file systems (Phase 3, Step 14).

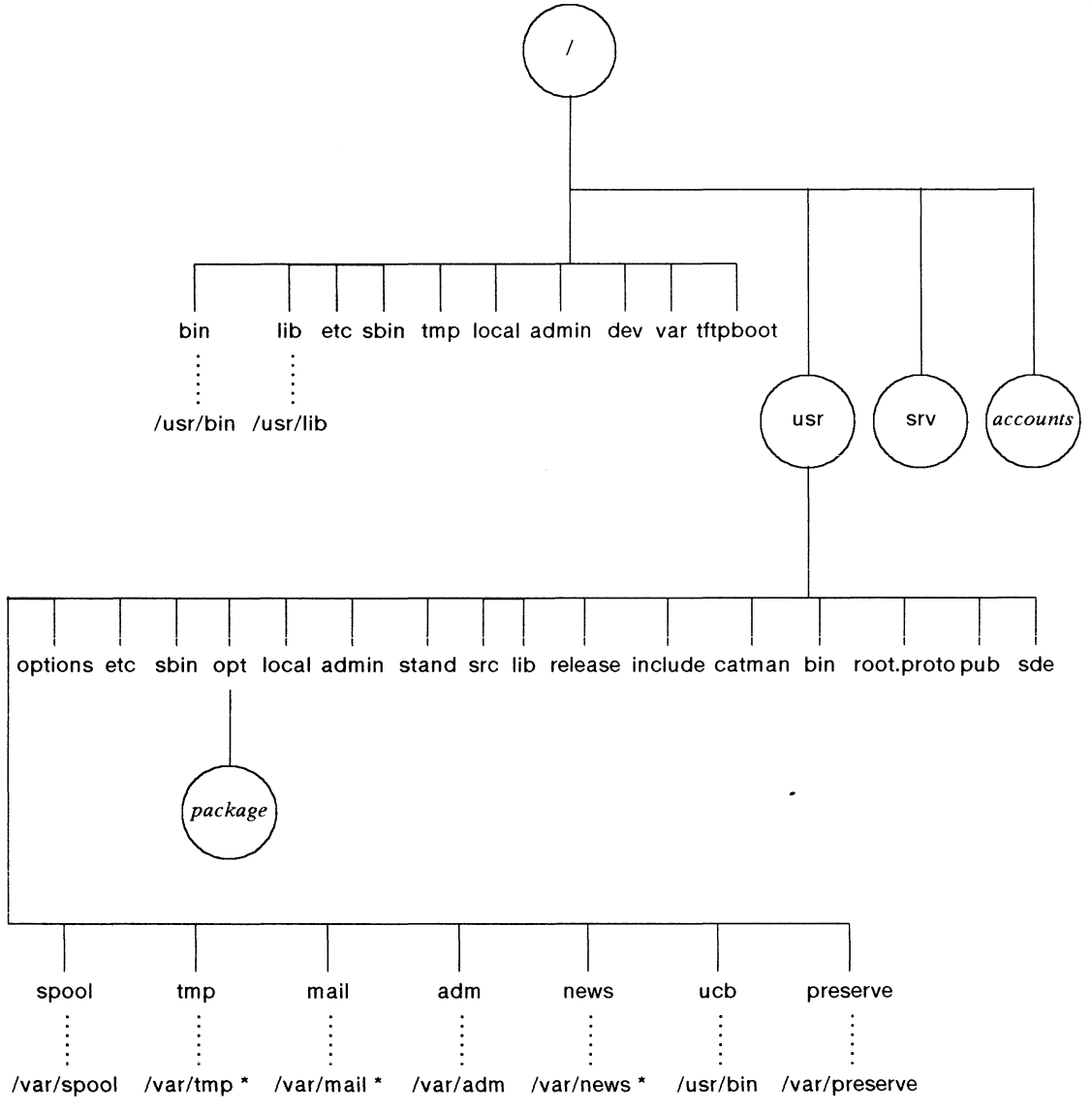


Figure 2-4 *An Example DG/UX Directory Tree*

Circles represent the mount point directories for constituent subdirectories.

accounts is a logical disk (which you would have to create yourself) intended to contain users' personal directories.

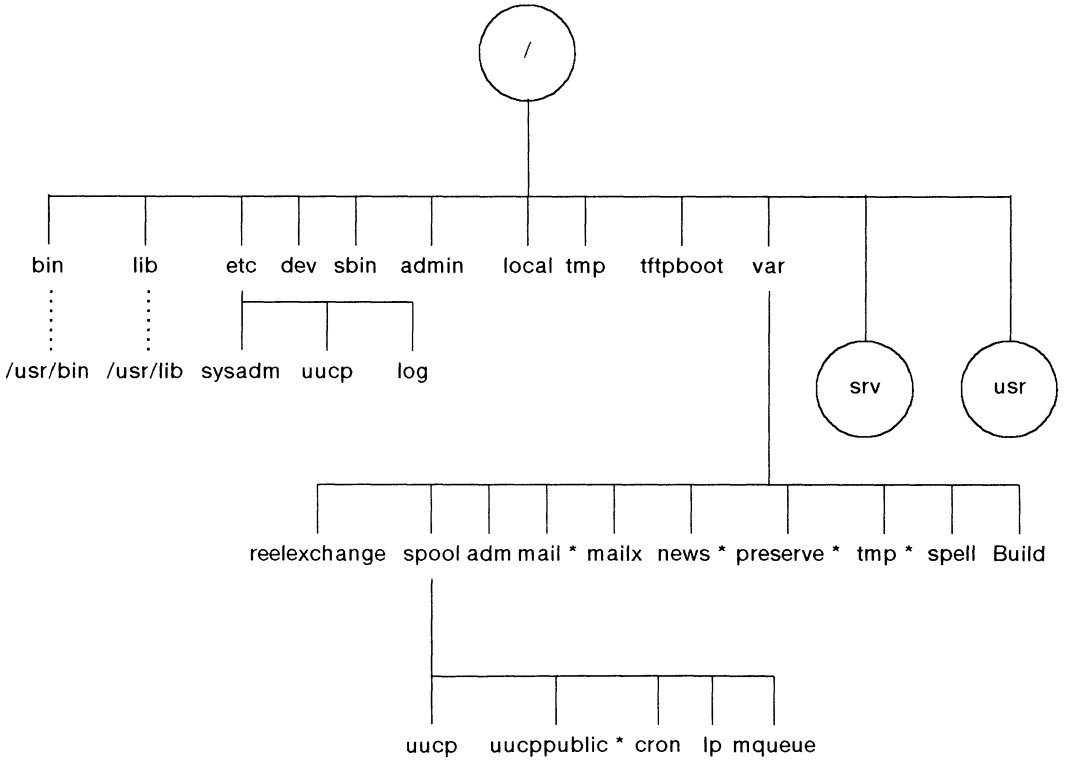
package is the mount point for the logical disk for a package; for example, **usr_opt_X11**.

The ... (dot) notation indicates a symbolic link to the named directory.

The * (asterisk) indicates that you may want to create a separate logical disk for the named directory.

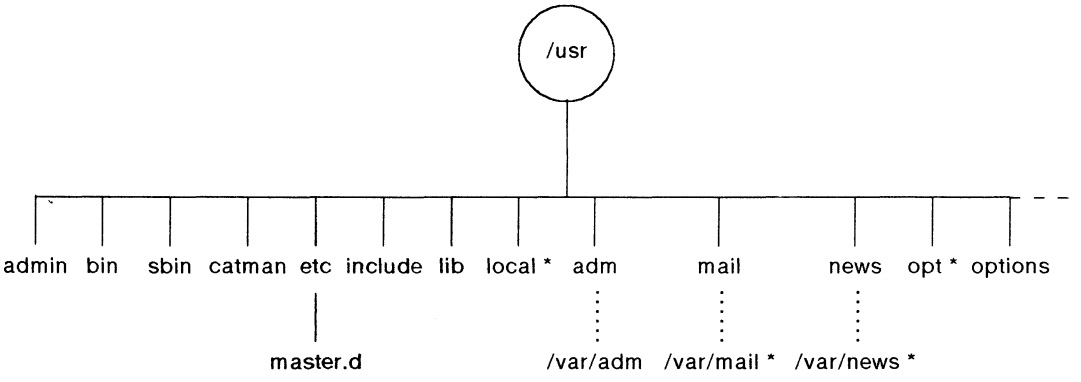
The / Directory Tree

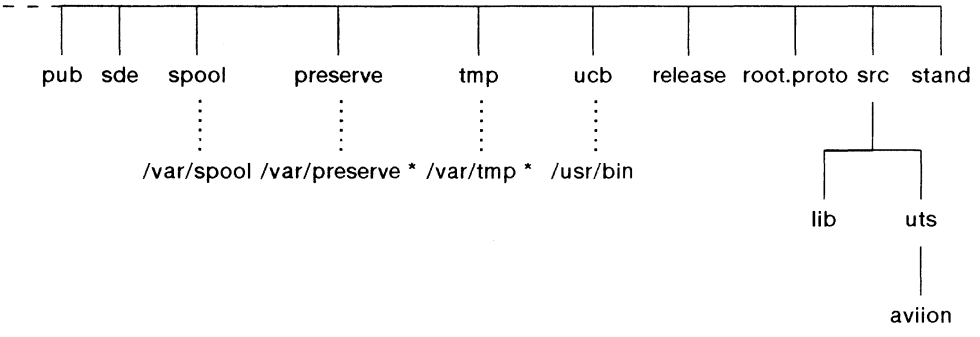
Here is the **/** file system. Commands traditionally found in **/bin** have been moved to **/usr/bin**. Note that **bin**, **dev**, **etc**, **lib**, **tmp**, and **sbin** must remain on the **/** logical disk. Do not move them elsewhere or use them as mount points. Depending on your usage, you may want to consider making logical disks for the directories marked with asterisks (*).



The /usr Directory Tree

The file system mounted on **/usr** contains architecture-dependent and read-only, shareable files. Directories containing writable, host-dependent files are indicated as symbolic links below.





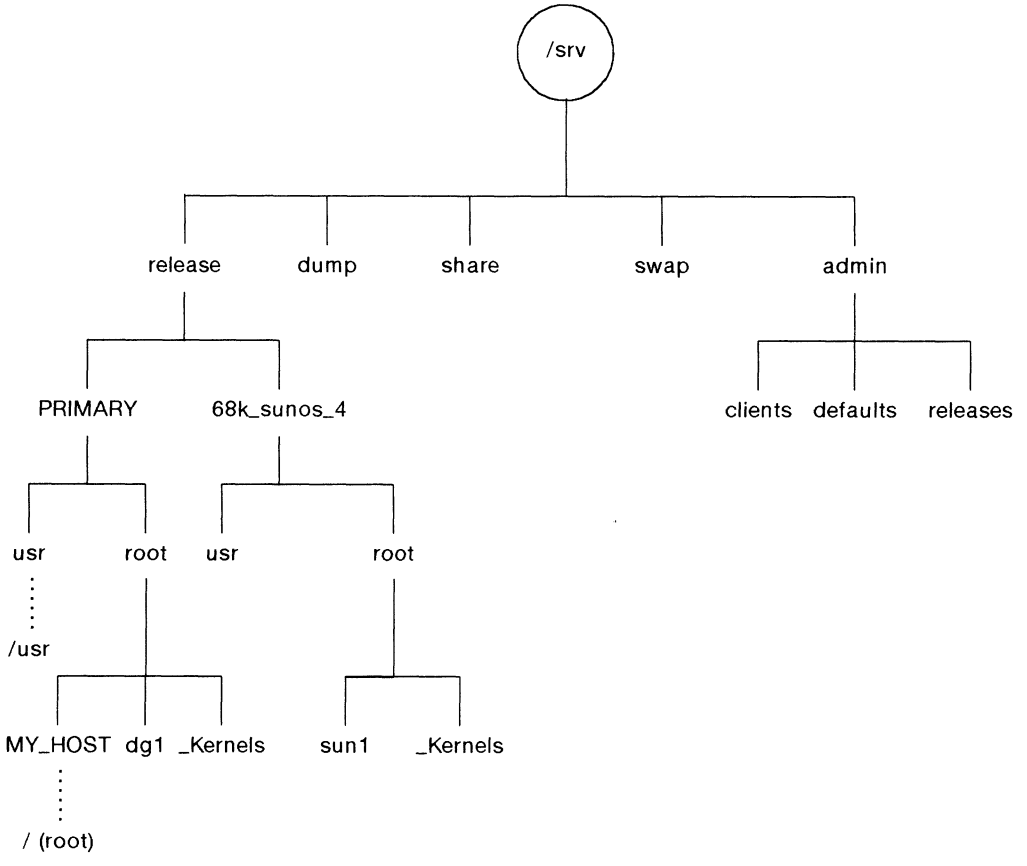
NOTE: The --- notation indicates a continuation of the first diagram to the second diagram.

The **/var/spool** directory replaces the traditional DG/UX **/usr/spool** directory. Files that are host dependent and change size dynamically (print and mail queues, log files, and so on) are located in **/var**. Symbolic links preserve the traditional directory tree structure of **/usr**.

The /srv Directory Tree

NOTE: This section applies to the OS client-server configuration only. If you do not have this configuration, skip over it.

The **/srv** directory tree holds file systems for OS clients. The example **/srv** tree below includes root directories and swap files for a Data General OS client, **dg1**, and a SunOS OS client, **sun1**. Except for these client-specific entries (and the **68k_sunos_4** release directory), the directories and files shown are standard components of the **/srv** tree.



The primary release, PRIMARY, has links to the server's **root** and **usr** logical disks. Having a **/srv** directory allows the **sysadm** program to manage releases and clients regardless of which DG/UX release the server is running. You should use the **sysadm** utility to create and maintain the **/srv** directory structure. Refer to Chapter 6, "Client Management" for more information.

File System Mapping: the DG/UX System to a Foreign System

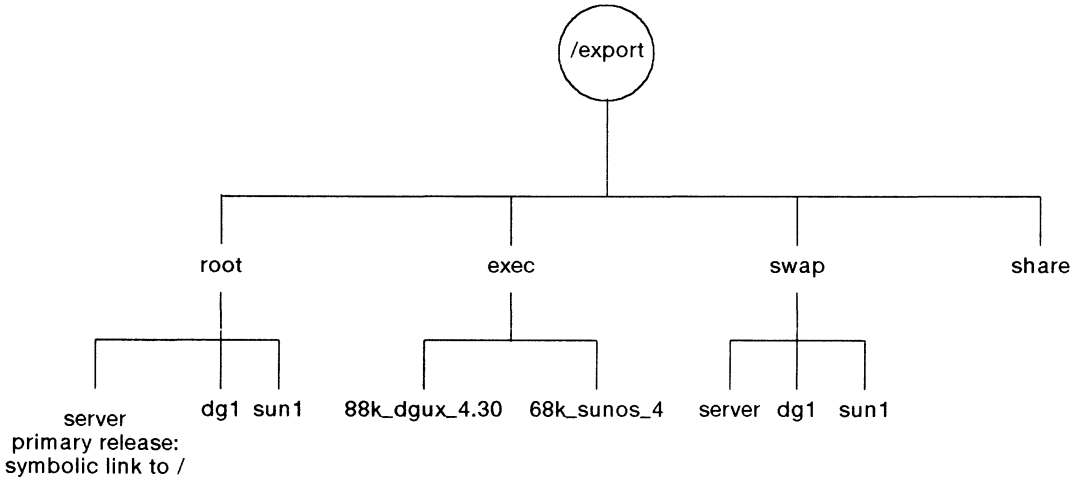
If you want to take advantage of **sysadm** to manage foreign clients, you need to set up a **/srv** structure for them. Below we compare Data General's **srv** tree to a foreign system's comparable structure. The **/**, **/usr**, and **/var** directories are basically the same, but there are major differences between **/srv** and the foreign structure. Refer to the preceding directory tree for the **/srv** structure.

A foreign system **/export** structure appears as follows:

On both structures, you'll see one Data General client (**dg1**) and one foreign client (**sun1**). There are two advantages of Data General's **srv** structure:

- Each client may be attached to multiple releases while maintaining a separate root for each release. This way, clients can choose among releases they wish to boot.

- The **srv** file system contains data files used to maintain clients and releases which are not associated with a particular root on the server. If the server boots another release, it won't lose information about clients and releases.



Step 6: Assembling Network Information for Your System

If your system will not be part of a network, you may skip this step. If you ordered the System Software Package for AViiON System or the Network Computing Package for AViiON System, follow the instructions in this section.

If you intend to install the TCP/IP, ONC/NFS, and/or YP packages, there are a number of things you need to know. If you are installing the system in an established network, confer with the network administrator first. For complete information, see:

- Chapter 13, "Network Management," in this manual
- *Setting Up and Managing TCP/IP on the DG/UX™ System*
- *Managing NFS and Its Facilities on the DG/UX™ System*

After reading the following sections, record all information on the planning worksheet at the end of the chapter.

Compiling TCP/IP Information

Before setting up the TCP/IP package, you need to gather a variety of information about your system and local network. See your network administrator or *Setting Up and Managing TCP/IP on the DG/UX™ System*. You need to know:

internet address	During network installation, you need to know the internet address of your own system as well those of other systems on your network. An example internet address is **128.223.2.1**, where: **128.223** refers to the network number, **2** refers to the subnet number, **1** refers to the host number. The dots are field separators.
host name	This name could be whatever you intend to call your system within your network. Step 3 discusses host names.
network name	This is the name of your local network. An example is **sales-net**.
subnet status	You need to know if your local network is subnetted.
network mask	The network mask you use depends on how your local network is subnetted. An example mask is **0xffffff00**.

controller device type	On some workstations and some servers, your controller device type is **inen**. For servers and workstations that have a Hawk LAN controller, it is **hken**.
controller device name	Your controller device name is the same as the type but with a **0** or **1** appended: **hken0** or **inen0**. AViiON 400- and 4000-series systems may have one Hawk LAN controller (**hken0**) in addition to the integrated controller (**inen0**). AViiON 5000- and 6000-series systems may have as many as two Hawk LAN controllers (**hken0** and **hken1**).
broadcast address type	The broadcast address may be either all zeroes (BSD 4.2 compatible) or all ones (BSD 4.3 compatible).
other systems	You may also want to have on hand the host names, Internet addresses, and Ethernet addresses of other systems in your network so that you may add entries for them to your **/etc/hosts** and **/etc/ethers** files during TCP/IP setup.

Compiling ONC/NFS Information

See your network administrator and *Managing NFS and Its Facilities on the DG/UX™ System* for more information on the ONC/NFS package.

Compiling YP Information

Before setting up the YP package, you need to decide if your system will function as a YP master, YP server, or YP client. You need to know the name of your system's YP domain. See your network administrator and *Managing NFS and Its Facilities on the DG/UX™ System* for more information on the YP package.

Compiling OS Client Information

If you are installing an OS server, you also need to decide what you need to name your OS client and what OS releases you want them to use.

If you have a preloaded system, skip Phase 2 and go to Phase 3. If you are loading from tape, go to Phase 2 for instructions on loading the primary release.

End of Phase 1

Phase 2: Loading the Primary Release from Tape to Disk

After you have completed all the instructions in Phase 1, and you are loading from tape, you are ready to start the procedures in Phase 2. If you have a preloaded disk, skip this phase and proceed to Phase 3, "Customizing the Primary Release."

The steps in this phase are:

- Booting the Disk Management Utility (**diskman**)
- Initializing Physical Disks with **diskman**
- Creating System Logical Disks and File Systems
- Loading DG/UX Files onto System Logical Disks
- Updating **/** and **/usr** File Systems

Step 7: Booting the Disk Management (diskman) Utility

Booting means to load an image from the boot device (disk or tape) into memory and then to execute the memory image. You will be booting the **diskman** utility from your release tape. The **diskman** utility enables you to create logical disks on physical disks and to load file systems from tape onto these logical disks.

To perform this step, you need to insert the tape into the boot device drive.

If you are updating your system (you received an update tape in addition to the major release tape), insert the update tape in the drive now.

If you are performing an initial installation only (you received a major release tape only), insert the major release tape in the drive now.

Before you use the **diskman** utility, you need to know the name of the boot (tape) device. Check your planning worksheet.

For AViiON workstations and AViiON 3000 and 4000 series systems, type:

```
SCM> b st(insc(),4) ↵
```

For AViiON 5000 and 6000 series systems, type:

```
SCM> b st(cisc(),4) ↵
```

where:

b stands for boot.

st(insc(),4) is the device identifier for the first tape drive on an integrated SCSI controller.

st(cisc(),4) is the device identifier for the first tape drive on a Ciprico SCSI controller.

You will see brief instructions on the screen for answering the prompts for device names and then the first prompt. Your entry (or entries) depends on the type of system console you are using.

If your keyboard is connected to the terminal (not to the computer itself, as with most workstations), you should specify **duart()**:

```
Device name? duart() ↵
```

If you are using a graphics monitor (where your keyboard is connected to the workstation), enter **kbd()** followed by **grfx()**:

```
Device name? kbd() ↵
Device name? grfx() ↵
```

If you make a typing mistake, press New Line and you will receive an error message followed by another device prompt. Then you can key in the correct answer.

After naming your starter devices, press New Line again to indicate that you are finished. The **diskman** utility then begins running.

The Diskman Main Menu

Table 2-5 lists the commands on using the **diskman** menus.

Table 2-5 Using Diskman Menus

User Input	Description
^	Return to previous menu.
?	Print HELP message, then redisplay menu.
number	Choose menu item by entering number.
number?	Give information on item number specified.
q	Exit **diskman**. Enter from any menu.
New Line	Select default response.

After the **diskman** utility is loaded, you will see the opening menu:

```
            Diskman Main Menu

1. Physical Disk Management Menu

2. Logical Disk Management Menu

3. File System Management Menu

4. Initial Installation Menu

5. Update Installation Menu

Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT.

Enter Choice: 4
```

If you want to perform an update only (you received an update release tape only), your goal is to update your installation. Choose option 5, "Update Installation Menu" now and proceed to Step 11 for instructions.

If you want to perform an initial installation (you received the major release tape and optionally, the update release tape), your goal is to perform initial installation. You will want to choose option 4 eventually.

However, before you make this selection, as an option, you can check the total disk space available for creating logical disks.

Checking Physical Disk Size (Option 1)

This optional step is useful for checking total physical disk space, the space consumed by the creation of logical disks, and the remaining unallocated disk space.

To check physical disk size, from the Diskman Main Menu:

- Select option 1 (Physical Disk Management Menu).
- Select option 3 (Display a Physical Disk's Layout).
- Provide the device specification for the device you are checking:

You will need to know the device's name before you can answer the prompt.

For AViiON workstations and AViiON 3000 and 4000 series systems, an example follows:

sd(insc(),0) ↵

For AViiON 5000 and 6000 series systems, an example follows:

sd(cisc(),0) ↵

where:

sd(insc(),0) is the device specification for the first SCSI disk on an integrated SCSI controller.

sd(cisc(),0) is the device specification for the first SCSI disk on a Ciprico SCSI controller.

This screen contains statistics about the physical disk. Read the screen carefully. The total physical disk size should approximate the physical disk size that you recorded on your planning worksheet at the end of the chapter.

Using the caret (^) will return control to the Diskman Main Menu.

Initial Installation Menu (Option 4)

From the Diskman Main Menu, after you select the fourth option, Initial Installation Menu, the default, you will see the Initial Installation Menu.

```
            Initial Installation Menu


1. Initialize Physical Disks

2. Create the Root Logical Disk and File System

3. Create the Swap Logical Disk

4. Create the /usr Logical Disk and File System

5. Load the Root File System

6. Load the /usr File System

7. All Installation Steps

Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT.

Enter Choice: 7
```

You need to perform all the steps in the order they are listed, so press New Line to accept the default choice (7). The **diskman** utility will automatically take you through the six steps, beginning with number 1, initializing physical disks.

Step 8: Initializing Physical Disks with diskman

In this step, the **diskman** utility will format the physical disks and install bootstraps.

After you choose number 7, the **diskman** utility leads you through a dialogue to initialize your physical disks. Throughout this dialogue, example user responses are given.

```
1. Initialize Physical Disks

Do you want to run this step? [y] ↵
```

Pressing New Line indicates that you want to initialize the physical disk.

```
Enter the Physical Disk specification in DG/UX common
format: sd(insc(),0) ↵
```

Type the appropriate device specification. Refer to your planning worksheet for this specifier.

```
Install a Disk Label on a Physical Disk
Do you want to run this step? [y] ↵
```

Pressing New Line indicates that you want to install a disk label on the disk.

```
Disk label already exists on disk insc(0,0).
Do you want to reinstall disk label? [n] y↵
```

NOTE: If you are initializing a SCSI device (its name begins with **sd**), you will not see the preceding message on the screen.

If you enter a specification that begins with **cied**, **cimd**, or **cisc**, entering **y** and pressing New Line produces this display:

```
   Disk Types

1. 6442    ESDI          322MB
2. 6555    ESDI          648MB
3. 6661    ESDI          322MB
4. 6541    SMD          1066MB
6. None of the Above.
Enter the type of disk you have:
```

You select the device by entering the number that corresponds to the model of disk that you have and pressing New Line. Refer to the planning worksheet for the applicable disk model number of your disk drive.

If you enter a specification that begins with **sd**, a list of disk types is not displayed.

The following message appears:

```
Disk label has been installed.

Performing Hardware Formatting on a Physical Disk
Do you want to run this step? [y] ↵
```

After being notified that the disk label was installed, you can press New Line to accept the default (y) response.

Next, the **diskman** utility displays the following prompt:

```
Create DG/UX System Areas on a Physical Disk
Do you want to run this step? [y] ↵
```

System areas contain the initial bootstrap program and information describing the layout of the physical disk. Pressing New Line indicates that you want the **diskman** utility to create areas on the disk for the DG/UX system. You then see a message warning that if you proceed, any data on the disk will be destroyed. For initial installation, there is nothing on the disk, so there is no fear of destroying existing data. The **diskman** utility prompts you to confirm your decision:

```
Do you want to continue? [y] ↵
```

Pressing New Line continues the formatting process.

A prompt similar to the following then appears on the screen:

```
The Physical Disk sd(insc(),0) is 663476 blocks in size.
Enter the number of blocks to allocate for the Remap Area: [189] ↵
```

The first line shows the device specification and block size for the disk you supplied earlier. In addition, the **diskman** utility calculates the number of blocks to allocate for the remap area, and displays that number as the default.

Pressing New Line accepts the supplied number of blocks.

If you choose to specify a different number of blocks to allocate for the remap area, it is better to enter a larger number rather than a smaller one. If you enter a smaller number, it is possible that the bad block table will fill up, and you will no longer be able to use the disk.

The following prompt then appears:

```
Enter the pathname of the boot.aviion file: [/usr/stand/boot.aviion] ↵
```

Since you are performing initial installation, pressing New Line accepts the default boot path, which locates the bootstrap file in memory. If you are installing a new revision of the software and you specified your own boot path in the previous revision, you can substitute your boot path for the default.

The next prompt appears:

```
Perform Surface Analysis on a Physical Disk
Do you want to run this step? [y] n ↵
```

Surface analysis builds a **bad block** table that specifies areas of the disk where the system should not store **data. You** may want to perform surface analysis to ensure that your entire disk surface is **unflawed**, and thus, useable. This procedure is time consuming. The **diskman utility runs** all of its test patterns on the disk. The process takes about 20 minutes per 100MBytes, depending on your physical disk model and CPU. An average is about one hour per physical disk. During testing, the **diskman** utility writes a status message to your screen every five minutes. At the conclusion of surface analysis, the number of bad blocks is reported and those blocks are rendered unuseable.

