Data General

Customer Documentation

Installing and Managing the DG/UX™ System

AViiON™
PRODUCT LINE

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

DATA GENERAL CORPORATION
4400 Computer Drive
Westboro, MA 01580

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

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 shows you how to do a first-time installation of the DG/UX™ operating system and how to operate and manage the 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.
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

Owner's Manual for Stand-Alone AViiON Workstation with a Hard Disk (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.

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.

If you have comments on this manual, please use the prepaid Comment Form that appears at the back. We want to know what you like and dislike about this manual.

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 1
The Big Picture: Managing DG/UX™ Systems

You have opened this manual because you need to know how to manage a DG/UX™ system. The DG/UX operating system supports diskless client systems in an OS server-client environment. An OS server system uses the Transmission Control Protocol/Internet Protocol (TCP/IP) and the Network File System (NFS®) to provide an operating system to remote client systems. This scenario involves several sets of tasks, which can be done by the OS server administrator or shared among OS client administrators.

This manual will help you manage a DG/UX system running on the following:

- **Stand-alone system** – this could be a multiuser system that supports conventional terminals or a workstation that boots from its own disk.

- **OS Client system** – a workstation that boots its operating system from an OS server system via TCP/IP and NFS.

- **OS Server system** – a system that provides a bootable DG/UX image and file system space to diskless client systems via TCP/IP and NFS.

The chapter section called "Using the sysadm Utility" explains how to use the **sysadm**(1M) program for routine system administration tasks, and gives general information about this manual. If you are managing a stand-alone system, and you prefer to skip information about OS servers and clients, you can just read that section of the chapter. After that, you are ready to move on to the installation material in the next chapter.

A later section, "Managing Servers and Clients," introduces the concept of a servnet and discusses the aspects of the server and client relationship.

Administrative Roles

A DG/UX system administrators' tasks fall into categories based on the kinds of systems they have: a stand-alone, an OS server, or an OS client. The next three sections address what tasks might be done on each system. You do not necessarily have to divide the tasks this way in your own environment; the scheme discussed here is just an example.

Tasks on Stand-Alone and OS Server Systems

A stand-alone multiuser system provides one operating system release at a time to many users at terminals. A stand-alone workstation usually provides one operating system release for a single user.

An OS server can provide one or more releases to a number of client systems. System management tasks done from a stand-alone or an OS server system can include:

- planning and installing the system
- formatting disks
- creating file systems
- starting and stopping the system
- reconfiguring kernels
- settting up printers and terminals
- doing system backups
- running the accounting programs
- managing user accounts
- managing the network
- setting up product parameter files

Tasks on a Data General OS Client

We'll rely on certain assumptions throughout this manual. Let's assume that client systems in this manual are workstations. The OS client gets its operating system from the OS server. We'll assume that OS clients have neither disks nor tape drives. We'll also assume that most system administration work will be done from the OS server. Remember, these are assumptions for the sake of example. You could have one server as the client of another server. A client could get its operating system from one or more servers on the network. A client could have its own disk (for swapping, storage, etc.) or tape drive.

Tasks that can be done from the OS client are

- setting up network parameters
- booting over the network from an OS server
- mounting file systems from an OS server and/or other hosts via NFS
- reconfiguring a different kernel than the OS server's
- running accounting programs

Tasks on a Foreign OS Client

A Data General OS server can provide operating system service to a foreign (non-Data General) OS client, but the client is largely responsible for its own administration. Notable among the foreign client's responsibilities are reconfiguring the kernel, setting up and managing networking software, and using its own operating system documentation.

To take advantage of the **sysadm** utility for foreign clients, you must set up a **srv** directory structure for the foreign client systems. Chapter 2 shows how you can do this in "File System Mapping: DG/UX to Foreign."

Using the sysadm Utility

The **sysadm** program is a menu-driven set of system administration procedures that can be used by all three categories of administrator: stand-alone, server, or client. Menu screens with interactive queries help you choose and execute commands. When you select a function, represented by a menu selection, the **sysadm** program passes control to that function. Then the function queries you for information, confirms the information you supply, acts on your request, and returns control to **sysadm**.

To exit from any menu at any time, type **q** to return to the shell.

Help?

You can call up a HELP script on any menu item by typing the item number followed by a question mark, **?**. Also, whenever you don't understand a query, type **?** and press the New Line key. The **sysadm** program will print definitions or instructions about the particular query.