Data General disk devices do not require surface analysis. Therefore, to save time, you may opt not to perform surface analysis.

If you answer **n** and press New Line, the following prompt then appears:

```
Do you want to format another Physical Disk? [n] ↵
```

Pressing New Line accepts the default.

If you have another physical disk that you want to format at this time, type **y** instead and press New Line. The program for initializing physical disks will repeat from the beginning.

The **diskman** utility then begins the procedure for creating system logical disks and file systems, which is described next.

Step 9: Creating System Logical Disks and File Systems

Your goal is to create three system logical disks and to prepare them to contain file systems. In this context, creating a file system means that an empty file system will be created for the eventual loading of files, which occurs in Step 10. The three system logical disks are listed:

- **root**
- **usr**
- **swap**

Creating the Root Logical Disk and File System

The dialogue begins as follows:

```
      2. Create the Root Logical Disk and File System

Do you want to run this step? [y] ↵
Enter the Logical Disk Name: [root] ↵
```

Pressing New Line accepts the default logical disk name (**root**).

The next prompt is displayed:

```
Enter the Physical Disk specification in DG/UX common
format: [sd(insc(),0)] ↵
The physical disk must be registered for this operation.
Do you want to register it? [y] ↵
Physical disk sd(insc(),0) has been registered.
Do you want to display the layout of this Physical Disk? [n]  ↵
```

To expedite this step, you can accept the default, which is no. However, following each step in which you create a logical disk, you may want to check the remaining unallocated physical disk space. Answering **y** will allow you to review your disk's layout in the "Display a Physical Disk's Layout" screen before continuing to the next prompt, which is displayed as follows:

```
Enter the Physical Disk Address of the starting block
of the Logical Disk Piece:  [default]  ↵
```

Accepting the default selects a location, which begins at the first available space on the disk.

```
Enter the size in blocks of the Logical Disk Piece: [40000] ↵
```

Consult your planning worksheet for the logical disk size. When in doubt about logical disk sizes, choose the default value supplied by the program. These values are based on experience with the system.

The following message is displayed:

```
The Logical Disk "root' has been created.

Making a file system on the logical disk "root' ...

Made a file system on the Logical Disk "root'.
```

The root logical disk and an empty file system have been created.

Creating the Swap Logical Disk

Swap space is the temporary storage location of an active page from a process on a logical disk. A page is swapped out (or paged) to swap space when there are more active processes than can simultaneously fit into the computer's main memory. Refer to the discussion of swap space in Phase 1.

```
      3. Create the Swap Logical Disk

Do you want to run this step? [y] ↵
Enter Logical Disk Name:  [swap] ↵
Enter the Physical Disk specification in DG/UX common
format: [sd(insc(),1)]  ↵
```

The default physical disk specification will place **swap** on the same physical disk with **root**.

```
Do you want to display the layout of this
Physical Disk? [n] ↵
Enter the Physical Disk Address of the starting block
of the Logical Disk Piece: [default]  ↵
```

Above, the default address calculated by the **diskman** utility begins at the first block after the end of the **root** logical disk. Again, this default is a hexadecimal location. Accept the default. The next query is for swap size. The default (50,000 blocks) is probably sufficient. Consult your planning worksheet for the size of swap space.

```
Enter the size in blocks of the Logical
Disk Piece: [50000]  ↵
The Logical Disk 'swap' has been created.
```

Creating the usr Logical Disk and File System

The dialogue for creating the **/usr** file system continues similarly to the previous dialogues. Consult your planning worksheet for the desired size of the **usr** logical disk. Or, you can accept the default.

```
        4. Create the /usr Logical Disk and File System

Do you want to run this step? [y] ↵
Enter the Logical Disk Name: [usr] ↵
Logical Disk Piece 1:
Enter Physical Disk specification in DG/UX common
format: sd(insc(),0)
Do you want to display the layout of this
Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block
of Logical Disk Piece 1: [default]  ↵
Enter the size in blocks of Logical Disk
Piece 1: [160000] ↵
```

CAUTION: Do not break **usr** into pieces. Breaking **usr** into pieces will prevent the **diskman** utility from being booted successfully from disk.

Assuming you do not create any more pieces for **usr**, you see the following text:

```
Do you want to specify any more pieces for this
Logical Disk? [n] ↵
The Logical Disk 'usr' has been created.

Making a file system on the logical disk 'usr' ...

Made a file system on Logical Disk 'usr'.

Press New Line when ready to continue.  ↵
```

The **usr** logical disk and an empty file system have been created.

Step 10: Loading the DG/UX Files onto System Logical Disks

In the previous step, you created system logical disks and corresponding empty file systems onto which you will now load the contents of the **/** and **/usr** file systems from the release tape.

Loading the / File System

In this step, the **diskman utility** will load the files from the release tape onto the **root** logical disk you created. These files include a starter system that contains a minimum of information on disk and tape devices.

```
        5. Load the Root File System

   Do you want to run this step? [y] ↵

   Do you want to see the names of the files being loaded? [y] n ↵
```

When asked if you want to see the names of the files being loaded, respond as you wish. If you respond **y**, you will see filenames scroll rapidly up and off the screen. Should error messages be displayed, they, too, will scroll quickly up and off the screen. Try to watch your screen carefully in the event of error messages.

If you respond **n**, you will see a loading message written periodically to your screen as a successful indication. Should errors occur during loading, you will receive error messages. If your computer has not issued a message for several minutes to confirm a successful load, you can conclude that an error has occurred. Verify your installation procedures and retry. If failures persist, report the condition to your system administrator.

The next series of dialogues follows:

```
   Enter the Logical Disk Unit Name: [root] ↵

   Enter the tape drive specification in DG/UX common
   format: st(insc(),4) ↵
   Ready to load the Root File System.
   Mount the first release tape on the tape drive [st(insc(),4)]
```

At this point the major release tape must be in the tape drive. If you have an update tape in the drive, remove it and insert the major release tape now.

If you are performing an initial installation, your major release tape will already be inserted in the tape drive.

The next prompt appears.

```
Press New Line when ready to continue... ↵
```

Make sure that your tape is mounted and online, then press New Line. Loading from tape takes about three minutes. When the load is complete, you will see:

```
The Root File System has been loaded.

Press New Line when ready to continue... ↵
```

Loading the /usr File System

In this step, the **diskman** utility will load the files from the release tape onto the **usr** logical disk you created.

```
      6. Load the /usr File System

Do you want to run this step? [y] ↵

Do you want to see the names of the files being loaded? [y] n ↵
```

When asked if you want to see the names of the files being loaded, respond as you wish. If you respond **y**, you will see filenames scroll rapidly up and off the screen. Should error messages be displayed, they, too, will scroll quickly up and off the screen. Try to watch your screen carefully in the event of error messages.

If you respond **n**, you will see a loading message written periodically to your screen as a successful indication. Should errors occur during loading, you will receive error messages. If your computer has not issued a message for several minutes to confirm a successful load, you can conclude that an error has occurred. Verify your installation procedures and retry. If failures persist, report the condition to your system administrator.

The next series of dialogues follows:

```
Enter the Logical Disk Unit Name:  [usr] ↵
Enter the tape drive specification in DG/UX common
format: [st(insc(),4)] ↵
Ready to load the /usr File System.
Mount the first release tape on the tape drive [st(insc(),4)]
```

Your major release tape has already been inserted in the tape drive. Do not change tapes at this point.

The next prompt appears.

```
Press New Line when ready to continue... ↵
```

Make sure that your tape is mounted and online, then press New Line.

Loading from tape takes about 15 minutes. When the load is complete, you will see:

```
The /usr file system has been loaded.

Your starter system has been installed.

Press New Line when ready to continue... ↵
```

You have completed all the installation steps in the Initial Installation Menu. Pressing New Line at this point returns control to the Initial Installation Menu.

If you are not updating your system (you did not receive an update tape), skip over Step 11 and proceed to Phase 3.

If you are updating your system (you did receive a update tape), proceed to Step 11 to load the contents of the update tape.

Step 11: Updating the / and /usr File Systems

You will perform this step under two conditions:

- You are performing an initial installation (from a major release tape) and an update (from an update tape). For example, you received both the 4.30 release and the 4.31 update release.

- You are performing an update only (from an update tape). For example, you received the 4.31 update release only. You have a 4.30 operating system installed.

If you do not have a second tape to load, skip over this step and go to Phase 3.

If you are installing a major release over another major release (for example, 4.20 is installed and you wish you install 4.30 over it), go to Appendix F. At this point, you should have your update tape inserted in the tape drive (make sure the tape is online and at the beginning of the tape).

Your goals are to:

- Reinstall the bootstraps on your disk (if updating only)

- Register the physical disk (if updating only)

- Update **/** file system

- Update **/usr** file system

Reinstalling Bootstraps on Disk

CAUTION: You need to perform this procedure if you are updating only (you have an update tape only).

A bootstrap is a rudimentary program that loads the DG/UX system kernel from disk. A different bootstrap is included with each release tape. Therefore, you need to reinstall it each time you load an update tape. The kernel contains a minimal set of functions necessary to perform I/O transactions, manage and control hardware, and schedule processes.

Follow these procedures to reinstall the bootstrap.

- From the Initial Installation Menu, select the caret (^) to go to the Diskman Main Menu.

- Select option 1, "Physical Disk Management Menu."

- Select option 5, "Format a Physical Disk."

- Select option 4, "Reinstall Bootstraps on a Physical Disk."

You are prompted:

```
Enter the Physical Disk Specification in DG/UX common
format: sd(insc(),0) ↵
Enter the pathname of the boot.aviion file: [/usr/stand/boot.aviion] ↵
Installed Bootstraps on the Physical Disk sd(insc(),0).
```

In the preceding dialogue, you named the disk on which the new bootstrap will be written and you accepted the default location for the boot file. Pressing New Line causes the new bootstrap to the loaded.

You are prompted:

```
Press New Line when ready to continue ... ↵
```

You will register your physical disk next.

Registering a Physical Disk

CAUTION: You need to perform this procedure if you are updating only (you have an update tape only).

You want to register a physical disk so that all logical disks on the physical disk are known to the system.

- From the Physical Disk Management Menu, select option 1, "Register, Deregister, or List Registered Physical Disks."
- Select option 1, "Register a Physical Disk."

You are prompted:

```
Enter the Physical Disk Specification in DG/UX common
format: sd(insc(),0) ↵
Physical Device sd(insc(),0) has been registered.
```

In the preceding dialogue, you supplied the name of the device to be registered. Pressing New Line starts the registration.

Your physical device has been registered.

Updating the / and /usr File Systems

Now you are ready to update the contents of the existing **/** and **/usr** file systems on disk with the contents of the update tape.

CAUTION: Make sure your update tape is inserted in the tape drive.

From the Diskman Main Menu, select option 5, "Update Installation Menu." The following menu is displayed:

```
            Update Installation Menu

1. Update the Root File System

2. Update the /usr File System

3. All Update Steps

Enter ? or <number>? for help, ^ to GO BACK, or q to QUIT.

Enter Choice: [3]
```

You are prompted:

```
All Update Steps
     1.  Update the Root File System
Do you want to run this step? [y] ↵
Do you want to see the names of the file being loaded? [y] n ↵
```

When asked if you want to see the names of the files being loaded, respond as you wish. If you respond **y**, you will see filenames scroll rapidly up and off the screen. Should error messages be displayed, they, too, will scroll quickly up and off the screen. Try to watch your screen carefully in the event of error messages.

If you respond **n**, you will see a loading message written periodically to your screen as a successful indication. Should errors occur during loading, you will receive error messages. If your computer has not issued a message for several minutes to confirm a successful load, you can conclude that an error has occurred. Verify your installation procedures and retry. If failures persist, report the condition to your system administrator.

The next series of dialogues follows:

```
Enter the Logical Disk Unit: [root] ↵
Enter the tape device specification in DG/UX common
format: [st(insc(),4)] ↵
Ready to load the Root File System.
```

The default root logical disk is selected and the specification for the device containing the update date is given.

The next prompt is given:

```
Mount the first release tape on the tape drive st(insc(),4).

Press New Line when ready to continue... ↵
Loading ...
```

```
.
.
.
The Root File System has been loaded.
```

Pressing New Line starts the root logical disk update. The loading message is displayed and rewritten to the screen for the duration of the load until the process has completed.

Next, the **/usr** file system will be updated. The preceding dialogue used for the update of the root file system is repeated for the **/usr** file system. A summary of the dialogue for updating **/usr** is given as follows:

```
     1.  Update the /usr File System
Do you want to run this step? [y] ↵
Do you want to see the names of the file being loaded? [y] n ↵
```

When asked if you want to see the names of the files being loaded, respond as you wish. If you respond **y**, you will see filenames scroll rapidly up and off the screen. Should error messages be displayed, they, too, will scroll quickly up and off the screen. Try to watch your screen carefully in the event of error messages.

If you respond **n**, you will see a loading message written periodically to your screen as a successful indication. Should errors occur during loading, you will receive error messages. If your computer has not issued a message for several minutes to confirm a successful load, you can conclude that an error has occurred. Verify your installation procedures and retry. If failures persist, report the condition to your system administrator.

The next series of dialogues follows:

```
Enter the Logical Disk Unit: [usr] ↵
Enter the tape device specification in DG/UX common
format: [st(insc(),4)] ↵
Ready to load the /usr File System.
Mount the first release tape on the tape drive st(insc(),4).

Press New Line when ready to continue... ↵
Loading ...
.
.
.
The /usr File System has been loaded.
You can now reboot a kernel from disk.
```

You have now completed updating both the **/** and **/usr** file systems. If you performed an initial installation only, type **q** to exit the **diskman** utility.

If you performed an update only, you will see this prompt:

```
Do you want to return to SCP-CLI? y↵
```

Entering **y** exits the **diskman** utility and control goes to the SCM.

In both instances, you will reboot the kernel from disk in Phase 3.

End of Phase 2

Phase 3: Customizing the Primary Release

At this point, your system should have the correct system logical disks and file systems, and the system software should be loaded. You are now ready to complete the setup and installation of your system. The steps in this phase are:

- Booting the Starter Kernel.
- Specifying Starter Devices
- Creating Other Logical Disks and File Systems.
- Loading Software Packages with **sysadm**.
- Setting Up Software Packages with **sysadm**.
- Building a Custom Kernel.
- Setting Default Boot Characteristics.
- Starting System Administration.

Step 12: Booting the Starter Kernel

Booting the starter kernel on a preloaded system is easy because the factory has set the computer's default boot path to the correct device specification. If you have a preloaded system, read the following section, "Booting the Starter Kernel on a Preloaded System."

Booting the starter kernel on a non-preloaded system, on the other hand, is more involved because you need to know the correct device specification yourself. If your system does not have a preloaded disk, skip the next section and proceed either to "Booting an AViiON 5000/6000 System" or to "Booting Other Servers and Stand-Alone Workstations."

Booting the Starter Kernel on a Preloaded System

After correctly installing your system's hardware, power on your system's peripherals, then power on the computer itself. After conducting power-on diagnostics, it will proceed to boot the starter kernel on your preloaded disk. On a correctly-installed preloaded system, you do not need to know the complete boot command, but you may want to learn it anyway. The following two sections discuss the complete boot command line. If you do not want to learn about booting at this time, skip the next two sections and proceed to the section, "Specifying Starter Devices."

If your system does not boot the starter kernel at power-on but instead puts you in the SCM, you see this prompt:

```
SCM>
```

If this happens, verify that you have installed the hardware correctly and that you have powered on your peripherals before turning on power to the computer. Next, you need to boot the starter kernel yourself from the SCM prompt. The following two sections tell how to boot.

Booting an AViiON 5000/6000 System

To boot the starter kernel from an ESDI disk, use a command line like this:

```
SCM> b cied(m,n)root:/dgux.starter ↵
```

where *m* is the disk controller number and *n* is the disk unit number. The controller and disk unit are both **0** unless you have deliberately configured a different disk to be your root disk. If you have only one ESDI controller, you can omit the controller number from the device specification (remember to include the comma): **cied(,***n***)**.

On an OS server with an ESDI disk, when the starter kernel prompts for a device name, enter **cied()**.

To boot an OS server (AViiON 5000/6000 systems only) that have a SCSI disk, use a command line like this:

```
SCM> b sd(cisc(m),n)root:/dgux.starter ↵
```

where *m* is the disk controller number and *n* is the SCSI ID. The controller and SCSI ID are both **0** unless you have deliberately configured a different disk to be your root disk. If you have only one **cisc** controller, you can omit the **0** controller specifier and more simply refer to the disk as **sd(cisc(),0)**.

On AViiON 5000/6000 systems with a SCSI disk, the starter kernel prompts you for the device name. Respond with **sd(cisc(),0)**. If your server has a SCSI tape device, respond with **st(cisc(),4)** after entering the SCSI disk device name.

To boot an OS server with an SMD disk, use a command line like this:

```
SCM> b cimd(m,n)root:/dgux.starter ↵
```

where *m* is the disk controller number and *n* is the SCSI ID. The controller and SCSI ID are both **0** unless you have deliberately configured a different disk to be your root disk. If you have only one SMD controller, and the first disk is the root disk, you can refer to it as **cimd(0)**.

On OS servers with an SMD disk, when the starter kernel prompts for a device name, enter **cimd()**.

Booting Other Servers and Stand-Alone Workstations

If you are booting a stand-alone workstation (that is, one that has its own disk) or a server other than an AViiON 5000/6000 series system, you boot the starter kernel with a command line like this:

```
SCM> b sd(insc(m),n)root:/dgux.starter ↵
```

where *m* is the disk controller number and *n* is the SCSI ID. You will probably be booting the device on controller **0** and SCSI ID **0**: **sd(insc(0),0)**, which you can also express as **sd(insc(),0)**.

If you have an AViiON 400- or 4000-series system that has its root disk attached to a Ciprico SCSI adapter rather than to the integrated SCSI adapter (which is unlikely), refer to the part in the preceding section about booting AViiON 5000- and 6000-series systems that have SCSI disks.

Step 13: Specifying Starter Devices

When the starter kernel begins execution, it first asks you for device names. If you loaded your system software yourself (that is, you do not have a preloaded system), you saw the same prompt after booting **diskman** from tape in Step 6; this time, however, you respond differently.

Your starter kernel is the same whether your system was preloaded originally or whether you loaded the system software yourself.

After the boot messages have appeared on your console, you see this display:

```
            DG/UX Starter System
Enter the names of the devices you will use in Common Device
Specification Format, with one name per line.  Enter just
newline when done.

Examples: sd(insc(),0) st(insc(),4) cird() st(cisc(),4)

Include duart() for servers and kbd() and grfx() for
workstations.

Device Name?
```

At the `Device name?` prompt, the devices you may enter are:

cied() If you have disks connected to a Ciprico ESDI controller.

cimd() If you have disks connected to a Ciprico SMD controller.

cird() If you have disks connected to Ciprico ESDI and/or SMD controllers (this specification is an abbreviation for **cied()** and **cimd()**).

st(cisc(),4) If you have a tape drive attached to a Ciprico SCSI adapter (on VME bus).

duart() If you have an integrated Duart terminal line controller.

sd(insc(),0) If you have disks attached to an integrated SCSI adapter.

st(insc(),4) If you have a tape drive attached to an integrated SCSI adapter.

kbd() If you have a graphics console keyboard.

grfx() If you have a graphics console monitor.

Table 2-6 shows the possible starter devices for AViiON systems.

Table 2-6 Possible Starter System Devices

AViiON 5000/6000 Systems	Other Systems
sd(cisc(),0)	**sd(insc(),0)**
st(cisc(),4)	**st(insc(),4)**
cird()	**sd(cisc(),0)**
cied()	**st(cisc(),4)**
cimd()	**duart()**
duart()	**inen()**
hken()	**hken()**
	kbd()
	grfx()

Refer to the device list you made in Step 3 for the disk device names you need to enter. The mnemonic **cird** covers both **cied** and **cimd** devices.

On a workstation or AViiON 4000 system, list the **duart()** if your keyboard is connected to the terminal (which is connected to your system's RS232 port). If you are using the graphics monitor, on the other hand, and your keyboard is connected to the keyboard port on the computer itself, enter the **kbd()** and **grfx()** devices instead of **duart()**.

You need the Ethernet LAN controller device entries (**hken()** and **inen()**) only if you want to be able to access a remote tape device during the initial setup and installation of your system. You do not need to do so for normal installation of the DG/UX system and network packages.

For example, for a typical workstation that has its own tape and disks, you would enter:

```
Device Name? kbd() ↵
Device Name? grfx() ↵
Device Name? st(insc(),4) ↵
Device Name? sd(insc(),0) ↵
Device Name? ↵
```

Another example: for an AViiON 6000-series system with a tape device and two SMD disks, you could enter:

```
Device Name? duart() ↵
Device Name? st(cisc(),4) ↵
Device Name? cird() ↵
Device Name? ↵
```

After you have finished entering your device names, terminate prompting by pressing the New Line key after the last `Device Name?` prompt. Your system then displays some messages on the console:

```
Using /dev/dsk/swap as swap file

** root:
No check necessary for root

Mounting /dev/dsk/root as root file system

INIT: Boot options are: init
INIT: Cannot open /etc/TIMEZONE. Environment not initialized.

INIT: /etc/inittab file created from /etc/inittab.proto prototype

INIT: Checking and mounting /usr...

INIT: /usr is now mounted

INIT: SINGLE USER MODE
su: unable to access /etc/passwd
#
```

The messages from `INIT` concerning **TIMEZONE** and **passwd** are not errors. They simply indicate that your system is not yet fully initialized.

Setting Up DG/UX: Initial Configuration

Run levels are explained in Chapter 3. For now, simply follow our instructions. To continue with installation, you need to change run levels and go to administrative mode, run level 1, where the **sysadm** program and local file systems are available. Note also that as you go into run level 1 for the first time, a number of system data base files are automatically created from prototype files.

To change run levels from single-user level S (which is where you are now) to administrative run level 1, type:

```
# init 1

INIT: New run level: 1

chk.fsck:

chk.date:
   Current date/time: Wed Jul 26 08:15 EDT 1990
-  Is the current date, time, and TIMEZONE correct? (y n) [n]:
```

By accepting the default to this question, you may reset the time. From this point, the system proceeds to perform a number of setup operations. After initializing a number of files in **/etc** and some other directories, the system displays this message:

```
chk.system:
   Cleanup the /etc/ps_data file and /etc/log files.
   Check for missing local passwords.
```

```
** WARNING: These local accounts have NO password.
   root::0:1:root:Special Admin Login/:/sbin/sh
   sysadm::1:sysadm:Regular Admin Login/admin:/sbin/sh
```

CAUTION: *Take special note of the warning above! Accounts that have no password allow any user to log into the system (without a password) as* **root** *or* **sysadm** *with superuser status. You should assign passwords for the* **root** *and* **sysadm** *logins as soon as possible; otherwise, you leave your system (and other systems on your network) open to security violations. Change these passwords with the* **passwd***(1) command.*

Every time you bring your system up to a level above run level 1, the **chk.system** script checks for local login accounts that do not have passwords. You should assign passwords to such accounts as soon as possible. As superuser, you can check for such insecure accounts at any time by running the **chk.system** script (located in **/usr/sbin/init.d**) yourself. See Chapter 3 for more information on the **chk.system** script and how to run it.

The last few scripts that run during DG/UX system setup perform functions like creating short names for device nodes, mounting some file systems, and checking to see if there are any packages that still need to be set up. Finally, you receive this message:

```
Press <RETURN> to display prompt.
```

Press New Line and log in as **sysadm**:

```
no_node
DG/UX Release 4.30
login: sysadm ↵
```

Note that we log in as **sysadm**, not **root**. Logging in as **sysadm** puts you in the **/admin** directory and gives you all the superuser privileges of the **root** login. Note when you log in that you do not need a password. Neither the **root** nor the **sysadm** logins required passwords initially (otherwise, you would have no way to log into the system the first time). You should set passwords for these profiles as soon as possible to avoid breaches in security. Use **passwd**(1) to set passwords.

If you keep **.profile** and other personal files in **/admin**, this ensures that **/** (root) remains a clean, protected directory that you can always boot.

Step 14: Creating Other Logical Disks and File Systems

At this point in the installation process, you create the remaining logical disks and file systems that you need on your system. Use the plans you made in Step 2 as a guide.

If you have multiple disks, you need to decide which logical disks belong on which physical disks. If you have trouble arranging your logical disks to fit on your physical disks, remember that you may split logical disks into as many as 32 pieces. With this in mind, you should have no trouble making your logical disks fit.

Another point to keep in mind is the performance benefit you may be able to achieve by distributing your most frequently accessed logical disks across multiple physical units. If many users will be accessing a single database, consider dividing the database's logical disk into several pieces and assigning the pieces to different physical disks. Distributing data like this provides for greater concurrency of access and better I/O performance for your users. This technique works particularly well in environment where you have multiple SCSI disks.

To create a logical disk, use the **sysadm diskmgmt** command:

```
# sysadm diskmgmt ↵
```

First, select the Logical Disk Management Menu, then select option 1, Create a Logical Disk. One by one, create the logical disks that you need. When you have finished, you create the file systems for the logical disks.

Adding File Systems to /etc/fstab with sysadm

Now that the file systems are created, we need to make them accessible. We do this with the **sysadm addfsys** command. For each file system, you need to decide on a point in your directory structure where you want the file system to reside. The file system you created on the **usr_opt_X11** logical disk, for example, resides on the **/usr/opt/X11** directory. The directory is called a mount point directory because that is where the file system is mounted.

If we wanted to mount the **sales** logical disk and file system on the **/sales/accounts** directory, the **addfsys** dialogue would proceed like this:

```
# sysadm addfsys ↵

Running subcommand 'addfsys' from menu 'fsmgmt',
FILE SYSTEM MANAGEMENT

Mount Directory Name?  /sales/accounts ↵
Is this a local file system?  [yes]  ↵
Writeable?  [yes]  ↵
Dump Cycle? [d] ↵
fsck Pass? [1] ↵
Export? [no] y ↵

The entry for /sales/accounts has been added.
```

```
The directory, /sales/accounts, does not exist.
Create /sales/accounts? [yes]  ↵
Mount the file system?  [yes]  ↵
```

Invoke the **addfsys** command again to mount the rest of your file systems. Diskless clients will use this same procedure to gain access to the file systems on the server's disk. See Chapter 8 for more information on accessing file systems.

NOTE: When you mount the **/srv** and **srv/swap** file systems, do not export them. The **sysadm** program will later export subdirectories of these file systems as OS clients are added.

Step 15: Loading Software Packages with sysadm

The **sysadm loadpackage** function allows you to load all software packages that are on a single tape at once. Your tape may include packages for TCP/IP, ONC/NFS, YP, Frame 1.3, and the X Window System™ in addition to the DG/UX system package. The **loadpackage** command displays the names of all packages on the tape; you select the ones you want to load.

CAUTION: If you received an update tape with a major release tape, you will have to perform these procedures twice; once for the major release tape and a second time for the update tape. Insert the update tape in the tape device before you start the procedure a second time.

Note that some packages load into directories under **/usr/opt**. The **/usr/opt** directory is on the **usr** logical disk, which is not large enough to accommodate any packages besides the DG/UX system software. To avoid overflowing the **usr** logical disk, create special logical disks for packages that load under **/usr/opt** and then mount these logical disks under **/usr/opt**.

Before loading packages, make sure that any logical disks and file systems intended for them are mounted in their correct places. If you forget and load a package without mounting its file system, the load may fail. When this happens, simply remove the directory created for the loaded package, then recreate it empty and mount the proper file system on top of it. If you are loading the X11 package, for example, make sure the **usr_opt_X11** logical disk's file system is mounted at **/usr/opt/X11**.

Note that **loadpackage** loads files into a release area relative to the load point specified in the release's tape table of contents.

Before you use **loadpackage**, you must first run **sysadm makesrv** to create the **/srv** directory tree. If you have created the **/srv** file system on a separate logical disk, make sure it is mounted before running **makesrv**. Then begin loading packages.

The following example shows a typical **loadpackage** dialogue.

```
# sysadm loadpackage ↵

Running subcommand 'loadpackage' from menu 'releasemgmt',
Software Release Management

Release Area? [PRIMARY] ↵
Tape Drive? [0] ↵
Is the Tape Mounted and Ready? y ↵
Load Package X11.lg? [yes] ↵
Load Package X11.man? [yes] ↵
Load Package X11? [yes] ↵
Load Package dgux.man? [yes] ↵
Load Package dtk.man? [yes] ↵
Load Package dtk? [yes] ↵
Load Package gcc.man? [yes] ↵
Load Package gcc? [yes] ↵
```

```
Load Package nfs.man? [yes] ↵
Load Package nfs? [yes] ↵
Load Package tcpip.man? [yes] ↵
Load Package tcpip? [yes] ↵
List file names while loading? [yes] n↵
Mount Volume 1.
Is the tape mounted and ready? y↵
Skipping tape file 0 to 40.
.
.
.
Updating proto root (/usr/root.proto).
Updating MY_HOST root (/srv/release/PRIMARY/root/MY_HOST).
loadpackage is finished.
#
```

The packages that **loadpackage** queries you about depend on the DG/UX package you bought. The packages listed above reflect the contents of the DG/UX Server/Client package. The number of files skipped at the beginning of the load also depends on the package you purchased.

Loading products can take as much as a half an hour or more, depending on how much you are loading. The release you are loading may be on more than one tape, in which case you repeat the process above until the load is complete.

Step 16: Setting Up Software Packages with sysadm

Different packages set themselves up in different ways. See the release notice for each product you need to set up. In the typical configuration, you need to set up TCP/IP, ONC/NFS, and YP before building a new kernel. In setting up a package with **sysadm setuppackage**, you answer queries or supply parameters or data needed by a given package. If you have never run **sysadm makesrv** on your system, do so before setting up software packages. If you have created the **/srv** file system on a separate logical disk, make sure it is mounted before running **makesrv**.

Setting Up TCP/IP

See your network administrator and *Setting Up and Managing TCP/IP on the DG/UX™ System* for detailed information on the TCP/IP product. To install TCP/IP, you need the information that you assembled in Step 6, "Assembling Network Information for Your System."

Once you have the required information, invoke **sysadm setuppackage**.

Setting Up ONC/NFS

You will not need to answer any questions when setting up the ONC/NFS package with the **sysadm** utility. See *Managing NFS and Its Facilities on the DG/UX™ System* for basic information and concepts. The parameters for ONC/NFS are in **/etc/nfs.params**. You may read this file for instructions on entries you may want to modify. A prototype of **nfs.params** will be initialized if you choose not to modify it.

Setting Up Yellow Pages (YP)

As stated in Step 4, you need to decide whether your system will function as a YP master, YP server, or YP client before setting up the Yellow Pages. You also need to know the name of your YP domain.

Initially, you set up YP the same way for all systems, whether they will be masters, servers, or clients. If your system will be a YP client, this initial setup is sufficient. If your system will be a master or server, however, there are few more things you need to do after you have built and booted your new kernel. You build a new kernel in the next step, Step 17, and you boot it in Step 18. If you are installing a YP master or server, you finish YP setup at the end of Step 18.

Step 17: Building a Custom Kernel

After setting up packages, build a new kernel for your system. To build or rebuild a kernel, you first edit the prototype system file to reflect what you actually have on your system. Note that you will need to be familiar with a text editor, such as **vi**. To begin, use **sysadm newdgux** to edit the system file and run the build programs. The **newdgux** command concatenates the available prototype files together into a single file. For example, your system file may include system prototypes for the DG/UX operating system, TCP/IP, and ONC/NFS. If you have other products, the prototype files for those products (if any) would also be concatenated.

To build a custom kernel, follow these steps:

1. Execute **newdgux**:

   # sysadm newdgux ↵

   The following message appears on the screen:

   ```
   Running subcommand 'newdgux' from menu 'sysmgmt',
   SYSTEM CONFIGURATION MANAGEMENT

   System Name? [aviion]
   ```

   Specify the name of the system file that contains your changes to the system's parameters and a list of all devices on your system. If your system is preloaded, or if you are loading your system for the first time, you will not already have such a file. In this case, you begin with the provided prototypes.

2. The default system name is the prototype system configuration file, **aviion** for servers and stand-alone machines. If you are configuring for a server or stand-alone machine, just press New Line.

   If you are configuring for a diskless client, type **diskless** and press New Line. The following prompt then appears:

   ```
   System File /usr/src/uts/aviion/Build/system.aviion does not
   exist.
   Create the system file? [yes]
   ```

3. Assuming that you do not already have a system file, press New Line to create one. If you want, you can quit at this point. To do so, type **q** and press New Line. If you type **n** and press New Line, **sysadm** returns you to the `System Name?` prompt.

   After you specify that you want to create the system file, the following prompt appears:

   ```
   Editor? [vi]
   ```

4. Press New Line to use **vi**, or type the name of the editor you want to use. The following section offers some helpful hints for **vi** novices.

Editing with vi

Do not expect **vi** to behave like your favorite personal computer editor. Keep in mind that **vi** operates in two modes: input mode and editing mode. In input mode, anything you type (except for the **Esc** character) goes into the file as text. In editing mode, any keys you type are commands that tell **vi** things like where to move the cursor, what text to delete, what text to move, and so on.

When you first invoke **vi**, it is in editing mode. Table 2-7 shows the commands you can use in editing mode. These commands are case sensitive.

Table 2-7 Hints for vi Novices

Command	Description
<Ctrl-F>	Move forward one screen.
<Ctrl-B>	Move backward one screen.
G	Move to the bottom of the file.
j	Move the cursor down one line.
5j	Move the cursor down five lines.
k	Move the cursor up one line.
/hken ↵	Move to the next occurrence of **hken**.*
?NODE ↵	Move to the previous occurrence of **NODE**.
l	Move ahead one character.
h	Move back one character.
w	Move ahead one word.
b	Move back one word.
dw	Delete to the end of this word.**
dd	Delete this line.
8dd	Delete this line and the next 7.
u	Undo my last change.
i	Enter input mode (described below), inserting text at the cursor.
I	Enter input mode, inserting text at the beginning of the line.
o	Enter input mode, opening a new line below the current position.
<Ctrl-R>	Redraw the screen.
:w ↵	Write (save) my changes to the file.
:w! ↵	Force changes to the file, overriding permissions.***
:q ↵	Exit **vi** (only works if you have not changed the file).
:q! ↵	Exit **vi** without writing my changes.

* Searches in **vi** wrap; that is, if **vi** has not found the requested string when it reaches the end of the file (or the beginning, if you search with **?**), **vi** will then search from the top (or the bottom) of the file back to the cursor position.

** Words are marked by punctuation as well as spaces.

*** The **w!** forced write works only if you are the superuser or the owner of the file.

In **vi** input mode, anything you type goes into the file as text until you press the **Esc** key, which returns you to editing mode.

For more information on **vi**, see *Using the DG/UX™ Editors* or the **vi**(1) manual page in the *User's Reference for the DG/UX™ System*.

Editing the system File

Once you specify the editor you want to use, the system file appears on your screen. The system file contains comments (lines starting with **#**) to help guide you through your decisions.

The following section highlights a few major parts of the system file that you may need to change. The first important part is for devices. Initially, the system file contains several lists, each for a different kind of Data General AViiON system. You should remove or comment out the lists for the systems that you do *not* have. Then verify that the remaining list is correct for the system that you do have. You may need to add entries for devices that you have but that do not already appear in the system file.

In the following example, text appearing in **boldface** represents text that we have added to the system file. The example is for an AViiON 5000 series system with ESDI disks and one Hawk LAN controller. The system also has a lineprinter, a tape device, and a **syac** asynchronous terminal controller. A **#** appears before devices we do not have (the **#** sign comments them out so they are not configured into the system). For more example system files, see the examples at the end of this chapter.

```
##### Typical AViiON 5000 or 6000 series server configuration:

  cird()          # Ciprico Rimfire or SMD disk controller
#  sd(cisc(),*)   # all SCSI disk drives on Ciprico SCSI adapter
  st(cisc(),*)    # all SCSI tape drives on Ciprico SCSI adapter
  syac()          # Systech terminal line controller
  duart()         # integrated Duart terminal line controller
  hken(0)         # 1st Interphase VME Ethernet controller
  lp()            # integrated line printer controller
#  hken(1)         # 2nd Interphase VME Ethernet controller

  ptc()           # pseudo-terminal controller device
  pts()           # pseudo-terminal slave device
  pmt()           # pseudo-magtape device
  log()           # Streams logger pseudo-device
  prf()           # profiler pseudo-device
```

Note the asterisk (*) in the SCSI ID field of the SCSI device specifications above. The asterisk is a shorthand way of specifying all devices of that type.

The next important section of the system file deals with tunable configuration parameters. Again, you will find helpful comments in the system file itself. Any parameter settings you provide in the system file override the system defaults. Chapter 4 discusses some of the tunable parameters in more detail.

You may need to set the following parameters:

TZ Your timezone, represented as the number of minutes that you follow Greenwich Mean Time (GMT). The Eastern Standard Time (EST) zone, for example, is 300 minutes behind GMT.

MAXUP The maximum number of processes that any user will be able to have at one time.

NODE The node name that **uname**(1) and UUCP use. This name should be unique within your network. If you have a subnetted environment where you have one main network linking a number of subnetworks, node names should be unique within the entire main network (not just unique within a subnet).

DUMP The tape device that will be the default for dumps taken in the event of system emergencies. Two entries for DUMP appear in the system file: one for workstations that have a tape device and one for OS client workstations that dump over the net. We select the one with the tape device and change it to refer to the Ciprico SCSI adapter (**cisc**()) instead of the integrated SCSI adapter (**insc**()).

The following example shows how you might set parameters for an AViiON AV6000 system named **sales**:

```
TZ              300
MAXUP           64
NODE            "sales"

DUMP            "st(cisc(),4)"
```

There are some tunable parameters that apply only to diskless workstations that are OS clients:

DUMP Same as the DUMP parameter just described, except this is the diskless workstation entry (**inen**()).

PERCENTNFS
Set this to **100** to get the best possible NFS performance.

NETBOOTDEV
The device from which your workstation boots over the net.

ROOTFSTYPE
The type of your workstation's root file system.

SWAPDEVTYPE
The type of your workstation's swap device.

Other parts of the system file have to do with the network configuration.

When you have finished editing and saving the file, exit your editor. Next, you will see the following:

```
Ready to Configure a Kernel? [yes]  ↵

sysadm will now run config on /usr/src/uts/aviion/Build/system.aviion
```

If **config** encounters errors, you will see this:

```
Warning config failed.  You may print the error output
from config.
Print the config output file?  [yes]
```

If you print the error output file, it will show you where the errors are. If **config** succeeds, you will see this:

```
Config succeeded.

sysadm will now attempt to build a kernel.
Building ...
```

If the build fails, you will see the following:

```
Warning:  The kernel build failed.  Since the system file
was checked by config, this failure should not have
happened.  There are two main reasons for such a failure.

1)  The logical disk containing the build area (usually
/usr) ran out of space.  Remove some files to make
space and try newdgux again.

2)  Some distribution files and libraries are missing.
Check the build area (/usr/src/uts) against the
distribution tape(s).

Newdgux must give up at this point.  You may print the
output file if you wish.

Print the Build Error File? [yes]
```

If the build succeeds, you see this:

```
The build succeeded.
```

Now you can install your new, customized kernel:

```
Install the New Kernel? [no] y↵
For a Diskless Client of this Host? [no] ↵
Kernel Pathname? [/dgux.aviion]  ↵

The new kernel has been copied to /dgux.aviion.
Link /dgux to the New Kernel? [yes]  ↵
```

```
The new kernel will not take effect until you shutdown and reboot.
To do this, quit sysadm, and say:

  cd /
  /etc/shutdown
  /etc/halt -q

Until you do this, a few commands which depend on the symbol
table in /dgux (such as the kernel profiler and netstat) may
not work correctly.  This should not cause any serious
difficulties.
#
```

As the instructions at the end of the **newdgux** display say, you may now take your system down:

```
# cd / ↵
# /etc/shutdown -g0 -y ↵
# /etc/halt -q ↵
```

Read the next step before booting.

Step 18: Setting Default Boot Characteristics

Now that your kernel is configured, you should do these things:

- Set the boot path default and then boot the system.
- Optionally change the initial run level from single user mode to multi-user mode.
- Complete YP setup.

Setting the SCM Boot Path Default

You can set a default boot path for the DG/UX operating system, which allows you to enter only the **b** (boot) command from the SCM prompt to boot your system.

Enter the **f** (format) command at the SCM prompt:

```
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) -> 1↵
```

Selecting the option to change the boot parameters displays the following menu:

```
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) -> 1↵
```

Selecting the option to change the boot parameters starts a dialogue. The following example dialogue shows how to set your default boot path if you have one ESDI disk:

```
System boot path = [ ]

Do you want to modify the boot path? [N] y ↵

Enter new system boot path -> cied()root:/dgux ↵

System boot path = [cied()root:/dgux]
Do you want to modify the boot path? [N] n ↵

Do you want to boot? [N] y ↵
```

Respond to the last prompt with **y** followed by the New Line key.

From this point on, with the system boot path set correctly, you type the single letter **b** at the SCM prompt to boot the system.

NOTE: As a future option, if you want to override the built-in root logical disk that gets mounted at boot time, use the **−a** option with the SCM **boot** command. When you provide this option, the kernel prompts you for all boot information. If you have a logical disk named **alt_root** from which you want to boot, use a command line like this:

```
SCM> b cied()alt_root:/dgux −a ↵
```

Changing the Initial Run Level

When your system comes up, it is by default in run level **s**. Normally, you want to come up in run level 3 because more services are available. To set your initial run level, edit **/etc/inittab** and set the default initial run level to the desired level. We recommend setting it to run level 3. Change the **s** entry to **3** on the line of the file containing the initdefault action so that it looks like this:

```
def:3:initdefault
```

NOTE: As a future option, you can override the default run level by adding an option to the SCM **boot** command line. For example, to come up in single user mode, add the **−s** option:

```
SCM> b cied()root:/dgux −s ↵
```

If your default init run level were already single user mode, you could come up in run level 3 by adding the **−3** option to the boot command line.

Completing YP Setup

You set up YP with **setuppackage** before you built and booted your new kernel. Initially, YP sets your system up as a YP client. If you intend to use your system as a YP master or server, however, there are still a few things you need to do.

To set your system up as the first (or only) YP master in your YP domain, see *Managing NFS and Its Facilities on the DG/UX™ System*.

To set your system up as a YP master in a YP domain that already has a master, follow these steps:

1. Edit **/etc/nfs.params** and find the line where **ypserv_START** is assigned a value. Set **ypserv_START** to equal "**MASTER**":

   ```
   ypserv_START="MASTER"
   ```

2. Find the line in **/etc/nfs.params** where **yppasswd_ARG** is assigned a value, and set it to equal "**/etc/passwd −m passwd**":

   ```
   yppasswd_ARG="/etc/passwd -m passwd"
   ```

3. Execute these commands in the shell:

   ```
   # ypinit −m ↵
   # yppasswd /etc/passwd −m passwd ↵
   ```

4. If you want to run your system as a YP server as well as a master, execute this command:

   ```
   # ypserv ↵
   ```

After completing these steps, your system runs as a YP master in a domain where another master already exists.

To set your system up as a YP server in an existing YP domain, follow these steps:

1. Edit **/etc/nfs.params** and find the line where **ypserv_START** is assigned a value. Set **ypserv_START** to equal "**SERVER**":

   ```
   ypserv_START="SERVER"
   ```

2. Execute these commands in the shell:

   ```
   # ypinit ↵
   # ypserv ↵
   ```

Your system is now running as a YP server in your domain.

For more information on the YP facility, see the ONC/NFS Release Notice and the manual *Managing NFS and Its Facilities on the DG/UX™ System*.

Step 19: Starting System Administration

In this section, you begin doing some of the more traditional system administration tasks. You can add user accounts, add terminals, add local and remote printers, and start system accounting programs. By setting some of these services up now, your system will be ready when users log in.

Adding User Accounts

All devices on your system should be configured at this point. Now you may want to add one or more users to your system. On a server and workstation client, you should have a login account for yourself in addition to the **sysadm** administrator's login. Go to Chapter 14, "User Account Management" for information on setting user defaults and adding user accounts.

Setting Up Terminals and Printers

The procedure for adding tty lines depends on the number of tty entries that you have in **/dev**. If you have less than 16, then use this procedure as described below. If you have more than 16, then you need to run **sysadm newdgux** and change the NPROC variable to suit your needs. NPROC determines the maximum number of processes that can be active at one time on your system. Each idle tty line uses one process. Each active tty line is usually using two to three processes. In addition, there are a number of proccess in the system that are not associated with a tty line. Therefore, a good rule of thumb is to set NPROC to four times the number of tty lines. If, for example, you have 84 ttys, you need to change the NPROC variable from its default value of 64 up to 336. Calculate as follows:

```
NPROC = 4 * number-of-tty-lines

336   = 4 * 84
```

Resetting the NPROC variable prevents the process table from overflowing when processes are started on the ttys. After running **newdgux**, reboot your system to initialize the new kernel, then run **sysadm installtty** which spawns a **getty** process on every available tty line (all tty entries in **/dev**). If you have **getty** processes running on unused lines, you can edit **/etc/inittab** and change the "respawn" field to "off" for those you don't want activated.

The following example installs all tty ports. To begin, type:

```
# sysadm installtty ♪

Running subcommand 'installtty' from menu 'ttymgmt',
TTY MANAGEMENT

Installtty adds tty login entries for all new tty devices.
A tty device is 'new' if it has a device entry in /dev but
has not yet been added to the list of login ttys.  Since you
may be adding more than one tty, you will define a single
```

```
set of tty values to be used for each entry.  You may use
modtty later to change a particular tty entry.

Login State? [on] ↵
Lineset Name? [9600] ↵

Hangup Delay (in seconds)? [0] ↵
TERM Variable? [vt100] ↵

Available in Init Administrative State? [no] ↵
Description? ↵

Ready to install ttys? [no] y ↵
The new ttys have been added.
```

Adding Line Printers

To add a printer, use the **sysadm** Line Printer Management Menu. For details on this menu, consult Chapter 11. If you do not want to add any line printers, you should edit the file **/etc/dgux.params**. You will see the parameter line **lpsched_START="true"**; change "**true**" to "**false**". This will prevent the automatic starting of the **lp** scheduler.

OS clients may access any printer on their local network, provided the system administrator for the remote printer grants access. The administrator of the remote printer grants access by adding the OS client's host name to the **/etc/hosts.equiv** file. Clients then need to get the name of the remote host and the name of the selected printer and use the **sysadm addlp** command and set up the server's printer as a remote printer.

To add a line printer on the server, enter the following:

```
# sysadm addlp ↵
```

An example dialogue may proceed as follows:

```
Running subcommand 'addlp' from menu 'lpmgmt',
LINE PRINTER MANAGEMENT

Sysadm must shut down the lp scheduler while performing
this operation on a printer.  This will interrupt any
requests currently printing.  These requests will be
printed in full when the add operation is complete.  Sysadm
will shut down the scheduler for you at this point.

Stop the scheduler now?  [yes]  ↵
The scheduler has been shut down.

Printer name?  mainlp ↵
Is this a local printer?  [yes]  ↵
Printer model?  [dumb]  ↵

Printer device file?  list ↵
```

```
The available devices are:

tty00 through tty23

Printer device file?  tty00 ↵

mainlp has been added.
Accept and enable mainlp?  [yes]  ↵

mainlp has been enabled.
Restart the scheduler now?  [yes]  ↵

The scheduler has been restarted.
```

Next, we specify **mainlp** as our default printer. We'll call the **defaultlp** function as follows:

```
# sysadm defaultlp ↵
```

The system responds as follows:

```
Running subcommand 'defaultlp' from menu 'lpmgmt',
LINE PRINTER MANAGEMENT

There is no current default.
New default printer?  mainlp ↵

The new default printer is mainlp.
```

Use the **lpstat −t** command to display status information on local and remote printers.

Starting the Accounting System

The DG/UX accounting system is a collection of C language programs and shell procedures with which you can monitor how system resources are being used. Accumulated data is organized and directed into summary files and reports. Note that there is some cost in starting the accounting system; a number of programs start up and begin using disk space. If you are unfamiliar with the DG/UX accounting programs, read Chapter 15.

When you bring the system to a multi-user state (run levels 2 or 3), you can have your default accounting system start up automatically. To do this, edit **/etc/dgux.params**. You will see the parameter line **account_START="false"**. Change "**false**" to "**true**".

End of Phase 3

Phase 4: Adding OS Releases and Clients

You must add a release to the system before you can attach a client to that release. Phase 3 completed the installation of the primary release, so you may now add clients to it. If you have foreign OS clients, you need to add secondary releases for them.

The steps in this phase are:

- Adding Secondary Releases.
- Building Kernels for Diskless Clients.
- Setting OS Client Defaults.
- Adding OS Clients.
- Booting and Setting Up an OS Client.

Step 20: Adding Secondary Releases

OS release software consists of one or more software packages that are loaded into the same directory tree. You use **sysadm addrelease** to create the appropriate directories for a secondary release. You should have already created one or more logical disks for the secondary release. Refer to the plans you made during Step 2. You use **sysadm addrelease** to add the secondary release before loading it.

The following example dialogue shows how you might create a secondary release area for an OS client running SunOS 4.0.

```
# sysadm addrelease ↵

New Release Name? 68k_sunos_4 ↵
Usr Directory?  [/srv/release/68k_sunos_4/usr]  ↵
Share Directory? [/srv/share] ↵
Client Root Parent Directory? [/srv/release/68k_sunos_4]  ↵
Client Swap Directory? [/srv/swap] ↵

Release 68k_sunos_4 has been added. You may now use loadpackage.
```

With the release added, you now load the software with **sysadm loadpackage**. Before attempting to add clients for a foreign release, consult the manuals supplied with the foreign release.

Step 21: Building Kernels for Diskless Clients

On Foreign Systems

Foreign OS clients must boot their own starter systems and build their own kernels. Data General diskless clients supported by foreign servers can use the TFTP bootstrap file **/usr/stand/boot.aviion** and the starter kernel **/usr/stand/dgux.diskless**.

On AViiON Systems

Although OS servers and OS clients can run the same primary release, a client's kernel is slightly different from a server's kernel. Servers boot the starter kernel supplied with the primary release. After the server is up and running, the server must build a kernel for diskless clients.

As the administrator of the server, you may choose to build one kernel for all of your OS clients, or you may choose to build individual kernels for each client. Once the client is up and running, the client user (or administrator) may wish to build his or her own personal kernel.

The advantage of building one kernel for all clients is in the disk space that you save. With a single client kernel, typically kept in **/srv/release/PRIMARY/root/_Kernels** as **dgux.diskless**, all clients whose root space is on the same logical disk as the kernel may create physical links to it (with **ln**(1)) from their root directories. The disadvantage of sharing a kernel like this is one of security: because all kernels have root access to the file, any user with superuser privilege on a client can alter the kernel, effectively changing the kernel that all other clients use as well.

The advantage of making individual kernels for each client is just the reverse: superusers on the clients may change only their own kernels, but not anyone else's. The disadvantage is that you use up more disk space storing an individual kernel for each client. If you are an OS server administrator, you should weigh these advantages and disadvantages before making a decision.

In the following example, we use the **newdgux** command to build a kernel for all the OS clients on an example OS server system. We then link all of the OS clients to the new kernel.

```
# sysadm newdgux ↵

Running subcommand 'newdgux' from menu 'sysmgmt',
SYSTEM CONFIGURATION MANAGEMENT

System Name? [aviion]   diskless ↵
```

By specifying diskless here, **newdgux** will create a kernel named **/dgux.diskless**.

```
Editor?  [vi]  ↵
```

Edit the system file as before, making sure to comment out entries for devices that your clients do not have (like **sd(insc(),*)** and **st(insc(),*)**, for example). If you are not familiar with the **vi editor**, see Step 15 for some editing hints. When you have finished editing the file, exit the editor. Next, you will see the following:

```
Ready to Configure a Kernel? [yes]  ↵

sysadm will now run config on
          /usr/src/uts/aviion/Build/system.diskless

Config succeeded.

sysadm will now attempt to build a kernel.
Building...
The build succeeded.
```

When the build concludes, you can install the new kernel in a location accessible to diskless clients.

```
Install the New Kernel? [no] y↵
For a Diskless Client of this Host? [no]  y↵
Kernel Pathname?
          [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵

The new kernel has been copied to
          /srv/release/PRIMARY/root/_Kernels/dgux.diskless.
Link ALL PRIMARY Clients to the New Kernel? [yes] ↵
```

The next time a client boots, the new kernel will take effect.

Step 22: Setting OS Client Defaults

The **sysadm clientdefaults** function records defaults for the **addclient** function. With **clientdefaults**, you can create sets of defaults to be used for different groups of diskless clients. That is, you might have a set called **dgset** for your Data General client machines running the primary release, and you might have a set called **sunset** for clients running the **68k_sunos_4** release.

The following example **clientdefaults** session shows how we might add a set of defaults for our AViiON systems.

```
# sysadm clientdefaults ↵
```

The system responds as follows:

```
Running subcommand 'clientdefaults' from menu 'clientmgmt',
Client Management

Defaults Set Name? [generic] dgset ↵
Default Release Name? PRIMARY ↵
Default Swap Size? [16m] 24m  ↵
Default Home Directory? [/home] /sales/accounts ↵
Default Kernel? [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵
Default Bootstrap File? [/usr/stand/boot.aviion] ↵
Defaults for Set dgset have been assigned.
```

Customizing the Server and Clients' File System Table (fstab)

NOTE: This is an optional advanced procedure that you may choose to perform if you have many OS clients for whom you want nonstandard file systems to be mounted when the system is booted.

When you add a client with **sysadm addclient**, certain entries are automatically put in the client's **fstab** file. Those are a **/**, **/srv**, **/usr**, **swap**, and a home directory. Sometimes a server administrator may have a list of file systems that he or she wants all clients to mount upon booting. To set this up, the server administrator would edit the **/srv/release/PRIMARY/usr/root.proto/etc/fstab.proto** file. The next time **sysadm addclient** is executed, the edited proto file would be written to the clients' area.

The following example **fstab** files show what you might have for a server named **sales** and an OS client named **dg1**.

Server: **sales** --

```
/dev/dsk/root                                /        dgux
/dev/dsk/usr                                /usr      dgux
/dev/dsk/swap                               swap      swap
/dev/dsk/srv                                /srv      dgux
/dev/dsk/srv_swap                       /srv/swap     dgux
/dev/dsk/usr_opt_X11                 /usr/opt/X11     dgux
/dev/dsk/srv_dgux430       /srv/release/PRIMARY       dgux
/dev/dsk/srv_sunos4    /srv/release/68k_sunos_4       dgux
/dev/dsk/var_tmp                        /var/tmp       dgux
/dev/dsk/sales_accounts         /sales/accounts       dgux
```

Client: **dg1** --

```
sales:/srv/release/PRIMARY/root/dg1          /        nfs
sales:/srv/swap/dg1                       swap        nfs
sales:/usr                                /usr        nfs
sales:/srv/share                      /usr/share      nfs
sales:/usr/opt/X11                 /usr/opt/X11       nfs
sales:/sales/accounts           /sales/accounts       nfs
```

Remember, when you mount **/srv** and **/srv/swap**, do not export them. The **sysadm** program exports subdirectories of these file systems as you add OS clients. If you export these file systems, **sysadm** will not work correctly.