Using diskman

In addition to **sysadm**, you will be using another utility called **diskman**, which comes in two versions. Sites without preloaded disks will boot the *stand-alone* version (**/usr/stand/diskman**) from tape to begin installation by loading the starter system components. Then with the starter system loaded, preloaded and non-preloaded systems are the same. From that point on, installation continues the same way for both kinds of systems--you use the *stand-among* version of **diskman** (**/usr/sbin/diskman**) for tasks such as creating logical disks, creating file systems, deleting logical disks, and obtaining information on physical and logical disks. You invoke stand-among **diskman** via the **sysadm diskmgmt** command. With **diskman**, you choose from menus structured similar to those of **sysadm**. See Chapter 7 for detailed information on **diskman**.

Using sysadm Commands

You can execute the **sysadm** command only when using the **root** login or the **sysadm** login. We recommend that you rely on the **sysadm** login instead of the **root** login. The system must also be at run level 1 or higher. The **sysadm** login has **/admin** as its home directory, while the **root** login has **/** as its home directory. The **sysadm** login ensures that your personal administrative files (such as **.login** or **.profile**) are kept out of the **/** directory; this results in a "cleaner" **/** directory. You can invoke the **sysadm** command alone or with an argument. When you type **sysadm** without an argument, the DG/UX system displays the following top-level menu:

```
                    SYSADM MAIN MENU

 1 diskmgmt         Enter the diskman program
 2 sysmgmt          System configuration management menu
 3 fsmgmt           File system management menu
 4 fileinfo         File information menu
 5 ttymgmt          TTY management menu
 6 lpmgmt           Line Printer management menu
 7 usermgmt         User management menu
 8 uucpmgmt         UUCP management menu
 9 networkmgmt      Network management menu
10 releasemgmt      Software release management menu
11 clientmgmt       Diskless and X terminal client management menu

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, q to QUIT:
```

Table 1-1 shows how you interact with the **sysadm** program. Note that the " ^ " character is not valid on the Main Menu. If you type it, you will be prompted again to choose from the list of menu options.

Table 1-1 How to Use sysadm Menus

User Input	Description
number	Select a command by number.
name	Select a command by name.
?	Print a HELP message to the screen.
!*command*	Execute any DG/UX command then return to **sysadm**.
q	Exit from **sysadm** and return to the shell.
^	Return to previous menu.
New Line	Select the default. Defaults are in brackets.

NOTE: The interrupt character (<Ctrl-C>) and the EOF character (<Ctrl-D>) are disabled while you are using **sysadm**. They are reenabled when you exit from **sysadm** or start a subshell.

Now you know what **sysadm** is and you know the rules for accessing it. Let's look at the menus and see exactly what you have to work with.

The sysadm Menu Set

This section shows you the menus that are displayed as you select items from the **sysadm** Main Menu.

diskmgmt: Enter the Diskman Program

Use **diskman** when you need to manipulate physical and logical disk units. Chapter 7 discusses the tasks in this menu in more detail.

```
              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
```

sysmgmt: System Configuration Management Menu

Use **sysmgmt** for miscellaneous system administration tasks. Chapter 4 discusses the tasks in this menu in more detail.

```
                System Configuration Management

1 datetime      Set date, time, time zone, daylight savings time
2 newdgux       Build and install a new DG/UX kernel
3 tapedefaults  Set defaults for tape use

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

fsmgmt: File System Management Menu

Use **fsmgmt** to manage your file systems. Chapter 8 discusses the tasks in this menu in more detail.

```
                    File System Management

 1 addfsys          Add an entry to the list of file systems
 2 delfsys          Delete an entry from the list of file systems
 3 lsfsys           List the available file systems
 4 modfsys          Modify an entry in the list of file systems
 5 addswap          Add a swap entry to the list of file systems
 6 delswap          Delete a swap entry from the list of file systems
 7 mountfsys        Mount a file system
 8 unmountfsys      Unmount a file system
 9 modcycle         Modify the current position in the dump cycle list
10 fsdump           Make backup tapes using dump
11 fsrestore        Restore a complete file system from fsdump tapes
12 filerestore      Extract a few files from fsdump tapes

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

fileinfo: File Information Menu

Use **fileinfo** to locate files and monitor disk space usage. Chapter 9 discusses the tasks in this menu in more detail.

```
                    File Information

 1 diskuse          Show free blocks on mounted file systems
 2 fileage          Locate idle files in a directory tree
 3 filename         Locates files by name in a directory tree
 4 filescan         Locate files with possible permission errors
 5 filesize         Locate very large files in a directory tree

 Enter a number, a name, the initial part of a name,
 ? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

ttymgmt: TTY Management Menu

Use **ttymgmt** to manage your system's terminal lines. Chapter 10 discusses the tasks in this menu in more detail.