Similarly, you may choose to edit these files to set common parameters for all your clients.

/srv/release/PRIMARY/usr/root.proto/etc/tcpip.params.proto, **/srv/release/PRIMARY/usr/root.proto/etc/nfs.params.proto**, **/srv/release/PRIMARY/usr/root.proto/etc/dgux.params.proto**, and **/srv/release/PRIMARY/usr/root.proto/etc/inittab.proto**.

Step 23: Adding OS Clients

To add diskless clients, you first add entries for those clients to the **/etc/host** and the **/etc/ethers** files or to the appropriate YP database. Do this with **sysadm addhost** and **addether**. The sequence of **sysadm** commands for adding clients is:

1. **addhost**
2. **addether**
3. **addclient**
4. Set up packages on the server or on the client.
5. Boot the client.

Adding Clients to /etc/hosts

The following example shows an **addhost** session where we add an entry for host **dg1**.

```
# sysadm addhost ↵

This host is the YP master.  You must choose between
accessing the global or local list.

Access the Global/Network List? [yes] ↵
Host name? dg1 ↵
Host address? 128.223.2.2 ↵
YP Server? [yes] no ↵
The YP server query is asked only on the master server.
The entry for dg1 has been added.
Do you want to add another host? [no] ↵

Updating the Yellow Pages host and network maps.
```

Adding Clients to /etc/ethers

The following example shows an **addether** session where we add an entry for host **dg1**.

```
# sysadm addether ↵

Host Name? dg1 ↵
Ethernet Address? 08:00:1b:00:a0:17 ↵
The entry for dg1 has been added.
Do you want to add another entry? [n] ↵
```

Adding an Example Client

Adding a client consists of attaching the client to an existing release. This means making a host-specific copy of the **/** file system, and linking a client to the single copy of **/usr**.

The following example shows an **addclient** session where we add client **dg1** to a host named **sales**.

```
# sysadm addclient ↵

Server Host Name? [sales] ↵
Client Host Name? dg1 ↵
Defaults Set Name? [generic] dgset ↵
Use all defaults from dgset? yes ↵
Creating client root.
Creating client swap file.
Creating client /etc/fstab.
Creating client /etc/hosts.
Creating client /tcpip.params.
Creating client /etc/nfs.params.
Client dg1 has been added.
Do you wish to add another client? [yes] no ↵
```

Setting Up Packages for All OS Clients

The first three paragraphs describe the types of tasks you will be performing. Instructions for package setup follow this discussion. Packages contain two types of setup procedures:

- Global facilities used by all OS clients,
- Private facilities used by individual OS clients.

You initiate both types of setup procedures with the command, **sysadm setuppackage**. You will perform the global facilities setup for all clients at the OS server machine. You will perform the private facilities setup for each OS client at each of the OS client machines.

You will also be setting up the TCP/IP, ONC/NFS, and YP packages (plus any optional package you may have loaded in Step 15). Step 6, "Assembling Network Information for Your System," contains information you need to set up these packages. Also, check your resource planning worksheets and those for each OS client for this information.

To initiate the package setup procedures, you type the following command from the OS server machine:

```
# sysadm setuppackage ↵
```

For each package that requires some setup activity, you may be asked if you want to set up the **usr** components (global facilities). Answer **y** (yes).

NOTE: If you are an OS client of the PRIMARY release, you will not be asked this question.

The next question applies to all OS clients. You will be asked if you want to set up the **root** components (private facilities) for each OS client. Answer **n** (no). You will set up **root** in Step 24.

Answer the questions appropriately, as you did for the OS server (see Step 16, "Setting Up software Packages with sysadm"). Normally, you will want to set the OS client host as a YP client. The YP setup procedures assume this default.

Step 24: Booting and Setting Up an OS Client

In this last phase, the client can now obtain a bootable OS image from the server machine. After booting, the server administrator or the client administrator can set up the client machine.

Booting an Example Diskless Client

We're ready to boot **dg1**. This means that the following commands will have to be done on the client machine. The machine should have the following System Control Monitor prompt. Type:

```
SCM> b inen() ↵

Booting inen()
Local Ethernet address is 08:00:1b:00:a0:17
Local Internet address is 128.223.2.2
Trying server at 128.223.2.1 or 80DE0354 hex for TFTP transfer

DG/UX Bootstrap Release 4.30 Version (diskless)

Boot: inen(0)
My name is dg1
My root is sales:/srv/release/PRIMARY/root/dg1

Using 8 Megabytes of physical memory
Found 1 processor(s)
Processor 0 running
INIT: SINGLE USER MODE
```

When the system comes up, the client will have access to those file systems listed in the client's **/etc/fstab**. The client administrator should check that this file contains the desired entries and modify it as necessary.

Setting Up Diskless Clients with sysadm

Perform these steps for each OS client.

- Set your default boot path with the SCM **b** command (refer back to Step 18, "Setting Default Boot Characteristics," for explicit procedures).
- Change your initial run level from 1 (single user mode) to 3 (multiuser mode). Refer back to Step 18 for explicit procedures.
- As prompted when the machine comes up, use **sysadm setuppackage** on the OS client machine to set up the OS client's **root**. Note that OS clients use the same **tcpip.params** setting as the OS server by default.
- Check **/etc/fstab** to see that all needed file systems are listed. Add as necessary and mount them with **sysadm fsmgmt**.

- Add a remote printer with **sysadm addlp**.

Continuing Administration Duties

This concludes the installation information on the DG/UX operating system. The remainder of this chapter contains installation examples for specific system configurations. The remainder of the manual discusses the various tasks that you will need to perform as system administrator.

End of Phase 4

Examples

This section reproduces example installation scenarios for specific system configurations. During installation, you may refer to these examples as guides. However, do not try to use them verbatim when installing your system. The procedures and actions shown in these examples work for the example configuration, but they may not work on your own system. You should read and understand the preceding portion of the chapter before examining these examples.

Example 1: Installing the DG/UX System on an Example AV310C System

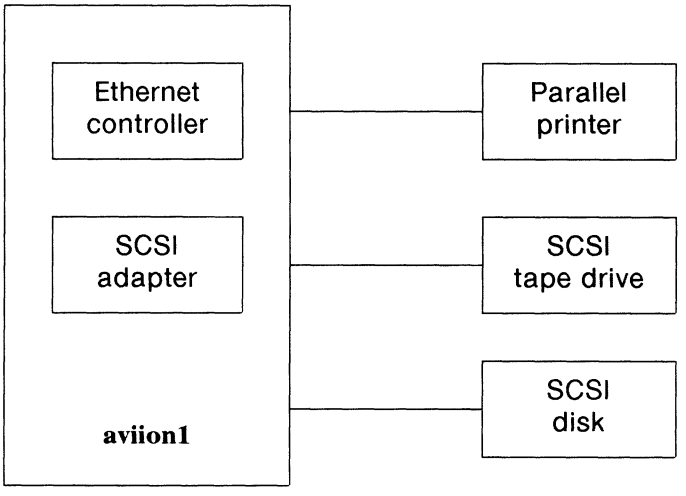
Read the planning and installation procedures in the previous sections of this chapter before reading this section.

This section shows the system/user interaction involved in installing the DG/UX system on an example system. The example system may not be the same as your own system, so do not use this example as a guide during installation.

The example configuration is a color workstation with graphics monitor and parallel printer to run X-based applications and word processing. The components of the system are:

- One AV310C named **aviion1** with 12MBytes of memory.
- One 322MBytes SCSI disk.
- One SCSI QIC-150 tape drive with SCSI adapter.
- One integrated Ethernet controller.
- One parallel printer.

An illustration of the system follows:



Phase 1: Planning the Installation

Step 1: Determining How You Will Use Your System

You have a standalone system and the System Software Package. Specific software products are listed as follows:

Table 2-8 Software Products

Products
DG/UX system, GNU C, DTK, DG/UX man pages
X Windows, Looking Glass, OSF/Motif, man pages
Frame 1.3

You have DG/UX system release 4.30 and the Frame 1.3 package. You will be loading from tape.

Step 2: Identifying Your System's Devices

You have an AV310C system. Your devices follow:

Table 2-9 Device Information for the Example System

Device	Specification	Device No.	SCSI ID No.
Disk controller (boot disk)	**sd(insc(),0)**	NA	0
Network controller	**inen()**	NA	NA
Tape drive	**st(insc(),4)**	NA	4

You have a 6491 SCSI (full height) device, which contains 322 MBytes (or 659456 blocks).

Step 3: Learning About Hosts, Software, Disks, and File Systems

Your host's name is **aviion1**.

Step 4: Allocating Disk Space

We have determined that we will need five logical disks including swap. We will create another logical disk named **udd_aviion1** as the working directory for the user(s) of **aviion1**. The following table lists the logical disk disk-file system plan.

Table 2-10 A Logical Disk-File System Plan

Disk Type	Physical Disk Name	Logical Disk Name	Piece	Mounted File System Name
SCSI	**sd(insc(),0)**	**swap**	1	
		root	1	**/**
		usr	1	**/usr**
		usr_opt_X11	1	**/usr/opt/X11**
		usr_opt_frame1.3	1	**/usr/opt/frame1.3**
		udd_aviion1	1	**/udd/aviion1**

System Logical Disks

Our main system logical disks are

root 40,000 blocks

usr 160,000 blocks

swap 50,000 blocks.

Other Logical Disks

The logical disks we need are

usr_opt_X11 105,000 blocks

usr_opt_frame1.3 30,000 blocks

udd_aviion1 50,000 blocks

Step 5: Understanding the DG/UX Directory Tree

No action is required for this step.

Step 6: Assembling Network Information for Your System

Network information follows:

Table 2-11 Assembled Network Information for the Example System

Host	IP address	Ethernet address	Release
aviion1	128.223.2.1	04:00:1c:00:2a:12	**PRIMARY**

Phase 2: Loading the Primary Release from Tape to Disk

Step 7: Booting the Disk Management (diskman) Utility

```
SCM> b st(insc(),4) ↵

Booting st(insc()4,)

DG/UX Bootstrap Release 4.30

Skipping tape file 1.

================================================================================

DG/UX System Release 4.30, Version Diskman
 Using 12 Megabytes of physical memory
 Found 1 processor(s)
 Processor 0 running


                    DG/UX Starter System
Enter the names of the devices you will use in Common Device Specification
Format, with one name per line.  Enter just newline when done.

Examples:  sd(insc(),0) st(insc(),4) cird st(cisc(),4)
Include duart() for servers and kbd() and grfx() for workstations.

Device name? kbd() ↵
Device name? grfx() ↵
Device name?  ↵
...
```

Step 8: Initializing Physical Disks with diskman

```
            Diskman Main Menu


1.  Physical Disk Management Menu

2.  Logical Disk Management Menu

3.  File System Management Menu

4.  Initial Installation Menu

5.  Update Installation Menu


Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT
Enter choice: 4


================================================================================

           Initial Installation Menu


1. Initialize Physical Disks

2. Create the Root Logical Disk and File System

3. Create the Swap Logical Disk

4. Create the /usr Logical Disk and File System

5. Load the Root File System

6. Load the /usr File System
```

```
7. All Installation Steps

==========================================================================

Enter ? or <number>? for help, ^ to GO BACK,
or q to QUIT.

Enter Choice: [7]


==========================================================================

All Installation Steps

==========================================================================


      1. Initialize Physical Disks

==========================================================================

Do you want to run this step? [y]  ↵

Enter the Physical Disk specification in DG/UX common format: sd(insc(),0) ↵
Install a Disk Label on a Physical Disk
Do you want to run this step? [y]  ↵
Disk label already exists on disk sd(insc(),0).
Do you want to reinstall disk label? [n] y ↵

                Disk Types

1. 6442     ESDI            322MB
2. 6555     ESDI            648MB
3. 6661     ESDI            322MB
4. 6491     SCSI            322MB
5. 6554     SCSI            662MB
6. 6541     SMD            1066MB
7. 6539     SCSI            179MB
8. 6662     SCSI            322MB
9. 6627     OPTICAL SCSI    295MB
10. None of the Above.

Enter the type of disk you have: 4 ↵
Disk label has been installed.

Perform Hardware Formatting on a Physical Disk
Do you want to run this step? [y]  ↵
WARNING:  This operation will DESTROY any data on the Physical
disk sd(insc(),0).

Do you want to continue? [y]  ↵

Create DG/UX System Areas on a Physical Disk
Do you want to run this step? [y]  ↵
WARNING:  This operation will DESTROY any data on the Physical
disk sd(insc(),0).

Do you want to continue? [y]  ↵

The Physical Disk sd(insc(),0) is 628906 blocks in size.
Enter the number of blocks to allocate for the remap Area: [189] ↵
Enter the pathname of the boot.aviion file: [/usr/stand/boot.aviion] ↵

Perform Surface Analysis on a Physical Disk
Do you want to run this step? [y] n ↵

Do you want to format another Physical Disk? [n]  ↵
```

Step 9: Creating System Logical Disks and File Systems

Creating the Root Logical Disk and File System

```
        2. Create the Root Logical Disk and File System

Do you want to run this step? [y]  ↵
Enter the Logical Disk Unit Name: [root]  ↵
Enter the Physical Disk specification in the DG/UX common
format: [sd(insc(),0)]  ↵
The Physical Disk must be registered for this operation.
Do you want to register it? [y]  ↵
Physical disk sd(insc(),0) has been registered.
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of the Logical
Disk Piece: [729]  ↵
Enter the size in blocks of the Logical Disk Piece: [40000]  ↵
The Logical Disk `root' has been created.

Making a file system on the Logical Disk `root' ...

Made a File System on the Logical Disk `root'.

        3. Create the Swap Logical Disk

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [swap]  ↵
Enter the Physical Disk specification in the DG/UX common
format: [sd(insc(),0)]  ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of the Logical
Disk Piece: [40729]  ↵
Enter the size in blocks of the Logical Disk Piece: [50000]  ↵
The Logical Disk 'swap' has been created.
```

Creating the usr Logical Disk and File System

```
        4. Create the /usr Logical Disk and File System

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [usr]  ↵
Logical Disk Piece 1:
Enter Physical Disk specification in DG/UX common
format: [sd(insc(),0)]  ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of Logical Disk
Piece 1: [90729]  ↵
Enter the size in blocks of Logical Disk Piece 1: [160000]  ↵
Do you want to specify any more pieces for this Logical Disk? [n]  ↵
The Logical Disk `usr' has been created.

Making a file system on logical disk `usr' ...

Made a File System on Logical Disk `usr'.
```

Step 10: Loading DG/UX Files onto System Logical Disks

Loading the / File System

```
        5. Load the Root File System

Do you want to run this step? [y]  ↵

Do you want to see the names of the files being loaded? [y]  ↵
Enter the Logical Unit Disk Name: [root]  ↵
```

```
Enter the tape drive specification in DG/UX common format:  [st(insc(),4)]   ↵
Ready to load the Root File System.

Mount the first release tape on the tape drive st(insc(),4).
Press New Line when ready to continue... ↵
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
The Root File System has been loaded.
```

Loading the /usr File System

```
        6. Load the /usr File System

Do you want to run this step? [y]  ↵

Do you want to see the names of the files being loaded? [y]  ↵
Enter the Logical Disk Unit Name: [usr]  ↵
Enter the tape drive specification in DG/UX common format:  [st(insc(),4)]   ↵
Ready to load the /usr File System.

Mount the first release tape on the tape drive st(insc(),4).
Press New Line when ready to continue... ↵
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
The /usr File System has been loaded.

Your starter system has been installed.

Press New Line when ready to continue...  ↵
```

At this point, we quit **diskman** and return to the SCM.

Step 11: Updating the / and /Usr File Systems

This step is not necessary for our example system.

Phase 3: Customizing the Primary Release

Step 12: Booting the Starter Kernel

```
SCM> b sd(insc(),0)root:/dgux.starter ↵

Booting sd(insc(),0)root:/dgux.starter

DG/UX Bootstrap Release 4.30

==========================================================================

DG/UX System Release 4.30, Version (starter)
 Using 12 Megabytes of physical memory
 Found 1 processor(s)
 Processor 0 running
```

Step 13: Specifying Starter Devices

```
                    DG/UX Starter System
Enter the names of the devices you will use in Common Device Specification
Format, with one name per line.  Enter just newline when done.

Examples:  sd(insc(),0) st(insc(),4) cird() st(cisc(),4)
Include duart() for servers and kbd() and grfx() for workstations.

Device name? kbd() ↵
Device name? grfx() ↵
Device name? st(insc(),4) ↵
Device name? sd(insc(),0) ↵
Device name? ↵
Using /dev/dsk/swap as swap file

** root:
No check necessary for root

Mounting /dev/dsk/root as root file system

INIT:  Boot options are: init
INIT:  Cannot open /etc/TIMEZONE. Environment not initialized.

INIT:  /etc/inittab file created from /etc/inittab.proto prototype

INIT:  Checking and mounting /usr...

INIT:  /usr is not mounted

INIT:  SINGLE USER MODE
su:  unable to access /etc/passwd
#
```

Setting Up DG/UX: Initial Configuration

```
# init 1 ↵

INIT: New run level: 1

chk.fsck:

chk.date:
     Current date/time:  Wed Jun 13 08:15 EDT 1990
--   Are the current date, time, and TIMEZONE correct? (y n) [n] : y↵

Setting up package: dgux
```

```
Initializing system database files from .proto files:
initialize /etc/passwd
initialize /etc/group
initialize /etc/dgux.params
.
.
.
Linking /dgux.starter kernel to /dgux
Set permissions on /etc/uucp// 755 root sys

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d directories.
+.....................................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.....................................................
Initializing /usr/root.proto directory
Initializing system database files from the original prototype files
initialize /usr/lib/acct/holidays
Cleaning old uucp directory (/usr/lib/uucp)
NOTE:  Merge your old configuration files from /usr/lib/uucp/*_4.20 with
       the new versions in /etc/uucp.

chk.system:
   Cleanup the /etc/ps_data file and /etc/log files
   Check for missing local passwords

** WARNING:  These local accounts have NO password
   root::0:1:Special Admin login/:/sbin/sh
   sysadm::1:sysadm:Regular Admin Login/admin:/sbin/sh

chk.devlink:
   Add short names (for device notes) to /etc/devlinktab
   .
   .
   .

   Link short names for /dev device notes:
   .
   .
   .

Executing the /etc/rc1.d scripts

Starting rc.tcload: terminal controllers
    /usr/sbin/tcload -a

Starting rc.update: update daemon
    update

Starting rc.localfs: local mounts
    mount -at dg/ux

    The following file systems are now mounted:

    /dev/dsk/root on / type dg/ux (rw)
    /dev/dsk/usr on /usr type dg/ux (rw)

Starting rc.setup:     check for packages that haven't been set up.
    All packages are set up.

Press <RETURN> to display prompt. ♪

no_node
DG/UX Release 4.30
login: sysadm ♪
```

Step 14: Creating Other Logical Disks and File Systems

Creating the usr_opt_X11 Logical Disk

```
# sysadm diskmgmt ↵

Running subcommand 'diskmgmt' from menu 'menu',
SYSADM MAIN MENU

                        Diskman Main Menu


1.  Physical Disk Management Menu

2.  Logical Disk Management Menu

3.  File System Management Menu

4.  Initial Installation Menu

5.  Update Installation Menu

==========================================================================

Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT
Enter choice: 2 ↵
```

In the **diskman** main menu, we select option 2, the Logical Disk Management Menu, then option 1, Create a Logical Disk.

```
Enter the Logical Disk Name: usr_opt_X11 ↵
Logical Disk Piece 1:
Enter the Physical Disk specification in DG/UX common format: sd(insc(),0) ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of Logical
Disk Piece 1: [250729]  ↵
Enter the size in blocks of Logical Disk Piece 1: [378177] 105000 ↵
Do you want to specify any more Pieces for this Logical Disk? [n]  ↵
The Logical Disk `usr_opt_X11' has been created.
Do you want to make a file system on this Logical Disk? [y]  ↵
No additional information is required, but you may specify mkfs
flags and options if you wish.

Enter the flags and options you want to specify:  ↵

Making a file system on Logical Disk `usr_opt_X11' ...

Made a File System on the Logical Disk `usr_opt_X11'

Press New Line when ready to continue... ↵
```

Creating the usr_opt_frame1.3 Logical Disk

```
Enter the Logical Disk Name: usr_opt_frame1.3 ↵
Logical Disk Piece 1:
Enter the Physical Disk specification in DG/UX common format: sd(insc(),0) ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of Logical
Disk Piece 1: [355729]  ↵
Enter the size in blocks of Logical Disk Piece 1: [273177] 30000 ↵
Do you want to specify any more Pieces for this Logical Disk? [n]  ↵
The Logical Disk `usr_opt_frame1.3' has been created.
Do you want to make a file system on this Logical Disk? [y]  ↵
No additional information is required, but you may specify mkfs
flags and options if you wish.

Enter the flags and options you want to specify:  ↵
```

```
Making a file system on Logical Disk `usr_opt_frame1.3' ...

Made a File System on the Logical Disk `usr_opt_frame1.3'

Press New Line when ready to continue... ↵
```

Creating the udd_aviion1 Logical Disk

```
Enter the Logical Disk Name: udd_aviion1 ↵

Logical Disk Piece 1:
Enter the Physical Disk specification in DG/UX common format: sd(insc(),0) ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of Logical
Disk Piece 1: [385729]  ↵
Enter the size in blocks of Logical Disk Piece 1: [243177] 50000 ↵
Do you want to specify any more Pieces for this Logical Disk? [n]  ↵
The Logical Disk `udd_aviion1' has been created.
Do you want to make a file system on this Logical Disk? [y]  ↵
No additional information is required, but you may specify mkfs
flags and options if you wish.

Enter the flags and options you want to specify:  ↵

Making a file system on Logical Disk `udd_aviion1' ...

Made a File System on the Logical Disk `udd_aviion1'

Press New Line when ready to continue... ↵
```

We then exit **diskman**.

Adding the /usr/opt/X11 File System to /etc/fstab with sysadm addfsys

```
# sysadm addfsys ↵

Running subcommand 'addfsys' from menu 'fsmgmt',
FILE SYSTEM MANAGEMENT

Mount Directory Name? /usr/opt/X11 ↵
Is this a local file system? [yes]  ↵
Writeable? [y]  ↵
Dump Cycle? [d]  ↵
fsck Pass? [1]  ↵
Export? [no]  ↵

The entry for /usr/opt/X11 has been added.

The directory, /usr/opt/X11, does not exist.
Create /usr/opt/X11? [yes]  ↵
Mount the file system? [yes]  ↵

The file system has been mounted.
#
```

Adding the /usr/opt/frame1.3 File System to /etc/fstab with sysadm addfsys

```
# sysadm addfsys ↵

Running subcommand 'addfsys' from menu 'fsmgmt',
FILE SYSTEM MANAGEMENT

Mount Directory Name? /usr/opt/frame1.3 ↵
Is this a local file system? [yes]  ↵
Writeable? [y]  ↵
Dump Cycle? [d]  ↵
fsck Pass? [1]  ↵
Export? [no]  ↵
```

```
The entry for /usr/opt/frame1.3 has been added.

The directory, /usr/opt/frame1.3, does not exist.
Create /usr/opt/frame1.3? [yes]  ↵
Mount the file system? [yes]  ↵

The file system has been mounted.
#
```

Adding the /udd/aviion1 File System to /etc/fstab with sysadm addfsys

```
# sysadm addfsys ↵

Running subcommand 'addfsys' from menu 'fsmgmt',
FILE SYSTEM MANAGEMENT

Mount Directory Name? /udd/aviion1 ↵
Is this a local file system? [yes]  ↵
Writeable? [y]  ↵
Dump Cycle? [d]  ↵
fsck Pass? [1]  ↵
Export? [no]  ↵

The entry for /udd/aviion1 has been added.

The directory, /udd/opt/aviion1, does not exist.
Create /udd/aviion1? [yes]  ↵
Mount the file system? [yes]  ↵

The file system has been mounted.
#
```

Step 15: Loading Software Packages with sysadm

```
# sysadm makesrv ↵

Running subcommand `makesrv' from menu `releasemgmt',
Software Release Management

Making the PRIMARY release area.
Making the MY_HOST client entry.
makesrv is finished
# sysadm loadpackage ↵

Running subcommand 'loadpackage' from menu 'releasemgmt',
Software Release Management

Release Area? [PRIMARY]  ↵
Tape Drive? [0]  ↵
Is the tape mounted and ready? y ↵
Load Package X11.lg? [yes]  ↵
Load Package X11.man? [yes]  ↵
Load Package X11? [yes]  ↵
Load Package dgux.man? [yes]  ↵
Load Package dtk.man? [yes]  ↵
Load Package dtk? [yes]  ↵
Load Package gcc.man? [yes]  ↵
Load Package gcc? [yes]  ↵
Load Package nfs.man? [yes]  ↵
Load Package nfs? [yes]  ↵
Load Package tcpip.man? [yes]  ↵
Load Package tcpip? [yes]  ↵
List file names while loading? [yes]  ↵
Mount Volume 1.
Is the tape mounted and ready? y ↵
Skipping tape file 0 to 40.
```

```
.
.
.
Updating proto root (/usr/root.proto).
Updating MY_HOST root (/srv/release/PRIMARY/root/MY_HOST).
loadpackage is finished.
```

Next we execute **loadpackage** for the Frame package.

```
# sysadm loadpackage ↵

Release Area? [PRIMARY]  ↵
Tape Drive? [0]  ↵
Is the tape mounted and ready? y↵
Load Package frame1.3? [yes]  ↵
List file names while loading? [yes]  n ↵
Mount Volume 1.
Is the tape mounted and ready? y↵
loadpackage is finished.
#
```

Step 16: Setting Up Software Packages with sysadm

Next, we execute **setuppackage**:

```
# sysadm setuppackage ↵

Running subcommand `setuppackage' from menu `releasemgmt',
Software Release Management

Release Name? [PRIMARY]  ↵

The following packages have setup scripts that have not been run:

X11       nfs       tcpip          yp
X11.lg
```