```
                TTY Management

1 ttydefaults   Define tty default settings
2 addtty        Add a single tty entry
3 deltty        Delete a tty entry
4 modtty        Modify a tty entry
5 lstty         List tty entries
6 installtty    Add multiple tty entries

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

lpmgmt: Line Printer Management Menu

Use **lpmgmt** to manage your system's printer queues. Chapter 11 discusses the tasks in this menu in more detail.

```
                   Line Printer Management

 1 addlp          Define a new printer
 2 dellp          Delete a printer
 3 modlp          Modify an existing printer
 4 lslp           List printers
 5 defaultlp      Define the default printer
 6 acceptlp       Set a printer to accept print requests
 7 rejectlp       Set a printer to reject print requests
 8 enablelp       Enable a printer
 9 disablelp      Disable a printer
10 queuelp        Display the print queue of a printer
11 cancellp       Cancel print requests
12 movelp         Move print requests from one printer to another
13 startlp        Start the lp scheduler
14 stoplp         Stop the lp scheduler

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

usermgmt: User Management Menu

Use **usermgmt** to manage the user and group profiles and mail aliases on your system. Chapter 14 discusses the tasks in this menu in more detail.

```
                User Management

 1 userdefaults   Set user account defaults
 2 adduser        Create a user account
 3 deluser        Delete a user account
 4 moduser        Modify a user account
 5 lsuser         List user account information
 6 addgroup       Add group entries
 7 delgroup       Delete group entries
 8 modgroup       Modify group entries
 9 lsgroup        List group entries
10 addalias       Add mail alias entries
11 delalias       Delete mail alias entries
12 modalias       Modify mail alias entries
13 lsalias        List mail alias entries

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

uucpmgmt: UUCP Management Menu

Use **uucpmgmt** to manage the UUCP subsystem. Chapter 12 discusses the tasks in this menu in more detail.

```
                UUCP Management

 1 addsystem     Add an entry to the UUCP Systems file
 2 delsystem     Delete an entry from the UUCP Systems file
 3 modsystem     Modify an entry in the UUCP Systems file
 4 lssystem      List entries in the UUCP Systems file
 5 adddevice     Add an entry to the UUCP Devices file
 6 deldevice     Delete an entry from the UUCP Devices file
 7 moddevice     Modify an entry in the UUCP Devices file
 8 lsdevice      List entries in the UUCP Devices file
 9 addpoll       Add an entry to the UUCP Poll file
10 delpoll       Delete an entry from the UUCP Poll file
11 modpoll       Modify an entry in the UUCP Poll file
12 lspoll        List entries in the UUCP Poll file
13 trysystem     Test a UUCP connection

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

networkmgmt: Network Management Menu

Use **networkmgmt** to manage your network. Chapter 13 discusses the tasks in this menu in more detail.

```
                Network Management

 1 addhost        Add an entry to the hosts file
 2 delhost        Delete an entry from the hosts file
 3 modhost        Modify an entry in the hosts file
 4 lshost         List entries in the hosts file
 5 addnetwork     Add an entry to the networks file
 6 delnetwork     Delete an entry from the networks file
 7 modnetwork     Modify an entry in the networks file
 8 lsnetwork      List entries in the networks file
 9 addether       Add an entry to the ethers file
10 delether       Delete an entry from the ethers file
11 modether       Modify an entry in the ethers file
12 lsether        List entries in the ethers file
13 nfsparams      Set boot time parameters for NFS and YP
14 tcpipparams    Set boot time parameters for TCP/IP

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

releasemgmt: Software Release Management Menu

Use **releasemgmt** to manage your software releases. Chapter 5 discusses the tasks in this menu in more detail.

```
            Software Release Management

 1 addrelease       Add a software release area
 2 delrelease       Delete a software release area
 3 lsrelease        List information about software releases
 4 loadpackage      Load software packages into a software release area
 5 setuppackage     Set up packages in a software release area
 6 makesrv          Create the initial /srv directory tree
 7 lstoc            List the table of contents from a release tape

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

clientmgmt: Diskless and X Terminal Client Management Menu

Use **clientmgmt** to manage your OS and X terminal clients. Chapter 6 discusses the tasks in this menu in more detail.

```
              Diskless Client Management


 1 addclient        Add a diskless client entry
 2 clientdefaults   Create or modify a set of diskless client defaults
 3 delclient        Delete a diskless client entry
 4 lsclient         List information about diskless clients
 5 bootdefault      Change the default release for a diskless client
 6 addxterminal     Add an X terminal display bootstrap client
 7 delxterminal     Delete an X terminal display bootstrap client
 8 lsxterminal      List X terminals that are served by this system

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Invoking sysadm with Arguments

Earlier, we mentioned that you could invoke **sysadm** with an argument from the command line. Any of the selections from any menu can be arguments. In the following example, we use **sysmgmt**, but we could just as easily have used **datetime** as an argument. Let's look at the example. From the shell, you can bypass the Main Menu and go directly to the **sysmgmt** menu by typing:

```
# sysadm sysmgmt ♪
```

Your screen displays the following:

```
                  System Configuration Management

 1 datetime       Set date, time, time zone, daylight savings time
 2 newdgux        Build and install a new DG/UX kernel
 3 tapedefaults   Set defaults for tape use

 Enter a number, a name, the initial part of a name,
 ? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

At this level, **sysadm** waits for your next choice. When you have completed the procedure that you selected, **sysadm** displays this prompt again:

```
Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

If you respond by typing ^ (the GO BACK option) and pressing New Line, **sysadm** returns you to the **sysmgmt** menu. When you invoke **sysadm** for a specific menu, however, you cannot use ^ to return to any menu higher than the one you invoked. For example, if you were to specify **datetime** on the **sysadm** command line, you would not be able to return to the Main Menu or the System Configuration Management menu.

Shell Escapes

You can escape to the shell from **sysadm** by typing **!** followed by the shell of your choice (**sh** or **csh**) or any other command. When the command terminates or when you exit from the shell, you return to **sysadm**. If you start a new shell this way, remember that your **sysadm** session continues to run in your previous shell. Do not begin a new **sysadm** session while the old one is still running; instead, return to your original **sysadm** session by terminating your new shell by pressing Ctrl-D or typing **exit**.

Using sysadm Procedures

In each chapter, we'll show examples of each **sysadm** procedure in the order it appears on its main menu. A chart precedes each procedure giving instant information on the purpose of the procedure, commands to use, location of those commands in the **sysadm** scheme, and any special instructions or references. Use these charts as a quick check that you have chosen the correct procedure before starting. The charts follow this style:

Purpose	Summary of the procedure's use.
Starting Conditions	The run level the OS should be in when you begin the procedure. Any special login requirements.
sysadm Menu	The part of the menu package that contains the commands to perform the procedure.
Commands	The commands used to perform the procedure.
Special Instructions References	Reference pages that you may want to consult. Host requirements if necessary: OS server, OS client, NFS server, or YP server.

Command References

Sometimes we will refer you to manual reference pages, or man pages. Each man page describes a DG/UX command, system call, file, or program. A man page has a *name* and a *section number*. The reference **wall**(1M), for example, refers to a command that means "write to **all**", a way of sending a broadcast message to all users. The (1M) tells you in what reference manual the command is found. The numbered sections are located in the following manuals:

Table 1-2 Reference Manuals for the DG/UX System

Section Number	Reference Manual
(1)	*User's Reference for the DG/UX™ System*
(1M), (7), (8)	*System Manager's Reference for the DG/UX™ System*
(1), (2)	*Programmer's Reference for the DG/UX™ System, Volume1*
(3), (4), (5), (6)	*Programmer's Reference for the DG/UX™ System, Volume 2*

Users can display any reference page on their screens with the **man** command. Here's how a user would display the **wall** man page:

```
$ man wall ↵
```

This command gives you the same information as the printed reference page.

Sections for Experts

At the end of some chapters in this manual, you will find an *expert* section, such as Chapter 10's "Expert TTY Information." We use the word *expert* only to point out that you do not have to read these sections to manage your system; **sysadm** is designed to guide you through any administrator's task. Enter these expert areas if you need to deviate from **sysadm** or to learn more about the workings of the DG/UX system.