Setting Up TCP/IP

```
Package Name? [all] tcpip ↵

Processing setup scripts for package tcpip.
Set up package tcpip in usr? [yes]  ↵

            Setting up package: tcpip

In revisions of the DG/UX operating system before 4.00,
the restricted shell command was named restsh
and the remote shell command was named rsh.
To be compatible with the System V Interface Definition (SVID),
the restricted shell command must be named rsh and
the remote shell command must have a different name.
To be SVID-compliant, Data General names the remote shell remsh.

You are prompted to choose whether or not the names of
the remote and restricted shells comply with the SVID.


If You Choose      The Result Is

    y              The restricted shell is named /bin/rsh
                   The remote shell is named /usr/bin/remsh

    n (default)    The restricted shell is named /bin/restsh
                   The remote shell is named /usr/bin/rsh.
Do you want names to comply with the System V Interface Definition? [n] y ↵
```

```
Restricted Shell is named /bin/rsh
Remote Shell is named /usr/bin/remsh

Remote Commands Installation Complete

Press NEWLINE when ready to continue...  ↵
Setup package tcpip in MY_HOST root? [yes]  ↵

            Setting up package: tcpip

Creating links for initialization scripts...Please Wait

File: /srv/release/PRIMARY/root/MY_HOST/etc/hosts has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/networks has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/services has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/protocols has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/ethers has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params has been created from prototype file.

Press NEWLINE when ready to continue... ↵

Do you want support for loop interface? [y]  ↵

Updating /srv/release/PRIMARY/root/MY_HOST/etc/hosts and
/srv/release/PRIMARY/root/MY_HOST/etc/networks files...Please Wait.

NOTE: Any entries encountered containing conflicting information
      will be deleted from the offending file.

The following lines have been removed from file
"/srv/release/PRIMARY/root/MY_HOST/etc/hosts"
-- Begin Remove List --
127.0.0.1        localhost
-- End of Remove List --

The entry "127.0.0.1  localhost" has been added
to file "srv/release/PRIMARY/root/MY_HOST/etc/hosts"

Updating "/srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params"
...Please wait...

IMPORTANT NOTE: You MUST have a "loop" entry specified in
your system configuration file.  Consult the help menu or the
system(4) man page for more information.

Local Loopback Environment Installation Complete

Press NEWLINE when ready to continue... ↵

The following queries refer to the host being installed

Enter host Internet address: 128.223.75.10 ↵
[128.223.75.10] Correct ? [y] ↵

Enter host name: aviion1 ↵
[aviion1] Correct ? [y] ↵

Enter network name: sales_net ↵
[sales_net] Correct ? [y] ↵

Is "sales_net" a subnetted network? [n] y ↵

Enter the network mask: 0xffffff00 ↵
[0xffffff00] Correct ? [y] ↵

Calculating network address...please wait...

Updating /srv/release/PRIMARY/root/MY_HOST/etc/hosts and
/srv/release/PRIMARY/root/MY_HOST/etc/networks files...please wait
```

```
NOTE: Any entries encountered containing conflicting information
      will be deleted from the offending file.

The entry "128.223.75.10    aviion1" has been added to
"/srv/release/PRIMARY/root/MY_HOST/etc/hosts"
The entry "128.223.75    sales_net" has been added to
"/srv/release/PRIMARY/root/MY_HOST/etc/networks"

Enter controller device name: inen0 ↵
[inen0] Correct ? [y] ↵

There are two variations of Broadcast addresses.  A BSD 4.2
compatible broadcast address has a host portion of all
zeros.  A BSD 4.3 compatible broadcast address has a host
portion of all ones.

Calculating network portion of broadcast address...please wait...

Do you want the host portion of the broadcast address to be all ones? [y]  ↵

Calculating broadcast address...please wait...

Updating /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params...
please wait...

IMPORTANT NOTE: You MUST have a "inen" entry specified in
your system configuration file.  Consult the help menu or the
system(4) man page for more information.

Local Environment Installation Complete.

Press NEWLINE when ready to continue. ↵

The following queries refer to IXE configuration.

Would you like to configure any IXE interfaces? [n]  ↵

IXE Configuration Complete

Press NEWLINE when ready to continue. ↵

Would you like to add a remote host entry? [y]  ↵

The following refers to other hosts on this network

Enter host Internet addres: 128.223.33.1 ↵

Enter host name: goober ↵

The entry "128.223.33.1    goober" has been added to the file
/srv/release/PRIMARY/root/MY_HOST/etc/hosts.

Do you want to add another remote host entry? [n]  ↵

Do you want to edit the
/srv/release/PRIMARY/root/MY_HOST/etc/protocols file? [n]  ↵

Press NEWLINE when ready to continue. ↵

Do you want to edit the
srv/release/PRIMARY/root/MY_HOST/etc/services file? [n]  ↵

Network Environment Installation Complete

Press NEWLINE when ready to continue. ↵

Enter FTP login directory [/var/ftp]:  ↵
[/var/ftp] Correct ? [y]  ↵

Modifying ftp password entry in
```

```
/srv/release/PRIMARY/root/MY_HOST/etc/passwd

Directory: /var/ftp exists
Directory: /var/ftp/bin exists
Directory: /var/ftp/etc exists
File "/usr/bin/ls" has been copied to "/var/ftp/bin/ls"
File "/usr/bin/pwd" has been copied to "/var/ftp/bin/pwd"
File "/srv/release/PRIMARY/root/MY_HOST/etc/group" has been
copied to "/var/ftp/etc/group"

FTP Installation Complete

Press NEWLINE when ready to continue.  ↵

File: /srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv has been
created from prototype file

Warning:  The following query may produce a security breach in your
system.  An entry in the
/srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv file allows a
user from the specified remote host having the same user name to remotely
login to your host WITHOUT having to enter a password.  Caution should
be exercised when adding entries to this file.

Do you wish to add a host to the
/srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv file? [n]  ↵
File "/srv/release/PRIMARY/root/MY_HOST/etc/pmterrtab" created from
prototype.
File "/srv/release/PRIMARY/root/MY_HOST/etc/pmttapetab" created from
prototype.

Remote Commands Installation Complete

Press NEWLINE when ready to continue.  ↵

"/srv/release/PRIMARY/root/MY_HOST/etc/sendmail.cf" created from
"/srv/release/PRIMARY/root/MY_HOST/etc/arpaproto.cf"

Do you need to customize ruleset 0? [n]  ↵

Modifying mail passwd entry in
/srv/release/PRIMARY/root/MY_HOST/etc/passwd.

Do you want to use sendmail as the mailx router? [y]  ↵

File "/srv/release/PRIMARY/root/MY_HOST/var/mailx/mailx.rc has
been created.

The entry "set sendmail=/usr/lib/sendmail" has been added to file
"/srv/release/PRIMARY/root/MY_HOST/var/mailx/mailx.rc"

File "/srv/release/PRIMARY/root/MY_HOST/etc/aliases" created from
prototype file.

Do you want to edit the
/srv/release/PRIMARY/root/MY_HOST/etc/aliases file? [n]  ↵

Executing /usr/bin/newaliases...please wait

3 aliases, longest 11 bytes, 53 bytes total

Sendmail Installation Complete

Press NEWLINE when ready to continue...  ↵

The Domain Name System provides a means to distribute
management of host information.  It can be used in
place of or in conjunction with Yellow Pages and/or
the /etc/hosts file.
```

```
To install and run the domain name server on your
machine you must have data bases set up for the name
server.  Chapter 5 of Setting Up and Managing DG/UX
TCP/IP explains in detail the domain name system and
the requirements to run this service.  Please read
this chapter before attempting to set up the domain
name service on your system.

The answers to the following questions will be used
to partially configure your system for domain name
service access.  The only files that will be edited
are /etc/resolv.conf, /etc/named.boot, and
/etc/svcorder.  If you do not want to edit these file
at this time, answer no to the first question.

Do you want to partially configure for domain name service? [n]  ↵

Partial Domain Name Server Installation Complete

Press NEWLINE when ready to continue...  ↵
Deleting obsolete files...Please wait...

setuppackage is finished
#
```

Setting Up ONC/NFS

```
# sysadm setuppackage ↵

Running subcommand `setuppackage' from menu `releasemgmt',
Software Release Management

Release Name? [PRIMARY]  ↵

The following packages have setup scripts that have not been run:

      X11       X11.lg      nfs       yp

Package Name? [all] nfs ↵
Processing setup scripts for package nfs.
Set up package nfs in usr? [yes] ↵

             Setting up package: nfs

Set up package nfs in MY_HOST root? [yes] ↵

             Setting up package: nfs

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d
+.....................................................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.....................................................................

That completes the automated portion of the NFS configuration

setuppackage is finished.
#
```

Setting Up YP

```
# sysadm setuppackage ↵

Running subcommand `setuppackage' from menu `releasemgmt',
Software Release Management

Release Name? [PRIMARY]  ↵

The following packages have setup scripts that have not been run:
```

```
      X11       X11.lg      yp

Package Name? [all] yp ↵

Processing setup scripts for package yp.

Set up package yp in usr? [yes] ↵

             Setting up package: yp

Set up package yp in MY_HOST root? [yes] ↵

             Setting up package: yp

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d
+.....................................................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.....................................................................

Enter the name of the YP domainname []: sales_domain ↵

---- This host will first run as a YP client.
---- Setting YP domain to:  sales_domain

Is the domainname correct? (y n) [n]: y ↵

   That completes the YP setup for a YP client
-- To initiate YP services you will have to change to init level 3.

-- To complete the YP setup as a YP server or master, please
   refer to the ONC/NFS release notice for this release.

setuppackage is finished
#
```

Setting Up X Windows

```
# sysadm setuppackage ↵

Running subcommand `setuppackage' from menu `releasemgmt',
Software Release Management

Release Name? [PRIMARY]  ↵

The following packages have setup scripts that have not been run:

         X11       X11.lg

Package Name? [all] X11 ↵

Processing setup scripts for package X11.
Set up package X11 in usr? [yes]  ↵

              Setting up package: X11

   Linking /usr/opt/X11/catman/M_man to /usr/catman/M_man
   Linking /usr/opt/X11/include/Xm to /usr/include/Xm
   Linking /usr/opt/X11/include/Mrm to /usr/include/Mrm
   Linking /usr/opt/X11/include/Uil to /usr/include/Uil

    Linking /usr/opt/X11 and /usr

setuppackage is finished
#
```

Setting Up Looking Glass

```
# sysadm setuppackage ↵

Running subcommand `setuppackage' from menu `releasemgmt',
Software Release Management

Release Name? [PRIMARY]  ↵

The following packages have setup scripts that have not been run:

         X11.lg

Package Name? [all] X11.lg ↵

Processing setup scripts for package X11.lg.
Set up package X11.lg in usr? [yes]  ↵

              Setting up package: X11.lg

Installing Looking Glass executable files ...
lg
lg_pause
vice
vls
vls_add
vls_del

Installing Looking Glass manual page ...
Set up package X11.lg in MY_HOST root? [yes]  ↵

              Setting up package: X11.lg

done

setuppackage is finished
#
```

Setting Up Other Software Package

See the appropriate product release notice and installation guides to setup other software packages.

Step 17: Building a Custom Kernel

```
# sysadm newdgux ↵

Running subcommand 'newdgux' from menu 'sysmgmt',
SYSTEM CONFIGURATION MANAGEMENT

System Name? [aviion]  ↵

System File /usr/src/uts/aviion/Build/system.aviion does not exist.
Create the system file? [yes]  ↵
Editor? [vi]  ↵

#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.dgux.proto     94.5"


#-----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
```

```
#
# (Product Name):       DG/UX
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for several typical configurations
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#
#

##### Typical AViiON 300 series workstation configuration:

# Note that your system can have a second duart() or an lp() controller,
# but not both!

    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)

    ptc()               # -- pseudo-terminal controller device
    pts()               # -- pseudo-terminal slave device
    pmt()               # -- pseudo-magtape device
    log()               # -- Streams logger pseudo-device
    prf()               # -- profiler pseudo-device

##### Typical AViiON 400 series workstation configuration:

#    kbd()              # -- keyboard
#    grfx()             # -- graphics display
#    sd(insc(),*)       # -- all SCSI disk drives on integrated SCSI adapter
#    st(insc(),*)       # -- all SCSI tape drives on integrated SCSI adapter
#    inen()             # -- integrated Ethernet controller
#    duart()            # -- integrated Duart terminal line controller
#    duart(1)           # -- second Duart
#    lp()               # -- integrated line printer controller
#
#    ptc()              # -- pseudo-terminal controller device
#    pts()              # -- pseudo-terminal slave device
#    pmt()              # -- pseudo-magtape device
#    log()              # -- Streams logger pseudo-device
#    prf()              # -- profiler pseudo-device


##### Typical AViiON 4000 series server configuration:

#    sd(insc(),*)       # -- all SCSI disk drives on integrated SCSI adapter
#    st(insc(),*)       # -- all SCSI tape drives on integrated SCSI adapter
#    sd(cisc(),*)       # -- all SCSI disk drives on Ciprico SCSI adapter
```

```
#     st(cisc(),*)        # -- all SCSI tape drives on Ciprico SCSI adapter
#     cird()              # -- Ciprico Rimfire or SMD disk controller
#
#     inen()              # -- integrated Ethernet controller
#     hken()              # -- Interphase VME Ethernet controller
#     syac()              # -- Systech terminal line controller
#     duart()             # -- integrated Duart terminal line controller
#     duart(1)            # -- second Duart
#     lp()                # -- integrated line printer controller
#
#     ptc()               # -- pseudo-terminal controller device
#     pts()               # -- pseudo-terminal slave device
#     pmt()               # -- pseudo-magtape device
#     log()               # -- Streams logger pseudo-device
#     prf()               # -- profiler pseudo-device


##### Typical AViiON 5000 or 6000 series server configuration:

#     cird()              # -- Ciprico Rimfire or SMD disk controller
#     sd(cisc(),*)        # -- all SCSI disk drives on Ciprico SCSI adapter
#     st(cisc(),*)        # -- all SCSI tape drives on Ciprico SCSI adapter
#     syac()              # -- Systech terminal line controller
#     duart()             # -- integrated Duart terminal line controller
#     hken(0)             # -- 1st Interphase VME Ethernet controller
#     hken(1)             # -- 2nd Interphase VME Ethernet controller
#     lp()                # -- integrated line printer controller
#
#     ptc()               # -- pseudo-terminal controller device
#     pts()               # -- pseudo-terminal slave device
#     pmt()               # -- pseudo-magtape device
#     log()               # -- Streams logger pseudo-device
#     prf()               # -- profiler pseudo-device
#
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You should not have to specify any additional protocols in order to
# use this product.
#
#
#   Protocol Name
#   -------------
#

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# It is recommended that you specify the Transport Provider Interface
# STREAMS modules, timod and tirdwr.
#
#
```

```
#    STREAMS Module Name
#    -------------------
#
     timod
     tirdwr
#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# The default values from the master file will be used unless
# explicitly overridden in this file.
#
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# You should set the TZ variable to accurately reflect your timezone
# (300 minutes west of GMT is USA Eastern time).
#
# You should set the MAXUP variable to the maximum number of processes
# that each user will be allowed to run simultaneously.  This number
# should be at least 64 for workstations.
#
# You should set the NODE variable to control your nodename for uname(1)
# and uucp(1), but not more than 255 characters.
#
# You should set the DUMP variable to the name of the tape device (in
# DG/UX Common Device Specification Format) that will be the default
# device to take dumps in case of system emergencies.  For diskless
# workstations, the DUMP variable should be set to the network device
# used to boot the machine.
#
# If your system is a diskless workstation, you should set the
# PERCENTNFS variable to 100 in order to get the best possible NFS
# performance.
#
# If either your system's root file system or its swap file will be
# mounted over NFS (a diskless workstation will NFS-mount both, a
# dataless workstation will NFS-mount only the root), you must set
# the NETBOOTDEV variable to the name of the network device (in DG/UX
# Common Device Specification Format) that will be used in booting
# over the network.
#
# If your system's root file system will be mounted over NFS (as will
# be done on both diskless and dataless workstations), you must set the
# ROOTFSTYPE variable to NETWORK_ROOT.
#
# If your system's swap file will be mounted over NFS (as will be done
# on diskless workstations), you must set the SWAPDEVTYPE variable to
# NETWORK_SWAP.
#
#
#    Parameter Name              Value
#    --------------              -----
#
     TZ                          300
     MAXUP                       64
     NODE                        "aviion1"

     DUMP                        "st(insc(),4)"
### DUMP                        "inen()"

### PERCENTNFS                  100
### NETBOOTDEV                  "inen()"
```

```
### ROOTFSTYPE                  NETWORK_ROOT
### SWAPDEVTYPE                 NETWORK_SWAP

#
#-----------------------------------------------------------------
#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.nfs.proto      94.2"


#-----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       NFS
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Devices:
#
# List all devices in this section, one entry per line.
# The string is the name of the device.
# Note that some pseudo-devices have no device code at
# all, so none should be listed.
# Any other text on a line will be ignored.
#
#
#   Device Name
#   -----------
#

    plm()               # -- network lock manager pseudo-device

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You will not need to specify any additional protocols to use this
# product.
#
#
#   Protocol Name
#   -------------
#

#
#-----------------------------------------------------------------
```

```
#--------------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# You will not need to specify any additional STREAMS modules
# to use this product.
#
#
#   STREAMS Module Name
#   -------------------
#

#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Tuneable Configuration Parameters:

# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.

# To use NFS, you must specify the NFS variable so that its implied
# value will be used.


#   Parameter Name              Value
#   --------------              -----

    NFS

#--------------------------------------------------------------------
#       Copyright (C) Data General Corporation, 1985 - 1989.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K    tcpip   90.1"


#--------------------------------------------------------------------
#
# Prototype fragment of system configuration for:

# (Product Name):       TCP/IP
# (Release):            4.30


# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.

#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Devices:
```

```
# List all devices and pseudo-devices in this section, one entry per
# line.  Verify typical configurations for both workstations and
# server systems. You will need at least one LAN controller
# (inen or hken). (see the DG/UX system.proto file for these)



# The protocol engines are Streams multiplexing drivers

        ip()
        tcp()
        udp()

# It is also recommended that you include the loopback pseudo-device.


    loop()


#-----------------------------------------------------------------

#-----------------------------------------------------------------
# Protocols:

# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.

# You will need the tcp, ip, udp and icmp protocols.


#   Protocol Name
#   -------------


        ipproto_ip
        ipproto_tcp
        ipproto_udp
        ipproto_icmp


#-----------------------------------------------------------------


#-----------------------------------------------------------------
# STREAMS Modules:

# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.



#   STREAMS Module Name
#   -------------------

        ether
        arp
        socsys
        netlog


#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Tuneable Configuration Parameters:

# List all configuration parameters you wish to override in this
```

```
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
:wq ↵
```

Installing the New Kernel

```
Ready to Configure a Kernel? [yes]  ↵

sysadm will now run config on /usr/src/uts/aviion/Build/system.aviion

Config succeeded.

sysadm will now attempt to build a kernel.
Building...
The build succeeded.

Install the New Kernel? [no] y↵
For a Diskless Client of this Host? [no]  ↵
Kernel Pathname? [/dgux.aviion]  ↵

The new kernel has been copied to /dgux.aviion.
Link /dgux to the New Kernel? [yes]  ↵

The new kernel will not take effect until you shutdown and reboot.
To do this, quit sysadm, and say:

  cd /
  /etc/shutdown
  /etc/halt -q

Until you do this, a few commands which depend on the symbol table
in /dgux (such as the kernel profiler and netstat) may not work correctly.
This should not cause any serious difficulties.

#
```

Bringing Down the System

```
# cd / ↵
# /etc/shutdown -g0 -y ↵
  ...
# halt -q ↵
```

Step 18: Setting Default Boot Characteristics

```
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 menu configuration

6. Change testing parameters

7. Return to previous screen
```

```
Enter choice(s) -> 1 ↵


Change boot parameters

1 Change system boot path

2 Change diagnostic boot path

3 Change data transfer mode [BLOCK]

4 Return to previous screen

Enter choice(s) -> 1 ↵


System boot path  = []

Do you want to modify the boot path? [N] y ↵

Enter new system boot path -> sd(insc(),0)root:/dgux ↵

System boot path = [sd(insc(),0)root:/dgux]
Do you want to modify the boot path? [N] ↵

Do you want to boot? [N] n ↵
```

Rebooting the System

```
SCM> b ↵
```

Bring the System up to Run Level 1

```
# init 1 ↵
```

Changing the Default Initial Run Level

We change the default initial run level by editing the **/etc/inittab** file and changing this line:

```
def:s:initdefault:
```

To this line:

```
def:3:initdefault:
```

Step 19: Starting System Administration

Adding Groups

Following Chapter 14, we add users in this step.

Adding User Accounts

Following Chapter 14, we add groups in this step, making sure that their home directories are in **/udd/aviion1**.

Setting Up Terminals

We do not set up terminals in this example.

Starting the Accounting System

Following Chapter 15, we set up accounting in this step.

Adding Lineprinters

Following Chapter 11, we add our printers in this step.

Bring the system up to multi-user mode

At this point we can bring our system up to run level 3.

```
# init 3 ↵
```

Phase 4: Adding OS Releases and Clients

This phase is not necessary for our example.

Example 2: Installing the DG/UX System on an Example AV400 System

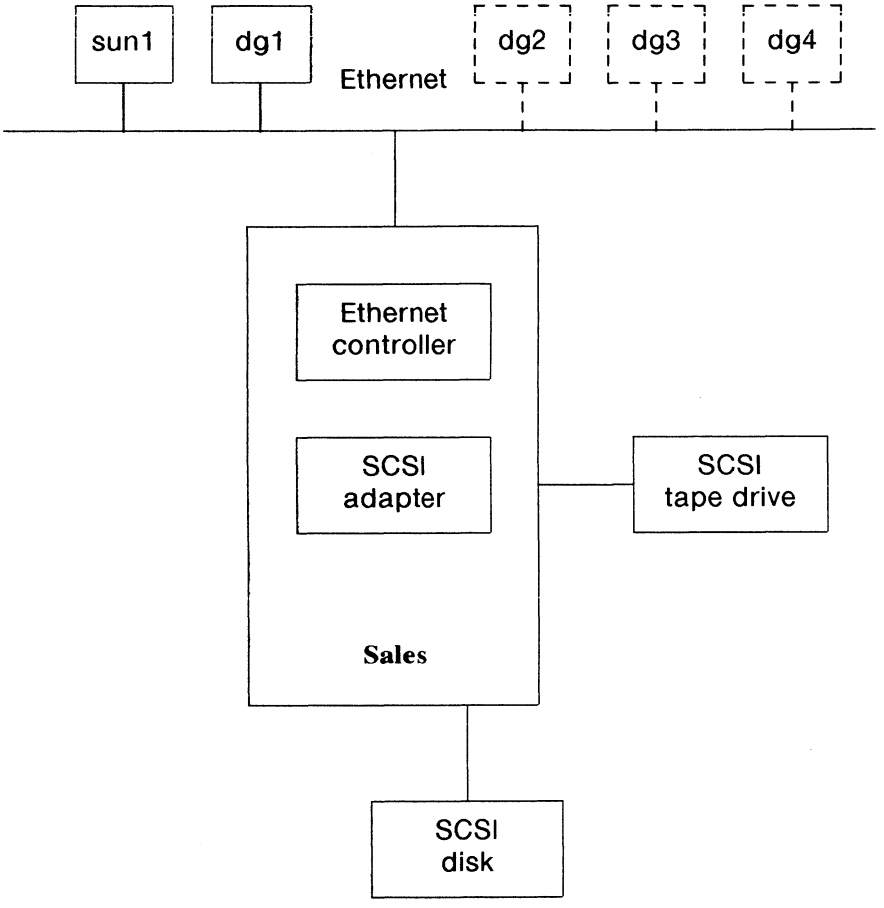
Read the planning and installation procedures in Chapter 2 before reading this section.

This section shows the system/user interaction involved in installing the DG/UX system on an example system. The example system may not be the same as your own system, so do not use this example as a guide during installation.

The example configuration is a workstation with graphics monitor. It will be an OS server for two clients, with plans to add three more in the future.

- One AViiON AV400 named **sales** with 16MBytes of memory.
- One 662MBytes SCSI disk.
- One SCSI QIC-150 tape drive with SCSI adapter.
- One integrated Ethernet controller.
- One AViiON AV300 workstation acting as an OS client; plans to add three more in the future.
- One foreign workstation acting as an OS client.

An illustration of the system follows.



Phase 1: Planning the Installation

Step 1: Determining How You Will Use Your System

You have a standalone system and the System Software Package. Specific products follow:

Table 2-12 Software Products

Products
DG/UX system, Gnu C, DTK
X11, OSF/Motif, Looking Glass, man pages
DG/UX system for OS clients

You have a primary DG/UX system release 4.30 and a foreign release called **68k_sunos_4** for release 4.0 of the SunOS system. You will be loading from tape.

Step 2: Identifying Your System's Devices

You have an AV400 as an OS server, an AV300 as an OS client, and a foreign workstation as another OS client. Your devices follow:

Table 2-13 Device Information for the Example System

Device	Specification	Device No.	SCSI ID No.
Disk controller (boot disk)	**sd(insc(),0)**	NA	0
Network controller	**inen()**	NA	NA
Tape drive	**st(insc(),4)**	NA	4

You have a 6554 SCSI (full-height) device, which contains 662 MBytes (or 1355776 blocks).

Step 3: Learning About Hosts, Software, Disks, and File Systems

Your host's name is **sales**.

Step 4: Allocating Disk Space

Your disk useage plans follow:

Table 2-14 A Logical Disk-File System Plan

Disk Type	Physical Disk Name	Logical Disk Name	Piece	Mounted File System Name
SCSI	**sd(insc(),0)**	**swap**	1	
		root	1	**/**
		usr	1	**/usr**
		usr_opt_X11	1	**/usr/opt/X11**
		sales_users	1	**/sales/users**
		sales_bin	1	**/sales/bin**
		sales_data	1	**/sales/data**
		srv	1	**/srv**
		srv_swap	1	**/srv/swap**
		srv_dgux430	1	**/srv/release/PRIMARY**
		srv_sunos4	1	**/srv/release/68k_sunos_4**
		tmp	1	**/var/tmp**

System Logical Disks

Our main system logical disks are:

root 40,000 blocks

usr 160,000 blocks

swap 50,000 blocks.

Other Logical Disks

The logical disks we need are:

usr_opt_X11 105,000 blocks

sales_users 100,000 blocks.

sales_bin 50,000 blocks.

sales_data 200,000 blocks.

srv 2,000 blocks.

srv_swap 154,000 blocks.

srv_dgux430 132,000 blocks.

srv_sunos4 90,000 blocks.

tmp 30,000 blocks.

Step 5: Understanding the DG/UX Directory Tree

No action is required for this step.

Step 6: Assembling Network Information for Your System

Network information follows:

Table 2-15 Assembled Network Information for the Example System

Host	IP address	Ethernet address	Release
dg1	128.223.10.74	08:00:1b:32:a8:04	**PRIMARY**
sun1	128.223.10.73	02:00:9c:f0:22:12	**68k_sunos_4**

Phase 2: Loading the Primary Release from Tape to Disk

Step 7: Booting the Disk Management (diskman) Utility

```
SCM> b st(insc(),4) ↵

Booting st(insc()4,)

DG/UX Bootstrap Release 4.30

Skipping tape file 1.

================================================================================

DG/UX System Release 4.30, Version Diskman
 Using 16 Megabytes of physical memory
 Found 1 processor(s)
 Processor 0 running


                    DG/UX Starter System

Enter the names of the devices you will use in Common Device Specification
Format, with one name per line.  Enter just newline when done.

Examples:  sd(insc(),0) st(insc(),4) cird() st(cisc(),4)

Include duart() for servers and kbd() and grfx() for workstations.

Device name? kbd() ↵
Device name? grfx() ↵
Device name?  ↵
...
```

Step 8: Initializing Physical Disks with diskman

```
              Diskman Main Menu


1.  Physical Disk Management Menu

2.  Logical Disk Management Menu
```

```
3.  File System Management Menu

4.  Initial Installation Menu

5.  Update Installation Menu

Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT
Enter choice: 4

========================================================================

           Initial Installation Menu

1. Initialize Physical Disks

2. Create the Root Logical Disk and File System

3. Create the Swap Logical Disk

4. Create the /usr Logical Disk and File System

5. Load the Root File System

6. Load the /usr File System

7. All Installation Steps

========================================================================

Enter ? or <number>? for help, ^ to GO BACK,
or q to QUIT.

Enter Choice: [7]

========================================================================

All Installation Steps

========================================================================


      1. Initialize Physical Disks

========================================================================

Do you want to run this step? [y]  ↵

Enter the Physical Disk specification in DG/UX common format: sd(insc(),0) ↵
Install a Disk Label on a Physical Disk
Do you want to run this step? [y]  ↵
Disk label already exists on disk sd(insc(),0).
Do you want to reinstall disk label? [n] y ↵

                Disk Types

1. 6442    ESDI            322MB
2. 6555    ESDI            648MB
3. 6661    ESDI            322MB
4. 6491    SCSI            322MB
5. 6554    SCSI            662MB
6. 6541    SMD            1066MB
7. 6539    SCSI            179MB
8. 6662    SCSI            322MB
9. 6627    OPTICAL SCSI    295MB
10. None of the Above.
```

```
Enter the type of disk you have: 5↵
Disk label has been installed.

Perform Hardware Formatting on a Physical Disk
Do you want to run this step? [y]  ↵
WARNING:  This operation will DESTROY any data on the Physical
disk sd(insc(),0).

Do you want to continue? [y] ↵

Create DG/UX System Areas on a Physical Disk
Do you want to run this step? [y]  ↵
WARNING:  This operation will DESTROY any data on the Physical disk sd(insc(),0).

Do you want to continue? [y]  ↵

The Physical Disk sd(insc(),0) is 1295922 blocks in size.
Enter the number of blocks to allocate for the remap Area: [315] ↵
Enter the pathname of the boot.aviion file: [/usr/stand/boot.aviion] ↵

Perform Surface Analysis on a Physical Disk
Do you want to run this step? [y] n↵

Do you want to format another Physical Disk? [n]  ↵
```

Step 9: Creating System Logical Disks and File Systems

Creating the Root Logical Disk and File System

```
      2. Create the Root Logical Disk and File System

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [root]  ↵
Enter the Physical Disk specification in the DG/UX common format: [sd(insc(),0)]  ↵
The Physical Disk must be registered for this operation.
Do you want to register it? [y]  ↵
Physical disk sd(insc(),0) has been registered.
Do you want to display the layout of this Physical Disk? [n] ↵
Enter the Physical Disk Address of the starting block of the Logical
Disk Piece: [859]  ↵
Enter the size in blocks of the Logical Disk Piece: [40000]  ↵
The Logical Disk `root' has been created.

Making a file system on the logical disk `root' ...

Made a File System on the Logical Disk `root'.

      3. Create the Swap Logical Disk

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [swap]  ↵
Enter the Physical Disk specification in the DG/UX common format: [sd(insc(),0)]  ↵
Do you want to display the layout of this Physical Disk? [n] ↵
Enter the Physical Disk Address of the starting block of the Logical
Disk Piece: [40859]  ↵
Enter the size in blocks of the Logical Disk Piece: [50000] ↵
The Logical Disk 'swap' has been created.
```

Creating the usr Logical Disk and File System

```
      4. Create the /usr Logical Disk and File System

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [usr]  ↵
Logical Disk Piece 1:
Enter Physical Disk specification in DG/UX common format: [sd(insc(),0)]  ↵
Do you want to display the layout of this Physical Disk? [n] ↵
```

```
Enter the Physical Disk Address of the starting block of Logical Disk
Piece 1: [90859]  ↵
Enter the size in blocks of Logical Disk Piece 1: [160000]  ↵
Do you want to specify any more pieces for this Logical Disk? [n] ↵
The Logical Disk `usr' has been created.

Making a file system on logical disk `usr' ...

Made a File System on Logical Disk `usr'.
```

Step 10: Loading DG/UX Files onto System Logical Disks

Loading the / File System

```
      5. Load the Root File System

Do you want to run this step? [y]  ↵

Do you want to see the names of the files being loaded? [y] n↵
Enter the Logical Disk Unit Name: [root]  ↵

Enter the tape drive specification in DG/UX common format: st(insc(),4) ↵
Ready to load the Root File System.

Mount the first release tape on the tape drive st(insc(),4).
Press New Line when ready to continue... ↵
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
The Root File System has been loaded.
```

Loading the /usr File System

```
      6. Load the /usr File System

Do you want to run this step? [y]  ↵

Do you want to see the names of the files being loaded? [y] n↵
Enter the Logical Disk Unit Name: [usr]  ↵
Enter the tape drive specification in DG/UX common format:  [st(insc(),4)]  ↵
Ready to load the /usr File System.

Mount the first release tape on the tape drive st(insc(),4).
Press New Line when ready to continue... ↵
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
```

```
Loading...
The /usr File System has been loaded.

Your starter system has been installed.

Press New Line when ready to continue...  ↵
```

Pressing Newline returns us to the Initial Installation Menu. We quit **diskman** and return to the SCM.

Step 11: Updating the / and /usr File Systems

This step is not necessary for our example system.

Phase 3: Customizing the Primary Release

Step 12: Booting the Starter Kernel

```
SCM> b sd(insc(),0)root:/dgux.starter ↵

Booting sd(insc(),0)root:/dgux.starter

DG/UX Bootstrap Release 4.30

==========================================================================

DG/UX System Release 4.30, Version (starter)
 Using 16 Megabytes of physical memory
 Found 1 processor(s)
 Processor 0 running
```

Step 13: Specifying Starter Devices

```
                    DG/UX Starter System
Enter the names of the devices you will use in Common Device Specification
Format, with one name per line.  Enter just newline when done.

Examples:  sd(insc(),0) st(insc(),4) cird() st(cisc(),4)

Include duart() for servers and kbd() and grfx() for workstations.

Device name? kbd() ↵
Device name? grfx() ↵
Device name? st(insc(),4) ↵
Device name? sd(insc(),0) ↵
Device name? ↵
Using /dev/dsk/swap as swap file

** root:
No check necessary for root

Mounting /dev/dsk/root as root file system

INIT:  Boot options are: init
INIT:  Cannot open /etc/TIMEZONE. Environment not initialized.

INIT:  /etc/inittab file created from /etc/inittab.proto prototype

INIT:  Checking and mounting /usr...
```

```
INIT:  /usr is now mounted

INIT:  SINGLE USER MODE
su:  unable to access /etc/passwd
#
```

Setting Up DG/UX: Initial Configuration

```
# init 1 ↵

INIT: New run level: 1

chk.fsck:

chk.date:
     Current date/time:  Wed Jun 13 08:15 EDT 1990
--   Are the current date, time, and TIMEZONE correct? (y n) [n] : y↵

Setting up package: dgux

Initializing system database files from .proto files:
initialize /etc/passwd
initialize /etc/group
initialize /etc/dgux.params
.
.
.
Linking /dgux.starter kernel to /dgux
Set permissions on /etc/uucp// 755 root sys

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d
+.....................................................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.....................................................................
Initializing /usr/root.proto directory
Initializing system database files from the original prototype files
initialize /usr/lib/acct/holidays
Cleaning old uucp directory (/usr/lib/uucp)
NOTE:  Merge your old configuration files from /usr/lib/uucp/*_4.20 with
       the new versions in /etc/uucp.

chk.system:
   Cleanup the /etc/ps_data file and /etc/log files
   Check for missing local passwords

** WARNING:  These local accounts have NO password
   root::0:1:Special Admin login/:/sbin/sh
   sysadm::1:sysadm:Regular Admin Login/admin:/sbin/sh

chk.devlink:
   Add short names (for device notes) to /etc/devlinktab
   .
   .
   .

   Link short names for /dev device notes:
   .
   .
   .

Executing the /etc/rcl.d scripts

Starting rc.tcload: terminal controllers
    /usr/sbin/tcload -a

Starting rc.update: update daemon
    update

Starting rc.localfs: local mounts
```

```
    mount -at dg/ux

    The following file systems are now mounted:

    /dev/dsk/root on / type dg/ux (rw)
    /dev/dsk/usr on /usr type dg/ux (rw)

Starting rc.setup:     check for packages that haven't been set up
    All packages are set up.

Press <RETURN> to display prompt ↵

no_node
DG/UX Release 4.30
login: sysadm ↵
```

Step 14: Creating Other Logical Disks and File Systems

After logging in as **sysadm**, we issue this command:

```
# sysadm diskmgmt ↵

Running subcommand 'diskmgmt' from menu 'menu',
SYSADM MAIN MENU

                        Diskman Main Menu


1.  Physical Disk Management Menu

2.  Logical Disk Management Menu

3.  File System Management Menu

4.  Initial Installation Menu

5.  Update Installation Menu

=============================================================================

Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT
Enter choice: 2 ↵
```

In the **diskman** main menu, we select option 2, the Logical Disk Management Menu, then option 1, Create a Logical Disk.

Creating the usr_opt_X11 Logical Disk

```
Enter the Logical Disk Name: usr_opt_X11 ↵
Logical Disk Piece 1:
Enter the Physical Disk specification in DG/UX common format: sd(insc(),0) ↵
Do you want to display the layout of this Physical Disk? [n] ↵
Enter the Physical Disk Address of the starting block of Logical Disk
Piece 1: [250859]  ↵
Enter the size in blocks of Logical Disk Piece 1: [1045063] 105000 ↵
Do you want to specify any more Pieces for this Logical Disk? [n] ↵
The Logical Disk `usr_opt_X11' has been created.
Do you want to make a file system on this Logical Disk? [y]  ↵
No additional information is required, but you may specify mkfs
flags and options if you wish.

Enter the flags and options you want to specify: ↵

Making a file system on logical disk `usr_opt_X11' ...

Made a File System on the Logical Disk `usr_opt_X11'
```

```
Press New Line when ready to continue... ↵
```

At this point, we return to the Logical Disk Management Menu and select option 1, Create a Logical Disk. We continue like this until we have created all the logical disks we planned in Phase 1. When we have finished, we quit **diskman**.

Adding /usr/opt/X11 and Other File Systems

To add file systems, we invoke the File System Management Menu:

```
# sysadm fsmgmt ↵
```

Then we select option 1, Add an entry to the list of file systems.

```
Mount Directory Name? /usr/opt/X11 ↵
Is this a local file system? [yes]  ↵
Writeable? [y]  ↵
Dump Cycle? [d]  ↵
fsck Pass? [1]  ↵
Export? [no] y ↵

The entry for /usr/opt/X11 has been added.

The directory, /usr/opt/X11, does not exist.
Create /usr/opt/X11? [yes]  ↵
Mount the file system? [yes]  ↵

Press the NEWLINE key to see the fsmgmt menu [?, q]:  ↵
```

In the **fsmgmt** menu, we select **addfsys** again. We continue to execute **addfsys** until we have added the file systems we planned during Phase 1.

Step 15: Loading Software Packages with sysadm

```
# sysadm makesrv ↵

Running subcommand 'makesrv' from menu 'releasemgmt',
Software Release Management

Making the PRIMARY release area.
Making the MY_HOST client entry.
makesrv is finished
# sysadm loadpackage ↵

Running subcommand 'loadpackage' from menu 'releasemgmt',
Software Release Management

Release Area? [PRIMARY]  ↵
Tape Drive? [0]  ↵
Is the tape mounted and ready? y ↵
Load Package X11.lg? [yes]  ↵
Load Package X11.man? [yes]  ↵
Load Package X11? [yes]  ↵
Load Package dgux.man? [yes]  ↵
Load Package dtk.man? [yes]  ↵
Load Package dtk? [yes]  ↵
Load Package gcc.man? [yes]  ↵
Load Package gcc? [yes]  ↵
Load Package nfs.man? [yes]  ↵
Load Package nfs? [yes]  ↵
Load Package tcpip.man? [yes]  ↵
Load Package tcpip? [yes]  ↵
List file names while loading? [yes] n ↵
Mount Volume 1.
Is the tape mounted and ready? y ↵
```

```
Skipping tape file 0 to 40.
.
.
.
Updating proto root (/usr/root.proto).
Updating MY_HOST root (/srv/release/PRIMARY/root/MY_HOST).
loadpackage is finished.
#
```

Step 16: Setting Up Software Packages with sysadm

Setting Up TCP/IP

```
# sysadm setuppackage ↵

Running subcommand 'setuppackage' from menu 'releasemgmt',
Software Release Management

Release Area? [PRIMARY]  ↵

The following packages have setup scripts that have not been run:

              Xll       nfs       tcpip          yp
              Xll.lg

Package Name? [all]  ↵

Processing setup scripts for package Xll.
Setup package Xll in usr? [yes]  ↵

            Setting up package: Xll

   Linking /usr/opt/Xll/catman/M_man to /usr/catman/M_man
   Linking /usr/opt/Xll/include/Xm to /usr/include/Xm
   Linking /usr/opt/Xll/include/Mrm to /usr/include/Mrm
   Linking /usr/opt/Xll/include/Uil to /usr/include/Uil

    Linking /usr/opt/Xll and /usr

Processing setup scripts for package Xll.lg.
Setup package Xll.lg in usr? [yes]  ↵

            Setting up package: Xll.lg

Installing Looking Glass executable files ...
lg
lg_pause
vice
vls
vls_add
vls_del

Installing Looking Glass manual page ...

Setup package Xll.lg in MY_HOST root? [yes]  ↵

            Setting up package: Xll.lg

done

Processing setup scripts for package nfs.
Set up package nfs in usr? [yes]  ↵

            Setting up package: nfs

Setup package nfs in MY_HOST root? [yes]  ↵
```

```
            Setting up package: nfs

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d
+.....................................................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.....................................................................

That completes the automated portion of the NFS configuration

Processing setup scripts for package tcpip.
Set up package tcpip in usr? [yes]  ↵

            Setting up package: tcpip

In revisions of the DG/UX operating system before 4.00,
the restricted shell command was named restsh
and the remote shell command was named rsh.
To be compatible with the System V Interface Definition (SVID),
the restricted shell command must be named rsh and
the remote shell command must have a different name.
To be SVID-compliant, Data General names the remote shell remsh.

You are prompted to choose whether or not the names of
the remote and restricted shells comply with the SVID.


If You Choose      The Result Is

    y              The restricted shell is named /bin/rsh
                   The remote shell is named /usr/bin/remsh

    n (default)    The restricted shell is named /bin/restsh
                   The remote shell is named /usr/bin/rsh.
Do you want names to comply with the System V Interface Definition? [n] ↵
Restricted Shell is named /usr/bin/restsh
Remote Shell is named /usr/bin/rsh

Remote Commands Installation Complete

Press NEWLINE when ready to continue... ↵
Setup package tcpip in MY_HOST root? [yes]  ↵

            Setting up package: tcpip

Creating links for initialization scripts...Please Wait

File: /srv/release/PRIMARY/root/MY_HOST/etc/hosts has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/networks has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/services has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/protocols has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/ethers has been created from prototype file.
File: /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params has been created from prototype file.

Press NEWLINE when ready to continue... ↵

Do you want support for loop interface? [y]  ↵

Updating /srv/release/PRIMARY/root/MY_HOST/etc/hosts and
/srv/release/PRIMARY/root/MY_HOST/etc/networks files...Please Wait.

NOTE: Any entries encountered containing conflicting information
      will be deleted from the offending file.

The following lines have been removed from file
"/srv/release/PRIMARY/root/MY_HOST/etc/hosts"
-- Begin Remove List --
127.0.0.1       localhost
-- End of Remove List --
```

```
The entry "127.0.0.1  localhost" has been added
to file "srv/release/PRIMARY/root/MY_HOST/etc/hosts"

Updating "/srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params"
...Please wait...

IMPORTANT NOTE: You MUST have a "loop" entry specified in
your system configuration file.  Consult the help menu or the
system(4) man page for more information.

Local Loopback Environment Installation Complete

Press NEWLINE when ready to continue... ↵

The following queries refer to the host being installed.

Enter host Internet address: 128.223.75.10 ↵
[128.223.75.10] Correct ? [y] ↵

Enter host name: sales ↵
[sales] Correct ? [y] ↵

Enter network name: sales_net ↵
[sales_net] Correct ? [y] ↵

Is "sales_net" a subnetted network? [n] y ↵

Enter the network mask: 0xffffff00 ↵
[0xffffff00] Correct ? [y] ↵

Calculating network address...please wait...

Updating /srv/release/PRIMARY/root/MY_HOST/etc/hosts and
/srv/release/PRIMARY/root/MY_HOST/etc/networks files...please wait

NOTE: Any entries encountered containing conflicting information
      will be deleted from the offending file.

The entry "128.223.75.10    sales" has been added to file
"/srv/release/PRIMARY/root/MY_HOST/etc/hosts"
The entry "sales_net  128.223.75" has been added to file
"/srv/release/PRIMARY/root/MY_HOST/etc/networks"

Enter controller device name: inen0 ↵
[inen0] Correct ? [y] ↵

There are two variations of Broadcast addresses.  A BSD 4.2
compatible broadcast address has a host portion of all
zeros.  A BSD 4.3 compatible broadcast address has a host
portion of all ones.

Calculating network portion of broadcast address...please wait...

Do you want the host portion of the broadcast address to be all ones? [y] ↵

Calculating broadcast address...please wait...

Updating /srv/release/PRIMARY/root/MY_HOST/etc/tcpip.params...
please wait...

IMPORTANT NOTE: You MUST have a "inen" entry specified in
your system configuration file.  Consult the help menu or the
system(4) man page for more information.

Local Environment Installation Complete.

Press NEWLINE when ready to continue. ↵

The following queries refer to IXE configuration.
```

```
Would you like to configure any IXE interfaces? [n]  ↵

IXE Configuration Complete

Press NEWLINE when ready to continue. ↵

Would you like to add a remote host entry? [y] n↵

Do you want to edit the
/srv/release/PRIMARY/root/MY_HOST/etc/protocols file? [n]  ↵

Press NEWLINE when ready to continue. ↵

Do you want to edit the
srv/release/PRIMARY/root/MY_HOST/etc/services file? [n]  ↵

Network Environment Installation Complete

Press NEWLINE when ready to continue. ↵

Enter FTP login directory [/var/ftp]:  ↵
[/var/ftp] Correct ? [y] ↵

Modifying ftp password entry in
/srv/release/PRIMARY/root/MY_HOST/etc/passwd

Directory: /var/ftp exists
Directory: /var/ftp/bin exists
Directory: /var/ftp/etc exists
File "/usr/bin/ls" has been copied to "/var/ftp/bin/ls"
File "/usr/bin/pwd" has been copied to "/var/ftp/bin/pwd"
File "/srv/release/PRIMARY/root/MY_HOST/etc/group" has been
copied to "/var/ftp/etc/group"

FTP Installation Complete

Press NEWLINE when ready to continue. ↵

File: /srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv has been
created from prototype file

Warning:  The following query may produce a security breach in your
system.  An entry in the /srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv
allows a user from the specified remote host having the same user name
to remotely login to your host WITHOUT having to enter a password.
Caution should be exercised when adding entries to this file.

Do you wish to add a host to the
/srv/release/PRIMARY/root/MY_HOST/etc/hosts.equiv file? [n]  ↵
File "/srv/release/PRIMARY/root/MY_HOST/etc/pmterrtab" created from
prototype.
File "/srv/release/PRIMARY/root/MY_HOST/etc/pmttapetab" created from
prototype.

Remote Commands Installation Complete

Press NEWLINE when ready to continue. ↵

"/srv/release/PRIMARY/root/MY_HOST/etc/sendmail.cf" created from
"/srv/release/PRIMARY/root/MY_HOST/etc/arpaproto.cf"

Do you need to customize ruleset 0? [n]  ↵

Modifying mail passwd entry in
/srv/release/PRIMARY/root/MY_HOST/etc/passwd.

Do you want to use sendmail as the mailx router? [y]  ↵

File "/srv/release/PRIMARY/root/MY_HOST/var/mailx/mailx.rc has
been created.
```

```
The entry "set sendmail=/usr/lib/sendmail" has been added to file
"/srv/release/PRIMARY/root/MY_HOST/var/mailx/mailx.rc"

File "/srv/release/PRIMARY/root/MY_HOST/etc/aliases" created from
prototype file.

Do you want to edit the
/srv/release/PRIMARY/root/MY_HOST/etc/aliases file? [n]  ↵

Executing /usr/bin/newaliases...please wait

3 aliases, longest 11 bytes, 53 bytes total

Sendmail Installation Complete

Press NEWLINE when ready to continue... ↵

The Domain Name System provides a means to distribute
management of host information.  It can be used in
place of or in conjunction with Yellow Pages and/or
the /etc/hosts file.

To install and run the domain name server on your
machine you must have data bases set up for the name
server.  Chapter 5 of Setting Up and Managing DG/UX
TCP/IP explains in detail the domain name system and
the requirements to run this service.  Please read
this chapter before attempting to set up the domain
name service on your system.

The answers to the following questions will be used
to partially configure your system for domain name
service access.  The only files that will be edited
are /etc/resolv.conf, /etc/named.boot, and
/etc/svcorder.  If you do not want to edit these file
at this time, answer no to the first question.

Do you want to partially configure for domain name service? [n] ↵

Partial Domain Name Server Installation Complete

Press NEWLINE when ready to continue... ↵
Deleting obsolete files...Please wait...

Processing setup scripts for package yp.

Setup package yp in usr? [yes]  ↵

     Setting up package: yp

Setup package yp in MY_HOST root? [yes]  ↵

     Setting up package: yp

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d
+.....................................................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.....................................................................

Enter the name of the YP Domainname []: sales_domain ↵

---- This host will first run as a YP client.
---- Setting YP domain to:  sales_domain

Is the domainname correct? (y n) [n]: y↵

   That completes the YP setup for a YP client
-- To initiate YP services you will have to change to init level 3.
```

```
-- To complete the YP setup as a YP server or master please refer
   to the ONC/NFS release notice for the release.

setuppackage is finished
#
```

Step 17: Building a Custom Kernel

```
# sysadm newdgux ↵

Running subcommand 'newdgux' from menu 'sysmgmt',
SYSTEM CONFIGURATION MANAGEMENT

System Name? [aviion]  ↵

System File /usr/src/utx/aviion/Build/system.aviion does not exist.
Create the system file? [yes]  ↵
Editor? [vi]  ↵

#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.dgux.proto     94.5"


#--------------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       DG/UX
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for several typical configurations
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#
#

##### Typical AViiON 300 series workstation configuration:

# Note that your system can have a second duart() or an lp() controller,
# but not both!

#    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)

#     ptc()             # -- pseudo-terminal controller device
#     pts()             # -- pseudo-terminal slave device
#     pmt()             # -- pseudo-magtape device
#     log()             # -- Streams logger pseudo-device
#     prf()             # -- profiler pseudo-device


##### Typical AViiON 400 series workstation configuration:

     kbd()              # -- keyboard
     grfx()             # -- graphics display
     sd(insc(),*)       # -- all SCSI disk drives on integrated SCSI adapter
     st(insc(),*)       # -- all SCSI tape drives 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

     ptc()              # -- pseudo-terminal controller device
     pts()              # -- pseudo-terminal slave device
     pmt()              # -- pseudo-magtape device
     log()              # -- Streams logger pseudo-device
     prf()              # -- profiler pseudo-device


##### Typical AViiON 4000 series server configuration:

#     sd(insc(),*)      # -- all SCSI disk drives on integrated SCSI adapter
#     st(insc(),*)      # -- all SCSI tape drives on integrated SCSI adapter
#     sd(cisc(),*)      # -- all SCSI disk drives on Ciprico SCSI adapter
#     st(cisc(),*)      # -- all SCSI tape drives on Ciprico SCSI adapter
#     cird()            # -- Ciprico Rimfire or SMD disk controller

#     inen()            # -- integrated Ethernet controller
#     hken()            # -- Interphase VME Ethernet controller
#     syac()            # -- Systech terminal line controller
#     duart()           # -- integrated Duart terminal line controller
#     duart(1)          # -- second Duart
#     lp()              # -- integrated line printer controller

#     ptc()             # -- pseudo-terminal controller device
#     pts()             # -- pseudo-terminal slave device
#     pmt()             # -- pseudo-magtape device
#     log()             # -- Streams logger pseudo-device
#     prf()             # -- profiler pseudo-device


##### Typical AViiON 5000 or 6000 series server configuration:

#     cird()            # -- Ciprico Rimfire or SMD disk controller
#     sd(cisc(),*)      # -- all SCSI disk drives on Ciprico SCSI adapter
#     st(cisc(),*)      # -- all SCSI tape drives on Ciprico SCSI adapter
#     syac()            # -- Systech terminal line controller
#     duart()           # -- integrated Duart terminal line controller
#     hken(0)           # -- 1st Interphase VME Ethernet controller
#     hken(1)           # -- 2nd Interphase VME Ethernet controller
#     lp()              # -- integrated line printer controller

#     ptc()             # -- pseudo-terminal controller device
#     pts()             # -- pseudo-terminal slave device
#     pmt()             # -- pseudo-magtape device
#     log()             # -- Streams logger pseudo-device
#     prf()             # -- profiler pseudo-device

#
#---------------------------------------------------------------------
```

```
#--------------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You should not have to specify any additional protocols in order to
# use this product.
#
#
#   Protocol Name
#   -------------
#

#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# It is recommended that you specify the Transport Provider Interface
# STREAMS modules, timod and tirdwr.
#
#
#   STREAMS Module Name
#   -------------------
#
    timod
    tirdwr
#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# The default values from the master file will be used unless
# explicitly overridden in this file.
#
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# You should set the TZ variable to accurately reflect your timezone
# (300 minutes west of GMT is USA Eastern time).
#
# You should set the MAXUP variable to the maximum number of processes
# that each user will be allowed to run simultaneously.  This number
# should be at least 64 for workstations.
#
# You should set the NODE variable to control your nodename for uname(1)
# and uucp(1), but not more than 255 characters.
#
# You should set the DUMP variable to the name of the tape device (in
# DG/UX Common Device Specification Format) that will be the default
# device to take dumps in case of system emergencies.  For diskless
# workstations, the DUMP variable should be set to the network device
# used to boot the machine.
#
# If your system is a diskless workstation, you should set the
```

```
# PERCENTNFS variable to 100 in order to get the best possible NFS
# performance.
#
# If either your system's root file system or its swap file will be
# mounted over NFS (a diskless workstation will NFS-mount both, a
# dataless workstation will NFS-mount only the root), you must set
# the NETBOOTDEV variable to the name of the network device (in DG/UX
# Common Device Specification Format) that will be used in booting
# over the network.
#
# If your system's root file system will be mounted over NFS (as will
# be done on both diskless and dataless workstations), you must set the
# ROOTFSTYPE variable to NETWORK_ROOT.
#
# If your system's swap file will be mounted over NFS (as will be done
# on diskless workstations), you must set the SWAPDEVTYPE variable to
# NETWORK_SWAP.
#
#
#   Parameter Name              Value
#   --------------              -----
#
    TZ                          300
    MAXUP                       64
    NODE                        "sales"

    DUMP                        "st(insc(),4)"
### DUMP                        "inen()"

### PERCENTNFS                  100
### NETBOOTDEV                  "inen()"
### ROOTFSTYPE                  NETWORK_ROOT
### SWAPDEVTYPE                 NETWORK_SWAP

#
#-----------------------------------------------------------------
#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.nfs.proto      94.2"


#-----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       NFS
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Devices:
#
# List all devices in this section, one entry per line.
# The string is the name of the device.
# Note that some pseudo-devices have no device code at
# all, so none should be listed.
```

```
# Any other text on a line will be ignored.
#
#
#   Device Name
#   -----------
#

    plm()               # -- network lock manager pseudo-device

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You will not need to specify any additional protocols to use this
# product.
#
#
#   Protocol Name
#   -------------
#

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# You will not need to specify any additional STREAMS modules
# to use this product.
#
#
#   STREAMS Module Name
#   -------------------
#

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Tuneable Configuration Parameters:

# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.

# To use NFS, you must specify the NFS variable so that its implied
# value will be used.


#   Parameter Name              Value
#   --------------              -----

    NFS
```

```
#-------------------------------------------------------------------
#       Copyright (C) Data General Corporation, 1985 - 1989.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K    tcpip   90.1"


#-------------------------------------------------------------------
#
# Prototype fragment of system configuration for:

# (Product Name):       TCP/IP
# (Release):            4.30


# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.

#-------------------------------------------------------------------


#-------------------------------------------------------------------
# Devices:

# List all devices and pseudo-devices in this section, one entry per
# line.  Verify typical configurations for both workstations and
# server systems. You will need at least one LAN controller
# (inen or hken). (see the DG/UX system.proto file for these)



# The protocol engines are Streams multiplexing drivers

        ip()
        tcp()
        udp()

# It is also recommended that you include the loopback pseudo-device.


    loop()


#-------------------------------------------------------------------

#-------------------------------------------------------------------
# Protocols:

# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.

# You will need the tcp, ip, udp and icmp protocols.


#   Protocol Name
#   -------------


        ipproto_ip
        ipproto_tcp
        ipproto_udp
        ipproto_icmp
```

```
#----------------------------------------------------------------


#----------------------------------------------------------------
# STREAMS Modules:

# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.


#   STREAMS Module Name
#   -------------------

        ether
        arp
        socsys
        netlog


#----------------------------------------------------------------


#----------------------------------------------------------------
# Tuneable Configuration Parameters:

# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
:wq ↵
```

Installing the New Kernel

```
Ready to Configure a Kernel? [yes]  ↵

sysadm will now run config on /usr/src/uts/aviion/Build/system.aviion

Config succeeded.

sysadm will now attempt to build a kernel.
Building...
The build succeeded.

Install the New Kernel? [no] y↵
For a Diskless Client of this Host? [no]  ↵
Kernel Pathname? [/dgux.aviion]  ↵

The new kernel has been copied to /dgux.aviion.
Link /dgux to the New Kernel? [yes]  ↵

The new kernel will not take effect until you shutdown and reboot.
To do this, quit sysadm, and say:

  cd /
  /etc/shutdown
  /etc/halt -q

Until you do this, a few commands which depend on the symbol table
in /dgux (such as the kernel profiler and netstat) may not work correctly.
This should not cause any serious difficulties.
#
```

Bringing Down the System

```
# cd / ↵
# /etc/shutdown -g0 -y ↵
  ...
# /etc/halt -q ↵
```

Step 18: Setting Default Boot Characteristics

```
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) -> 1 ↵


Change boot parameters

1 Change system boot path

2 Change diagnostic boot path

3 Change data transfer mode [BLOCK]

4 Return to previous screen

Enter choice(s) -> 1 ↵

System boot path  = []

Do you want to modify the boot path? [N] y↵

Enter new system boot path -> sd(insc(),0)root:/dgux ↵

System boot path = [sd(insc(),0)root:/dgux]
Do you want to modify the boot path? [N] ↵

Do you want to boot? [N] y↵
```

Bring the System up to Run Level 1

```
# init 1 ↵
```

Changing the Default Initial Run Level

We change the default initial run level by editing the /etc/inittab file and changing this line:

```
def:s:initdefault:
```

To this line:

```
def:3:initdefault:
```

Step 19: Starting System Administration

Adding Groups

Following Chapter 14, we add groups in this step.