You should be aware that **sysadm** imposes certain restrictions on the use of file structures. Say, for instance, you're the administrator of a stand-alone system and you use the **sysadm addtty** command for adding terminals. This command writes entries to the **inittab**(4) file in a specific format that the **sysadm** program looks for and uses when you invoke other **sysadm** commands such as **lstty** and **deltty**. If, for some reason, you were to use a text editor and make a tty entry in a slightly different format, the terminal would function normally, but the **sysadm** program would be unaware of that terminal's existence. So, if you're expert enough to make such changes to various files, then we want you to have the information to ensure that your actions result in changes that the **sysadm** program can use. For our example above using **addtty**, you need to know that entries that are written to the **inittab** file look like the following three entries:

```
rc4:4:wait:/etc/rc.init  4  >  /dev/console 2>&1
00:23:respawn:/etc/getty  tty00  9600  vt100
01:23:off:/etc/getty  tty05  9600  vt100
```

In the expert sections, you may see details on files, directories, commands, alternate procedures, shell programs, subsystems, etc., depending on the subject matter of the chapter. But remember, these sections are optional; you do not need to read them to manage your system.

Managing Servers and Clients

We have made a distinction between systems using the terms stand-alone, OS servers, and OS clients. Each has its own manager, but how these managers cooperate is a matter of negotiation. Our examples throughout this manual will assume that the manager of the OS server system does the bulk of administration work for his system and for his client's systems. The OS server runs TCP/IP and NFS. The YP database is maintained on the OS server system. You may choose to do things differently.

Terms

To begin understanding the server and client relationship, let's look at some terms we'll be using in this manual.

host — refers to any system: stand-alone, server, or client.

stand-alone — a system with its own disks. Some stand-alones support dumb terminals in a traditional multiuser environment where all terminals are running the same release. A stand-alone does not need TCP/IP or NFS to service its terminals, and it has no OS clients. A workstation with its own disk is also a stand-alone.

OS server — refers to a host providing an operating system and disk space for client systems. Servers can be homogeneous or heterogeneous.

A homogeneous OS server provides a *single system release* to many other clients which all share a single copy of some of the system release files (**usr**), but which also maintain private copies of other system release files (**root**). Processing is done on each client's processor.

A heterogeneous OS server provides *many system releases* to many clients. Clients share release software (**usr**) and maintain private files (**root**) the same way clients of homogeneous OS servers do; clients have the ability to boot other OS releases supported by the server if the other releases are compatible with client hardware.

Note that there are other kinds of servers besides OS servers. A Yellow Pages (YP) server provides YP database information to YP clients. An NFS server provides file system access to remote NFS clients. When the term *server* is used unmodified it refers to an OS server.

client — a host which gets its system files from a disk connected to an OS server. When the term *client* is used unmodified it refers to an OS client.

diskless client — an OS client with no disks of its own. An OS client may have its own local disk but use an OS server for its system software (the traditional **/** and **usr** directories).

workstation — a system with its own processor, its own graphics terminal, and graphics software (shared or host dependent). A workstation could be an OS server, or an OS client with or without disks.

servnet — the collective unit formed by an OS server, its clients, and its releases. For example, a server supplying OS releases for two Data General workstations and one foreign workstation would be a servnet.

directory tree — refers to the **/** (root), **/usr**, **/srv**, and other directories that make up the directory space of a DG/UX system. Clients, rather than having their own file systems, use directories on the OS server for their root and **/usr** file systems. The OS server has a **root** logical disk file system and a **usr** logical disk file system which contains the software associated with the primary release. (The primary release resides in **/srv/release/PRIMARY**.) By logical disk file system, we mean a file system directly associated with a specific logical disk; in general, there is one file system per logical disk. The exceptions are logical disks used for swap space, and logical disks for which you have not yet created a file system. See Chapter 7 for more information on logical disks.

mapping — associating the elements of two different representations of a system (like a directory tree) so that a correspondence exists between the two systems. Every element in one system can be mapped to an element in the other system.

The **sysadm** program maps all of a host's **root** and **usr** directory trees through the **/srv** directory tree. Mapping all **root** and **usr** directory trees in a single, consistent way eliminates the need for conversions if a host's role changes from stand-alone, server, or client.