Adding User Accounts

Following Chapter 14, we add users in this step. Their home directories are in **/sales/users**.

Setting Up Terminals

Following Chapter 10, we set up our terminal lines in this step.

Starting the Accounting System

Following Chapter 15, we set up accounting in this step.

Adding Lineprinters

Following Chapter 11, we add our printers in this step.

Bring the system up to multi-user mode

At this point we can bring our system up to run level 3.

```
# init 3 ↵
```

Phase 4: Adding OS Releases and Clients

Step 20: Adding Secondary Releases

```
# sysadm addrelease ↵

New Release Name? 68k_sunos_4 ↵
Usr Directory?  [/srv/release/68k_sunos_4/usr]  ↵
Share Directory? [/srv/share]  ↵
Client Root Parent Directory? [/srv/release/68k_sunos_4]   ↵
Client Swap Directory? [/srv/swap]  ↵

Release 68k_sunos_4 has been added. You may now use loadpackage.
```

Step 21: Building Kernels for Diskless Clients

On AViiON Systems

```
# sysadm newdgux ↵

Running subcommand 'newdgux' from menu 'sysmgmt',
SYSTEM CONFIGURATION MANAGEMENT

System Name? [aviion] diskless ↵

Editor?  [vi]   ↵

#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.dgux.proto     94.5"


#-----------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       DG/UX
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for several typical configurations
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#
#

##### Typical AViiON 300 series workstation configuration:

# Note that your system can have a second duart() or an lp() controller,
# but not both!

    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)

    ptc()               # -- pseudo-terminal controller device
    pts()               # -- pseudo-terminal slave device
    pmt()               # -- pseudo-magtape device
    log()               # -- Streams logger pseudo-device
```

```
    prf()               # -- profiler pseudo-device


##### Typical AViiON 400 series workstation configuration:

#     kbd()             # -- keyboard
#     grfx()            # -- graphics display
#     sd(insc(),*)      # -- all SCSI disk drives on integrated SCSI adapter
#     st(insc(),*)      # -- all SCSI tape drives on integrated SCSI adapter
#     inen()            # -- integrated Ethernet controller
#     duart()           # -- integrated Duart terminal line controller
#     duart(1)          # -- second Duart
#     lp()              # -- integrated line printer controller
#
#     ptc()             # -- pseudo-terminal controller device
#     pts()             # -- pseudo-terminal slave device
#     pmt()             # -- pseudo-magtape device
#     log()             # -- Streams logger pseudo-device
#     prf()             # -- profiler pseudo-device


##### Typical AViiON 4000 series server configuration:

#     sd(insc(),*)      # -- all SCSI disk drives on integrated SCSI adapter
#     st(insc(),*)      # -- all SCSI tape drives on integrated SCSI adapter
#     sd(cisc(),*)      # -- all SCSI disk drives on Ciprico SCSI adapter
#     st(cisc(),*)      # -- all SCSI tape drives on Ciprico SCSI adapter
#     cird()            # -- Ciprico Rimfire or SMD disk controller
#
#     inen()            # -- integrated Ethernet controller
#     hken()            # -- Interphase VME Ethernet controller
#     syac()            # -- Systech terminal line controller
#     duart()           # -- integrated Duart terminal line controller
#     duart(1)          # -- second Duart
#     lp()              # -- integrated line printer controller
#
#     ptc()             # -- pseudo-terminal controller device
#     pts()             # -- pseudo-terminal slave device
#     pmt()             # -- pseudo-magtape device
#     log()             # -- Streams logger pseudo-device
#     prf()             # -- profiler pseudo-device


##### Typical AViiON 5000 or 6000 series server configuration:

#     cird()            # -- Ciprico Rimfire or SMD disk controller
#     sd(cisc(),*)      # -- all SCSI disk drives on Ciprico SCSI adapter
#     st(cisc(),*)      # -- all SCSI tape drives on Ciprico SCSI adapter
#     syac()            # -- Systech terminal line controller
#     duart()           # -- integrated Duart terminal line controller
#     hken(0)           # -- 1st Interphase VME Ethernet controller
#     hken(1)           # -- 2nd Interphase VME Ethernet controller
#     lp()              # -- integrated line printer controller
#
#     ptc()             # -- pseudo-terminal controller device
#     pts()             # -- pseudo-terminal slave device
#     pmt()             # -- pseudo-magtape device
#     log()             # -- Streams logger pseudo-device
#     prf()             # -- profiler pseudo-device
#
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
```

```
# configure into your system.
#
# You should not have to specify any additional protocols in order to
# use this product.
#
#
#   Protocol Name
#   -------------
#

#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# It is recommended that you specify the Transport Provider Interface
# STREAMS modules, timod and tirdwr.
#
#
#   STREAMS Module Name
#   -------------------
#
    timod
    tirdwr
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# The default values from the master file will be used unless
# explicitly overridden in this file.
#
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# You should set the TZ variable to accurately reflect your timezone
# (300 minutes west of GMT is USA Eastern time).
#
# You should set the MAXUP variable to the maximum number of processes
# that each user will be allowed to run simultaneously.  This number
# should be at least 64 for workstations.
#
# You should set the NODE variable to control your nodename for uname(1)
# and uucp(1), but not more than 255 characters.
#
# You should set the DUMP variable to the name of the tape device (in
# DG/UX Common Device Specification Format) that will be the default
# device to take dumps in case of system emergencies.  For diskless
# workstations, the DUMP variable should be set to the network device
# used to boot the machine.
#
# If your system is a diskless workstation, you should set the
# PERCENTNFS variable to 100 in order to get the best possible NFS
# performance.
#
# If either your system's root file system or its swap file will be
# mounted over NFS (a diskless workstation will NFS-mount both, a
```

```
# dataless workstation will NFS-mount only the root), you must set
# the NETBOOTDEV variable to the name of the network device (in DG/UX
# Common Device Specification Format) that will be used in booting
# over the network.
#
# If your system's root file system will be mounted over NFS (as will
# be done on both diskless and dataless workstations), you must set the
# ROOTFSTYPE variable to NETWORK_ROOT.
#
# If your system's swap file will be mounted over NFS (as will be done
# on diskless workstations), you must set the SWAPDEVTYPE variable to
# NETWORK_SWAP.
#
#
#   Parameter Name              Value
#   --------------              -----
#
    TZ                          300
    MAXUP                       64
    NODE                        "diskless"

### DUMP                        "st(insc(),4)"
    DUMP                        "inen()"

    PERCENTNFS                  100
    NETBOOTDEV                  "inen()"
    ROOTFSTYPE                  NETWORK_ROOT
    SWAPDEVTYPE                 NETWORK_SWAP

#
#-------------------------------------------------------------------
#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.nfs.proto      94.2"


#-------------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       NFS
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-------------------------------------------------------------------


#-------------------------------------------------------------------
# Devices:
#
# List all devices in this section, one entry per line.
# The string is the name of the device.
# Note that some pseudo-devices have no device code at
# all, so none should be listed.
# Any other text on a line will be ignored.
#
#
#   Device Name
#   -----------
```

```
#
    plm()               # -- network lock manager pseudo-device

#
#----------------------------------------------------------------


#----------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You will not need to specify any additional protocols to use this
# product.
#
#
#   Protocol Name
#   -------------
#

#
#----------------------------------------------------------------


#----------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# You will not need to specify any additional STREAMS modules
# to use this product.
#
#
#   STREAMS Module Name
#   -------------------
#

#
#----------------------------------------------------------------


#----------------------------------------------------------------
# Tuneable Configuration Parameters:

# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.

# To use NFS, you must specify the NFS variable so that its implied
# value will be used.


#   Parameter Name              Value
#   --------------              -----

    NFS

#----------------------------------------------------------------
#       Copyright (C) Data General Corporation, 1985 - 1989.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
```

```
#       This software is made available solely pursuant to the
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K    tcpip   90.1"


#-----------------------------------------------------------------
#
# Prototype fragment of system configuration for:

# (Product Name):       TCP/IP
# (Release):            4.30


# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.

#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Devices:

# List all devices and pseudo-devices in this section, one entry per
# line.  Verify typical configurations for both workstations and
# server systems. You will need at least one LAN controller
# (inen or hken). (see the DG/UX system.proto file for these)



# The protocol engines are Streams multiplexing drivers

        ip()
        tcp()
        udp()

# It is also recommended that you include the loopback pseudo-device.


    loop()


#-----------------------------------------------------------------

#-----------------------------------------------------------------
# Protocols:

# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.

# You will need the tcp, ip, udp and icmp protocols.


#   Protocol Name
#   -------------


        ipproto_ip
        ipproto_tcp
        ipproto_udp
        ipproto_icmp


#-----------------------------------------------------------------
```

```
#--------------------------------------------------------------------
# STREAMS Modules:

# List all explicit STREAMS modules in this section, one entry per line.
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.


#   STREAMS Module Name
#   -------------------

        ether
        arp
        socsys
        netlog


#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Tuneable Configuration Parameters:

# List all configuration parameters you wish to override in this
# section, one entry per line.
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
:wq ↵

Ready to Configure a Kernel? [yes]  ↵

sysadm will now run config on /usr/src/uts/aviion/Build/system.diskless

Config succeeded.

sysadm will now attempt to build a kernel.
Building...

The build succeeded.

Install the New Kernel? [no]  ↵

For a Diskless Client of this Host? [no] y↵

Kernel Pathname? [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵

Link all primary clients to the new kernel? [y]  ↵
```

Step 22: Setting OS Client Defaults

```
# sysadm clientdefaults  ↵

Running subcommand 'clientdefaults' from menu 'clientmgmt',
Client Management

Defaults Set Name? [generic] dgset ↵
Default Release Name? PRIMARY ↵
Default Swap Size? [16M]  ↵
Default Home Directory? [/home] /sales/accounts ↵
Default Kernel? [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵
Default Bootstrap File? [/usr/stand/boot.aviion]  ↵
Defaults for Set dgset have been assigned.
```

Step 23: Adding OS Clients

Adding Clients to /etc/hosts

```
# sysadm addhost ↵

This host is the YP master.  You must choose between
accessing the global or local list.

Access the Global/Network List? [yes]  ↵
Host name? dg1 ↵
Host address? 128.223.2.2 ↵
YP Server? [yes] no ↵
The YP server query is asked only on the master server.
The entry for dg1 has been added.
Do you want to add another host? [no]  ↵

Updating the Yellow Pages host and network maps.
```

Adding Clients to /etc/ethers

```
# sysadm addether ↵

Host Name? dg1 ↵
Ethernet Address? 08:00:1b:00:a0:17 ↵
The entry for dg1 has been added.
Do you want to add another entry? [n]  ↵
```

Adding a Client to a Release

```
# sysadm addclient ↵

Server Host Name? [sales] ↵
Client Host Name? dg1 ↵
Defaults Set Name? [generic] dgset ↵
Use all defaults from dgset? yes ↵
Creating client root.
Creating client swap file.
Creating client /etc/fstab.
Creating client /etc/hosts.
Creating client /tcpip.params.
Creating client /etc/nfs.params.
Client dg1 has been added.
Do you wish to add another client? [yes] no ↵
```

Step 24: Booting and Setting Up an OS Client

We boot an OS client with the following command line:

```
SCM> b inen() ↵
```

After the boot has completed, we come up in single-user mode. Coming up to run level one sets up the DG/UX system as on any other system. When DG/UX system setup is complete, we run **setuppackage** as on any other system to set up packages.

Example 3: Installing the DG/UX System on an Example AV6200 System

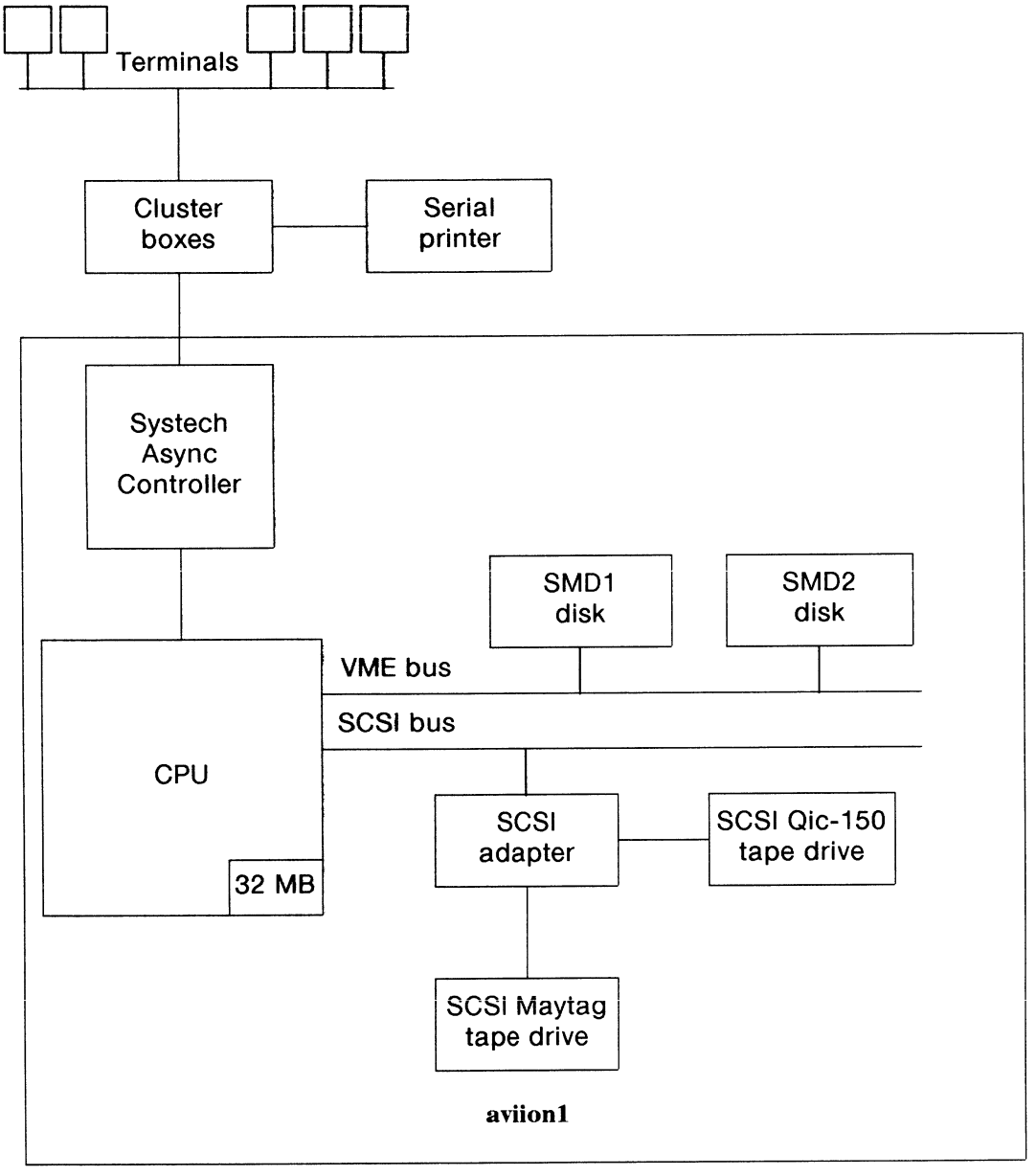
Read the planning and installation procedures in Chapter 2 before reading this section.

This section shows the system/user interaction involved in installing the DG/UX system on an example system. The example system may not be the same as your own system, so do not use this example as a guide during installation.

The example configuration is a multiuser system supporting approximately 50 users. The system is designed to support database and program development activities. The components of the system are:

- One AV6200 named **aviion1** with 32MBytes of memory.
- Two 1GB SMD disks.
- One SCSI QIC-150 150MBytes tape drive with SCSI adapter.
- One SCSI 2GB Maytag tape drive.
- One Systech asynchronous controller supporting 128 terminals.
- One serial printer.

An illustration of the system follows.



Phase 1: Planning the Installation

Step 1: Determining How You Will Use Your System

You have a standalone system (multiuser system supporting approximately 50 users). You also have the Operating Systems Package. Specific software products are listed as follows:

Table 2-16 Software Products

Products
DG/UX, Gnu C, DTK, DG/UX man pages
Language compiler
Database system

You have DG/UX system release 4.30. You will be loading from tape.

Identifying Your System's Devices

You have an AV6200 system. Your devices follow:

Table 2-17 Device Information for Our Example System

Device	Specification	Device No.	SCSI ID No.
First disk controller (boot disk)	**cimd(0,0)**	18	NA
Second disk controller	**cimd(0,1)**	19	NA
Asynchronous controller	**syac()**	60	NA
Tape drive	**st(cisc(),4)**	NA	4

You have two 6541 SMD disk devices, which contain 1066 MBytes (or 2183168 blocks) each.

Step 3: Learning About Hosts, Software Disks, and File Systems

Your host's name is **aviion1**.

Step 4: Allocating Disk Space

After reading Step 4 and the product release notices, you determine that you will need five logical disks including swap. Table 2-18 shows the logical disks and their software packages.

You will create another logical disk named **udd_aviion1** as the working directory for the user(s) of **aviion1**. Because you expect a fair number of users to be compiling and editing files at one time, you will create a separate logical disk for **/var/tmp**.

Table 2-18 A Logical Disk-File System Plan

Disk Type	Physical Disk Name	Logical Disk Name	Piece	Mounted File System Name
SMD	cimd(0,0)	swap	1	
		root	1	/
		usr	1	/usr
		var_tmp	1	/var/tmp
SMD	cimd(0,1)	usr_opt_lang1	1	/usr/opt/lang1
		usr_opt_db1	1	/usr/opt/db1
		udd_aviion1	1	/udd/aviion1

System Logical Disks

Our main system logical disks are

root 40,000 blocks

usr 160,000 blocks

swap 50,000 blocks

Other Logical Disks

The logical disks we need are

var_tmp 30,000 blocks

usr_opt_lang1 5,000 blocks

usr_opt_db1 50,000 blocks

udd_aviion1 500,000 blocks

Step 5: Understanding the DG/UX Directory Tree

No action is required for this step.

Step 6: Assembling Network Information for Your System

This step is not necessary for our example system.

Phase 2: Loading the Primary Release from Tape to Disk

Step 7: Booting the Disk Management (diskman) Utility

```
SCM> b st(cisc(),4) ↵

Booting st(cisc()4,)

DG/UX Bootstrap Release 4.30
```

```
Skipping tape file 1.

================================================================================

DG/UX System Release 4.30, Version Diskman
 Using 32 Megabytes of physical memory
 Found 1 processor(s)
 Processor 0 running


                    DG/UX Starter System
Enter the names of the devices you will use in Common Device Specification
Format, with one name per line.  Enter just newline when done.

Examples:  sd(insc(),0) st(insc(),4) cird() st(cisc(),4)

Include duart() for servers and kbd() and grfx() for workstations.

Device name? duart() ↵
Device name?  ↵
...
```

Step 8: Initializing Physical Disks with diskman

```
             Diskman Main Menu


1.  Physical Disk Management Menu

2.  Logical Disk Management Menu

3.  File System Management Menu

4.  Initial Installation Menu

5.  Update Installation Menu


Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT
Enter choice: 4


================================================================================

           Initial Installation Menu


1. Initialize Physical Disks

2. Create the Root Logical Disk and File System

3. Create the Swap Logical Disk

4. Create the /usr Logical Disk and File System

5. Load the Root File System

6. Load the /usr File System

7. All Installation Steps

================================================================================

Enter ? or <number>? for help, ^ to GO BACK,
or q to QUIT.

Enter Choice: [7]
```

```
=============================================================================

All Installation Steps

=============================================================================


      1. Initialize Physical Disks

=============================================================================

Do you want to run this step? [y]  ↵

Enter the Physical Disk specification in DG/UX common format: cimd(0,0) ↵
Install a Disk Label on a Physical Disk
Do you want to run this step? [y]  ↵
Disk label already exists on disk cimd(0,0).
Do you want to reinstall disk label? [n] y↵

                Disk Types

1. 6442    ESDI            322MB
2. 6555    ESDI            648MB
3. 6661    ESDI            322MB
4. 6491    SCSI            322MB
5. 6554    SCSI            662MB
6. 6541    SMD            1066MB
7. 6539    SCSI            179MB
8. 6662    SCSI            322MB
9. 6627    OPTICAL SCSI    295MB
10. None of the Above.

Enter the type of disk you have: 6↵
Disk label has been installed.

Perform Hardware Formatting on a Physical Disk
Do you want to run this step? [y]  ↵
WARNING:  This operation will DESTROY any data on the Physical
disk cimd(0,0).

Do you want to continue? [y]  ↵

Create DG/UX System Areas on a Physical Disk
Do you want to run this step? [y]  ↵
WARNING:  This operation will DESTROY any data on the Physical
disk cimd(0,0).

Do you want to continue? [y]  ↵

The Physical Disk cimd(0,0) is 2095922 blocks in size.
Enter the number of blocks to allocate for the remap Area: [315]  ↵
Enter the pathname of the boot.aviion file: [/usr/stand/boot.aviion]  ↵

Perform Surface Analysis on a Physical Disk
Do you want to run this step? [y] n↵

Do you want to format another Physical Disk? [n] y↵
```

We perform same step for the **cimd(0,1)** disk.

Step 9: Creating System Logical Disks and File Systems

Creating the Root Logical Disk and File System

```
      2. Create the Root Logical Disk and File System

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [root]  ↵
Enter the Physical Disk specification in the DG/UX common format: [cimd(0,0)]  ↵
The Physical Disk must be registered for this operation.
Do you want to register it? [y]  ↵
Physical disk cimd(0,0) has been registered.
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of the Logical
Disk Piece: [859]  ↵
Enter the size in blocks of the Logical Disk Piece: [40000]  ↵
The Logical Disk `root' has been created.

Making a file system on the Logical Disk `root' ...

Made a File System on the Logical Disk `root'.

      3. Create the Swap Logical Disk

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [swap]  ↵
Enter the Physical Disk specification in the DG/UX common format: [cimd(0,0)]  ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of the Logical
Disk Piece: [40859]  ↵
Enter the size in blocks of the Logical Disk Piece: [50000]  ↵
The Logical Disk 'swap' has been created.
```

Creating the usr Logical Disk and File System

```
      4. Create the /usr Logical Disk and File System

Do you want to run this step? [y]  ↵
Enter the Logical Disk Name: [usr]  ↵
Logical Disk Piece 1:
Enter Physical Disk specification in DG/UX common format: [cimd(0,0)]  ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of Logical Disk
Piece 1: [90859]  ↵
Enter the size in blocks of Logical Disk Piece 1: [160000]  ↵
Do you want to specify any more pieces for this Logical Disk? [n]  ↵
The Logical Disk `usr' has been created.

Making a file system on logical disk `usr' ...

Made a File System on Logical Disk `usr'.
```

Step 10: Loading DG/UX Files onto System Logical Disks

Loading the / File System

```
       5. Load the Root File System

Do you want to run this step? [y]  ↵

Do you want to see the names of the files being loaded? [y]  ↵
Enter the Logical Disk Unit Name: [root]  ↵

Enter the tape drive specification in DG/UX common format: st(cisc(),4) ↵
Ready to load the Root File System.

Mount the first release tape on the tape drive
Press New Line when ready to continue... ↵
Loading...
Loading...
Loading...
```

```
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
The Root File System has been loaded.

      6. Load the /usr File System

Do you want to run this step? [y]  ↵

Do you want to see the names of the files being loaded? [y]  ↵
Enter the Logical Disk Unit Name: [usr]  ↵
Enter the tape drive specification in DG/UX common format:  [st(cisc(),4)]   ↵
Ready to load the /usr File System.

Mount the first release tape on the tape drive st(cisc(),4).
Press New Line when ready to continue... ↵
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
Loading...
The /usr File System has been loaded.

Your starter system has been installed.

Press New Line when ready to continue...  ↵
```

At this point, we quit **diskman** and return to the SCM.

Step 11: Updating the / and /usr File Systems

This step is not necessary for our example system.

Phase 3: Customizing the Primary Release

Step 12: Booting the Starter Kernel

```
SCM> b cimd(0,0)root:/dgux.starter ↵

Booting cimd(0,0)root:/dgux.starter

DG/UX Bootstrap Release 4.30

==========================================================================
```

```
DG/UX System Release 4.30, Version (starter)
 Using 32 Megabytes of physical memory
 Found 1 processor(s)
 Processor 0 running
```

Step 13. Specifying Starter Devices

```
                    DG/UX Starter System
Enter the names of the devices you will use in Common Device Specification
Format, with one name per line.  Enter just newline when done.

Examples:  sd(insc(),0) st(insc(),4) cird() st(cisc(),4)
Include duart() for servers and kbd() and grfx() for workstations.

Device name? duart() ↵
Device name? st(cisc(),4) ↵
Device name? cimd(0,0) ↵
Device name? ↵
Using /dev/dsk/swap as swap file

** root:
No check necessary for root

Mounting /dev/dsk/root as root file system

INIT:  Boot options are: init
INIT:  Cannot open /etc/TIMEZONE. Environment not initialized.

INIT:  /etc/inittab file created from /etc/inittab.proto prototype

INIT:  Checking and mounting /usr...

INIT:  /usr is now mounted

INIT:  SINGLE USER MODE
su:  unable to access /etc/passwd
#
```

Setting Up DG/UX: Initial Configuration

```
# init 1 ↵

INIT: New run level: 1

chk.fsck:

chk.date:
     Current date/time:  Wed Jun 13 08:15 EDT 1990
--   Are the current date, time, and TIMEZONE correct? (y n) [n] : y ↵

Setting up package: dgux

Initializing system database files from .proto files:
initialize /etc/passwd
initialize /etc/group
initialize /etc/dgux.params
.
.
.
Linking /dgux.starter kernel to /dgux
Set permissions on /etc/uucp// 755 root sys

Setting up the rc#.d directory links.
Remove links in /srv/release/PRIMARY/root/MY_HOST/etc/rc#.d
+.........................................
Link from /usr/sbin/init.d to /srv/release/PRIMARY/root/MY_HOST/etc
+.........................................
```

```
Initializing /usr/root.proto directory
Initializing system database files from the original prototype files
initialize /usr/lib/acct/holidays
Cleaning old uucp directory (/usr/lib/uucp)
NOTE:  Merge your old configuration files from /usr/lib/uucp/*_4.20 with
       the new versions in /etc/uucp.

chk.system:
   Cleanup the /etc/ps_data file and /etc/log files
   Check for missing local passwords

** WARNING:  These local accounts have NO password
   root::0:1:Special Admin login/:/sbin/sh
   sysadm::1:sysadm:Regular Admin Login/admin:/sbin/sh

chk.devlink:
   Add short names (for device notes) to /etc/devlinktab
   .
   .
   .

   Link short names for /dev device notes:
   .
   .
   .

Executing the /etc/rc1.d scripts

Starting rc.tcload: terminal controllers
    /usr/sbin/tcload -a

Starting rc.update: update daemon
    update

Starting rc.localfs: local mounts
    mount -at dg/ux

    The following file systems are now mounted:

    /dev/dsk/root on / type dg/ux (rw)
    /dev/dsk/usr on /usr type dg/ux (rw)

Starting rc.setup:     check for packages that haven't been set up.
    All packages are set up.

Press <RETURN> to display prompt. ↵

no_node
DG/UX Release 4.30
login: sysadm ↵
```

Step 14: Creating Other Logical Disks and File Systems

After logging in as **sysadm**, we issue this command:

```
# sysadm diskmgmt ↵

Running subcommand 'diskmgmt' from menu 'menu',
SYSADM MAIN MENU

                        Diskman Main Menu


1.  Physical Disk Management Menu

2.  Logical Disk Management Menu
```

```
3.  File System Management Menu

4.  Initial Installation Menu

5.  Update Installation Menu

=========================================================================

Enter ? or <number>? for HELP, ^ to GO BACK, or q to QUIT
Enter choice: 2 ↵
```

In the **diskman** main menu, we select option 2, the Logical Disk Management Menu, then option 1, Create a Logical Disk.

Creating the usr_opt_lang1 Logical Disk

```
Enter the Logical Disk Name: usr_opt_lang1 ↵
Logical Disk Piece 1:
Enter the Physical Disk specification in DG/UX common format: cimd(0,0) ↵
Do you want to display the layout of this Physical Disk? [n]  ↵
Enter the Physical Disk Address of the starting block of Logical
Disk Piece 1: [250859] ↵
Enter the size in blocks of Logical Disk Piece 1: [1845063] 5000 ↵
Do you want to specify any more Pieces for this Logical Disk? [n]  ↵
The Logical Disk `usr_opt_lang1' has been created.
Do you want to make a file system on this Logical Disk? [y]  ↵
No additional information is required, but you may specify mkfs
flags and options if you wish.

Enter the flags and options you want to specify: ↵

Making a file system on Logical Disk `usr_opt_lang1' ...

Made a File System on the Logical Disk `usr_opt_lang1'

Press New Line when ready to continue... ↵
```

Again, we return to the Logical Disk Management Menu and select option 1, Create a Logical Disk. We continue like this until we have created all the logical disks we planned in Phase 1. When we have finished, we quit **diskman**.

Adding the /usr/opt/lang1 File System to /etc/fstab with sysadm addfsys

To add file systems, we invoke the File System Management Menu:

```
# sysadm fsmgmt ↵
```

Then we select option 1, Add a file system.

```
Mount Directory Name? /usr/opt/lang1 ↵
Is this a local file system? [yes]  ↵
Writeable? [y]  ↵
Dump Cycle? [d]  ↵
fsck Pass? [1]  ↵
Export? [no]  ↵

The entry for /usr/opt/lang1 has been added.

The directory, /usr/opt/lang1, does not exist.
Create /usr/opt/lang1? [yes]  ↵
Mount the file system? [yes]  ↵

Press the NEWLINE key to see the fsmgmt menu.  ↵
```

In the **fsmgmt** menu, we select **addfsys** again. We continue to execute **addfsys** until we have added the file systems we planned during Phase 1.

Step 15: Loading Software Packages with sysadm

```
# sysadm makesrv ↵

Running subcommand `makesrv' from menu `releasemgmt',
Software Release Management

Making the PRIMARY release area.
Making the MY_HOST client entry.
makesrv is finished
# sysadm loadpackage ↵

Running subcommand 'loadpackage' from menu 'releasemgmt',
Software Release Management

Release Area? [PRIMARY]   ↵
Tape Drive? [0]   ↵
Is the tape mounted and ready? y ↵
Load Package dgux.man? [yes]  ↵
Load Package dtk.man? [yes]  ↵
Load Package dtk? [yes]  ↵
Load Package gcc.man? [yes]  ↵
Load Package gcc? [yes]  ↵
List file names while loading? [yes]  n ↵
Mount Volume 1.
Is the tape mounted and ready? y ↵
Skipping tape file 0 to 35.
.
.
.
Updating proto root (/usr/root.proto).
Updating MY_HOST root (/srv/release/PRIMARY/root/MY_HOST).
loadpackage is finished.
#
```

We now load our language compiler and database system following the installation instructions given in their respective release notices and installation guides.

Step 16: Setting Up Software Packages with sysadm

None of the DG/UX packages we loaded require setup via **sysadm setuppackage**.

Step 17: Building a Custom Kernel

```
# sysadm newdgux ↵

Running subcommand 'newdgux' from menu 'sysmgmt',
SYSTEM CONFIGURATION MANAGEMENT

System Name? [aviion] ↵

System File /usr/src/uts/aviion/Build/system.aviion does not exist.
Create the system file? [yes] ↵
Editor? [vi] ↵

#       Copyright (c) Data General Corporation 1990.
#       All Rights Reserved.
#       Licensed Material -- Property of Data General Corporation.
#       This software is made available solely pursuant to the
```

```
#       terms of a DGC license agreement which governs its use.

# sccsid = "@(#) 88K 1990 system.dgux.proto     94.5"


#-------------------------------------------------------------------
#
# Prototype fragment of system configuration for:
#
# (Product Name):       DG/UX
# (Release):            4.30
#
#
# This prototype is provided to assist you in creating your
# customized system configuration file.
# This file consists of system file entries pertaining to this
# product.  Include this fragment in your customized system file
# and edit it to reflect your system's configuration.
# See this product's master file (in /usr/etc/master.d) for more details.
#
#-------------------------------------------------------------------


#-------------------------------------------------------------------
# Devices:
#
# List all devices and pseudo-devices in this section, one entry per
# line.  Typical configurations for several typical configurations
# have been provided below; delete entries that do not apply to your
# system and add to the list any devices your system has that are not
# already listed.
#
#

##### Typical AViiON 300 series workstation configuration:

# Note that your system can have a second duart() or an lp() controller,
# but not both!

#    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)

#    ptc()              # -- pseudo-terminal controller device
#    pts()              # -- pseudo-terminal slave device
#    pmt()              # -- pseudo-magtape device
#    log()              # -- Streams logger pseudo-device
#    prf()              # -- profiler pseudo-device


##### Typical AViiON 400 series workstation configuration:

#    kbd()              # -- keyboard
#    grfx()             # -- graphics display
#    sd(insc(),*)       # -- all SCSI disk drives on integrated SCSI adapter
#    st(insc(),*)       # -- all SCSI tape drives on integrated SCSI adapter
#    inen()             # -- integrated Ethernet controller
#    duart()            # -- integrated Duart terminal line controller
#    duart(1)           # -- second Duart (if present on system)
#    lp()               # -- integrated line printer controller

#    ptc()              # -- pseudo-terminal controller device
#    pts()              # -- pseudo-terminal slave device
#    pmt()              # -- pseudo-magtape device
#    log()              # -- Streams logger pseudo-device
```

```
#     prf()               # -- profiler pseudo-device

##### Typical AViiON 4000 series server configuration:

#     sd(insc(),*)        # -- all SCSI disk drives on integrated SCSI adapter
#     st(insc(),*)        # -- all SCSI tape drives on integrated SCSI adapter
#     sd(cisc(),*)        # -- all SCSI disk drives on Ciprico SCSI adapter
#     st(cisc(),*)        # -- all SCSI tape drives on Ciprico SCSI adapter
#     cird()              # -- Ciprico Rimfire or SMD disk controller

#     inen()              # -- integrated Ethernet controller
#     hken()              # -- Interphase VME Ethernet controller
#     syac()              # -- Systech terminal line controller
#     duart()             # -- integrated Duart terminal line controller
#     duart(1)            # -- second Duart
#     lp()                # -- integrated line printer controller

#     ptc()               # -- pseudo-terminal controller device
#     pts()               # -- pseudo-terminal slave device
#     pmt()               # -- pseudo-magtape device
#     log()               # -- Streams logger pseudo-device
#     prf()               # -- profiler pseudo-device


##### Typical AViiON 5000 or 6000 series server configuration:

     cird()              # -- Ciprico Rimfire or SMD disk controller
#     sd(cisc(),*)        # -- all SCSI disk drives on Ciprico SCSI adapter
     st(cisc(),*)        # -- all SCSI tape drives on Ciprico SCSI adapter
     syac()              # -- Systech terminal line controller
     duart()             # -- integrated Duart terminal line controller
#     hken(0)             # -- 1st Interphase VME Ethernet controller
#     hken(1)             # -- 2nd Interphase VME Ethernet controller
     lp()                # -- integrated line printer controller

     ptc()               # -- pseudo-terminal controller device
     pts()               # -- pseudo-terminal slave device
     pmt()               # -- pseudo-magtape device
     log()               # -- Streams logger pseudo-device
     prf()               # -- profiler pseudo-device

#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# Protocols:
#
# List all protocols in this section, one entry per line.
# Each entry consists of the name of a protocol you want to
# configure into your system.
#
# You should not have to specify any additional protocols in order to
# use this product.
#
#
#    Protocol Name
#    -------------
#

#
#--------------------------------------------------------------------


#--------------------------------------------------------------------
# STREAMS Modules:
#
# List all explicit STREAMS modules in this section, one entry per line.
```

```
# Each entry consists of the name of a streams module you want to
# configure into your system and that has not already been implicitly
# configured because of protocols you have specified.
#
# It is recommended that you specify the Transport Provider Interface
# STREAMS modules, timod and tirdwr.
#
#
#   STREAMS Module Name
#   -------------------
#
        timod
        tirdwr
#
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# Tuneable Configuration Parameters:
#
# List all configuration parameters you wish to override in this
# section, one entry per line.
# The default values from the master file will be used unless
# explicitly overridden in this file.
#
# Each entry consists of the name of a parameter you want to
# override, followed by the value you wish to assign to it.
# If you list just the name of the parameter but not a value for it,
# its Implied Value from the master file will be used.
#
# You should set the TZ variable to accurately reflect your timezone
# (300 minutes west of GMT is USA Eastern time).
#
# You should set the MAXUP variable to the maximum number of processes
# that each user will be allowed to run simultaneously.  This number
# should be at least 64 for workstations.
#
# You should set the NODE variable to control your nodename for uname(1)
# and uucp(1), but not more than 255 characters.
#
# You should set the DUMP variable to the name of the tape device (in
# DG/UX Common Device Specification Format) that will be the default
# device to take dumps in case of system emergencies.  For diskless
# workstations, the DUMP variable should be set to the network device
# used to boot the machine.
#
# If your system is a diskless workstation, you should set the
# PERCENTNFS variable to 100 in order to get the best possible NFS
# performance.
#
# If either your system's root file system or its swap file will be
# mounted over NFS (a diskless workstation will NFS-mount both, a
# dataless workstation will NFS-mount only the root), you must set
# the NETBOOTDEV variable to the name of the network device (in DG/UX
# Common Device Specification Format) that will be used in booting
# over the network.
#
# If your system's root file system will be mounted over NFS (as will
# be done on both diskless and dataless workstations), you must set the
# ROOTFSTYPE variable to NETWORK_ROOT.
#
# If your system's swap file will be mounted over NFS (as will be done
# on diskless workstations), you must set the SWAPDEVTYPE variable to
# NETWORK_SWAP.
#
#
#   Parameter Name              Value
#   --------------              -----
#
        TZ                      300
```

```
        MAXUP                   64
        NODE                    "aviion1"

        DUMP                    "st(cisc(),4)"
### DUMP                        "inen()"

### PERCENTNFS                  100
### NETBOOTDEV                  "inen()"
### ROOTFSTYPE                  NETWORK_ROOT
### SWAPDEVTYPE                 NETWORK_SWAP

#
```

Installing the New Kernel

```
Ready to Configure a Kernel? [yes]  ↵

sysadm will now run config on /usr/src/uts/aviion/Build/system.aviion

Config succeeded.

sysadm will now attempt to build a kernel.
Building...
The build succeeded.

Install the New Kernel? [no] y↵
For a Diskless Client of this Host? [no]  ↵
Kernel Pathname? [/dgux.aviion]  ↵

The new kernel has been copied to /dgux.aviion.
Link /dgux to the New Kernel? [yes]  ↵

The new kernel will not take effect until you shutdown and reboot.
To do this, quit sysadm, and say:

  cd /
  /etc/shutdown
  /etc/halt -q

Until you do this, a few commands which depend on the symbol table
in /dgux (such as the kernel profiler and netstat) may not work correctly.
This should not cause any serious difficulties.
#
```

Bringing Down the System

```
# cd /↵
# /etc/shutdown -g0 -y↵
  ...
# /etc/halt -q↵
```

Step 18: Setting Default Boot Characteristics

```
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 menu configuration
```

```
6. Change testing parameters

7. Return to previous screen

Enter choice(s) -> 1 ↵


Change boot parameters

1 Change system boot path

2 Change diagnostic boot path

3 Change data transfer mode [BLOCK]

4 Return to previous screen

Enter choice(s) -> 1 ↵


System boot path  = []

Do you want to modify the boot path? [N] y↵

Enter new system boot path -> cimd(0,0)root:/dgux ↵

System boot path = [cimd(0,0)root:/dgux]
Do you want to modify the boot path? [N] ↵

Do you want to boot? [N] n↵
```

Rebooting the System

```
SCM> b ↵
```

Bring the System up to Run Level 1

```
# init 1 ↵
```

Changing the Default Initial Run Level

We change the default initial run level by editing the **/etc/inittab** file and changing this line:

```
def:s:initdefault:
```

To this line:

```
def:3:initdefault:
```

Step 19: Starting System Administration

Adding Groups

Add the desired groups for the user(s) on this system. See Chapter 14 for information on adding user groups.

Adding User Accounts

Add the desired users with their home directories in **/udd/aviion1**. See Chapter 14 for information on adding users.

Setting Up Terminals

See Chapter 10 for adding terminals to the system.

Starting the Accounting System

See Chapter 15 for information on setting up the accounting package.

Adding Lineprinters

See Chapter 11 for information on setting up printers.

Bring the system up to multi-user mode

At this point we can bring our system up to run level 3.

```
# init 3 ↵
```

Phase 4: Adding OS Releases and Clients

This phase is not necessary for our example.

End of Examples

Resource Planning Worksheet (Standalone System and OS Server System)

1. Your computer's role; check one: (Phase 1: Step 1)

   ☐ Standalone system — Any computer system that has its own disk containing the operating system that it uses, but which does not make an operating system image available to other systems. It may or may not be connected to a local area network; if it is not, it must have a tape device.

   ☐ OS Server system — Any computer system that has its own disk containing a bootable operating system image and file system space that are provided to client systems on a local area network. (Also complete the planning worksheet for an OS client.)

2. DG/UX™ system revision number: _____ (Phase 1: Step 1)
   DG/UX™ system update number (if applicable): _____
   Example: UD-1, Rev. 4.10

   (If loading from tape, check the tape's label; if preloaded, check the screen during automatic booting process.)

3. AViiON™ System package type: (Phase 1: Step 1)

   (If loading from tape, check the tape's label; if preloaded, see the Release Notice. Check the appropriate package and products.)

System Software (Client-Server) ☐	**Operating System** ☐	**Network Computing** ☐
☐ DG/UX™ ☐ GNU-C ☐ DG/UX™ DTK ☐ DG/UX™ X Windows ☐ Looking Glass® ☐ DG/UX™ TCP/IP ☐ ONC™/NFS® ☐ OSF/Motif™	☐ DG/UX™ ☐ GNU-C ☐ DG/UX™ DTK	☐ DG/UX™ X Windows ☐ Looking Glass® ☐ DG/UX™ TCP/IP ☐ ONC™/NFS® ☐ OSF/Motif™

Third party packages:

☐ Other _____

☐ Other _____

4. Source of system software: (Phase 1: Step 1)

   ☐ Preloaded on disk at factory

   ☐ Load from tape

5. AViiON Series computer type: (Phase 1: Step 2)

   ☐ 200 or 300 (e.g., *300, 310, 310C, 310CD*)

   ☐ 400 (e.g., *402, 410, 412*)

   ☐ 3000 or 4000 (e.g., *3200, 4020, 4120*)

   ☐ 5000 or 6000 (e.g., *5100, 5220, 6120, 6200*)

6. Ethernet address (colon-separated): _____ (Phase 1: Step 2)
   Example: 8:0:1B:18:D:D8

7. Device information: (Phase 1: Step 2)

   a. Graphics, keyboard, LAN board, parallel printer, terminal controller board, and mouse/RS232 ports.

      (Check applicable devices.)

Pick Devices	Device Type	Device Name	Controller #	Device Specification
☐	graphics monitor	grfx	NA	grfx()
☐	keyboard	kbd	NA	kbd()
☐	integrated LAN	inen	NA	inen()
☐	parallel printer	lp	NA	lp()
☐	asynch terminal controller board	syac	0	syac()
☐	asynch terminal controller board	syac	1	syac(1)
☐	synch terminal controller board	sdcp	NA	sdcp()
☐	mouse/RS232 port	duart	0	duart()
☐	second RS232 port	duart	1	duart(1)
☐	Hawk integrated LAN	hken	0	hken()
☐	Hawk integrated LAN	hken	1	hken(1)

NA = not applicable

b. Disk and tape devices; (Check applicable devices and supply other information.)

Pick Devices	Device Type	Device Name	Con Name	Con #	Device #	Device Specification (1)
☐	*Examples: system disk* *tape device*	*sd* *NA*	*insc* *cimd*	*0* *0*	*0* *4*	*sd(insc(),0)* *cimd(0,4)*
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico SMD controller	NA	cimd			
☐	device on Ciprico SMD controller	NA	cimd			
☐	device on Ciprico SMD controller	NA	cimd			
☐	device on Ciprico SMD controller	NA	cimd			

NA = not applicable; device = disk or tape

8. Physical disk size: (Phase 1: Step 2)

Locate your device's model number and note its corresponding size (in blocks).

Disk Model Number	Approx Size (MBytes)	Approx Size (Blocks)
5070DR Optical Disk	2458	5033164
5070S Optical Disk	900	1843200
6442 ESDI (full-height)	327	669696
6491 SCSI (full-height)	322	659456
6539 SCSI (half-height)	179	366592
6541 SMD	1066	2183168
6542 SMD	2132	4366336
6554 SCSI (full-height)	662	1355776
6555 ESDI (full-height)	648	1327104
6627 CD ROM	590	1208329
6627 CD ROM	650	1331200
6661 ESDI (half-height)	330	675840
6662 SCSI (half-height)	322	659456
6685 SCSI (full-height)	1040	2129920
6740 SCSI (full-height)	1040	2129920

NOTE: SMD disks apply to AViiON 6000 series systems only.

To determine your disk sizes, enter your device specification (1) from the Device information table (Table 7b), and its associated model number (2) with approximate size in blocks (3) from the Physical disk size table (Table 8).

Device Specification (1)	Disk Model Number (2)	Approx Size (Blocks) (3)
Example: sd(insc(),0)	*6539 SCSI (half-height)*	*349609*
	TOTAL DISK CAPACITY	

9. Logical disk unit (LDU) allocation: (Phase 1: Step 4)

What Is Loaded?	Example LDU Name	Actual LDU Name (4)	Example Block Size	Actual Block Size (5)
*DG/UX system	*root*		*40000*	
*DG/UX system	*usr*		*160000*	
*swap space	*swap*		*50000*	
temp space	*var_tmp*		*20000*	
Other; X11	*usr_opt_X11*		*105000*	
Other	*local*		*40000*	
Other	*var_mail*		*20000*	
Other	*home*		*200000*	
Other				
Other				
Other				
TOTAL			*835000*	

* denotes that these LDUs are required.

Check your release notice(s) for information about the size requirements for other packages.

Provide the total blocks used per LDU in the next table.

Actual LDU Name	Total Blocks Used
Example: usr_opt_X11	*105000*

10. OS Client logical disk unit (LDU) allocation: (Phase 1: Step 4)

    This table applies to OS clients only.

What Is Loaded?	Example LDU Name	Actual LDU Name (4)	Example Block Size	Actual Block Size (5)
Client-server only	*srv*		*5000*	
Client dump space	*srv_dump*		*40000*	
Client root space	*srv_root*		*120000*	
Client swap space	*srv_swap*		*175000*	
Client secondary root space	*root_420*		*120000*	
Client secondary usr space	*usr_420*		*175000*	
TOTAL			*635000*	

Provide the total blocks used per LDU in the next table.

Actual LDU Name	Total Blocks Used
Example: srv_root	*120000*

11. Logical disk unit (LDU) mount points: (Phase 1: Step 4)
(Record your LDU names (4) and actual block sizes (5) from Table 9 into corresponding columns in this table.)

Disk Name	LDU Name (4)	Piece Size (in Blocks) (5)	Piece #	Actual Mount Point
Example: sd(insc(),2)	*home*	*200000*	*1*	*/home*
	*root		*1*	**root**
	***usr**		*1*	**/usr**
	***swap**		*1*	NA

* denotes that these LDUs are required, NA = not applicable

NOTE: LDUs **root** and **/usr** cannot be broken into pieces. All others can.

12. Logical disk unit (LDU) mount points for OS clients: (Phase 1: Step 4)
This table applies to OS clients only.

Disk Name	LDU Name (4)	Piece Size (in Blocks) (5)	Piece #	Actual Mount Point
Example: sd(insc(),2)	*srv_swap*	*175000*	*1*	*/srv/swap*

NOTE: LDUs **srv_root** and **srv_swap** cannot be broken into pieces. All others can.

13. For TCP/IP: name of remote shell and restricted shell:

    If you are installing a DG/UX system within an existing environment, be sure to choose the naming convention that is consistent with your environment. Consult your system administrator.

    /bin/rsh is the restricted shell (SVID compliant).

    /bin/rsh is the remote shell (as in BSD-based systems).

    Unimportant; accept the default **/bin/rsh** as the remote shell.

14. Network information: (Phase 1: Step 6)

Networking Information	Example Information	Actual Information
Host's name	*simon*	
Internet address	*123.227.2.14*	
Network name	*my_net*	
Subnet status	*yes*	
Network mask	*0xffffff00*	
Controller device name	*inen0*	
Broadcast address type	BSD 4.3 *(all ones)* BSD 4.2 *(all zeroes)*	
Yellow Pages domain name	*my_domain*	

End of Standalone System and OS Server System Worksheet

Resource Planning Worksheet (OS Client)

Your computer's role: OS Client system — any computer system that boots its operating system from an OS server system on a local area network. (You should also complete the worksheet for the OS server.)

1. AViiON Series computer type: (Phase 1: Step 2)

   ☐ 200 or 300 (e.g., *300, 310, 310C, 310CD)*

   ☐ 400 (e.g., *402, 410, 412*)

2. Ethernet address (colon-separated): _____ (Phase 1: Step 2)
   Example: 8:0:1B:18:D:D8

3. Device information:

   a. Graphics, keyboard, LAN board, parallel printer, terminal controller board, and mouse/RS232 ports.

      (Check applicable devices.)

Pick Devices	Device Type	Device Name	Controller #	Device Specification
☐	graphics monitor	grfx	NA	grfx()
☐	keyboard	kbd	NA	kbd()
☐	integrated LAN	inen	NA	inen()
☐	parallel printer	lp	NA	lp()
☐	asynch terminal controller board	syac	0	syac()
☐	synch terminal controller board	sdcp	NA	sdcp()
☐	mouse port	duart	0	duart()
☐	RS232 port	duart	1	duart(1)

NA = not applicable

b. Disk and tape devices.

(Check applicable devices, supply controller number and disk number, and supply device specification.)

Pick Devices	Device Type	Device Name	Con Name	Con #	Device #	Device Specification (1)
☐	*Examples: system disk* *tape device*	*sd* *NA*	*insc* *cimd*	*0* *0*	*0* *4*	*sd(insc(),0)* *cimd(0,4)*
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	SCSI device on integrated SCSI adapter	sd or st	insc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico SCSI adapter	sd or st	cisc			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico ESDI controller	NA	cied			
☐	device on Ciprico ESDI controller	NA	cied			

NA = not applicable
device = disk or tape

4. For TCP/IP: Name of remote shell and restricted shell:

   If you are installing a DG/UX system within an existing environment, be sure to choose the naming convention that is consistent with your environment. Consult your system administrator.

   - **/bin/rsh** is the restricted shell (SVID compliant).
   - **/bin/rsh** is the remote shell (as in BSD-based systems).
   - Unimportant; accept the default **/bin/rsh** as the remote shell.

5. Network information:

Networking Information	Example Information	Actual Information
Host's name	*alvin*	
Internet address	*123.227.2.14*	
Network name	*my_net*	
Subnet status	*yes*	
Network mask	*0xffffff00*	
Controller device name	*inen0*	
Broadcast address type	BSD 4.3 *(all ones)* BSD 4.2 *(all zeroes)*	
Yellow Pages domain name	*my_domain*	

End of OS Client Worksheet

End of Chapter

Index

Note: Boldfaced page numbers (e.g., **1-5**) indicate definitions of terms or other key information.

A

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

X

Y