Because information describing the OS clients and releases is kept in the **/srv** directory tree, OS server hosts should create a separate logical disk named **srv**. Within the **srv** directory tree, the directory **PRIMARY** is used to manage primary releases. The directory **MY_HOST** is used to manage a host's **root** directory tree. Both of these names are reserved and should not be used elsewhere.

release dependent — the system commands and files that are dependent on a release. A file that can't or shouldn't be shared between releases is *release dependent*. If, for example, man pages and certain ASCII data files can be shared by more than one release, then they are *release independent*. The master files and most commands are examples of release-dependent files. Release-dependent files may or may not be host dependent.

host dependent — the commands and files that are dependent or unique to an individual host. If a given file can't or shouldn't be shared between several hosts, then it is *host dependent*. Every client host needs its own copy of host dependent files and needs to be able to

	write to its own data files. This class of files also contains the set of commands and data files required to boot a host. The **/etc/passwd** file and **.profile** are examples of host-dependent files. Host-dependent files may or may not be release dependent.
package	a set of software traditionally called a product.
release	a set of software intended for a specific architecture and operating system. A release encompasses all software required for a host. A release identity is defined by the contents of the release's **/usr** directory. Additional software packages loaded into the release's **/usr** directory become part of the release.
composed release	multiple products packaged on a single tape set that can be installed by **sysadm**. Each product tape has its own install and setup routines that can be executed at load time.
primary release	the release that the server uses. On the server, the primary release resides on the server's **root** and **usr** logical disks. Clients that use the primary release have their own root file space, but they use the same **usr** logical disk as the server. The directory structure that supports clients attached to the primary release is **/srv/release/PRIMARY**. Other directories under **/srv/release** may contain secondary releases.
foreign release	a release supplied by a vendor other than Data General. Releases supplied by Data General are called native releases.
tapeless	an OS client without a tape drive. To read or write a tape, a tapeless system must use the network to access a host that has a tape drive.

Managing in the Servnet

Now that you've read the terms, let's begin using them to show how they come together. Since you're reading this section of the chapter, we assume you are not managing a stand-alone multiuser system. You are a manager in a servnet, and you are managing either an OS server or an OS client. Let's start with the OS server.

The OS Server

All system software for releases and clients resides on the OS server's disk. The server has its own separate **/** (root) file system. It is a single-piece logical disk and contains only the server's **/** system files for the primary release. Its size is not a function of the number of clients or releases on the system; client roots reside on other logical disks. The size of the logical disks that contain client roots varies according to client requirements. The **sysadm** program accesses the server's root by the path **/srv/release/PRIMARY/root/MY_HOST**.

The server also has a separate **/usr** file system; a single-piece logical disk containing the server's primary release **usr** files. If a server runs a secondary release, the **/usr** directory tree for the secondary release is stored on a logical disk separate from the server's **root** and **usr** logical disks.

In a heterogenous servnet, there may be many **/usr** directory trees, each of which is associated with a specific release. There may also be many **/** directory trees, each of which is associated with a specific host and release.

The OS server uses **sysadm**(1M) and the **/srv** file system to manage hosts and releases. This file system contains subdirectories for releases, kernels, and client roots. It also contains data files used by the **sysadm** program in tracking and managing clients and releases.

Phase One in Chapter 2 covers the topics introduced in this section in more detail. See Appendix B, "Directories and Files", for a breakdown of the **/**, **/usr**, and **/srv** directories.

The OS Client

An OS client may or may not have its own physical disk. In either case, the OS client will still get its bootable DG/UX image from an OS server. The client may also boot more than one release on a server, or if a server is down, the client could boot from another server on the local network. So, while a client is dependent on a server for its OS, it has the independence of booting from another source if necessary. The **sysadm** program accesses the client's root directory by the path **/srv/release/PRIMARY/root/***client_hostname*.

For our examples in this manual, we'll assume that OS clients are diskless. Some clients may have their own physical disks that they might use for swapping or general storage, while still booting their operating system from a server.

As we said, tasks are arranged by negotiation. Doing backups is such a task. Tapeless client managers might arrange to access a remote tape device and do their own individual backups. Or, the server manager might do all backups on all clients at one time.

Like any other system running NFS on a network, the client acquires and uses the resources of other systems by mounting remote file systems.

Windows

Most client workstations will be running some kind of windowing program. Our example system includes a server that has a logical disk named **u_opt_x11** for the X Window System™. Users on workstations may start a windowing program from the command line, or they may be set up such that an **rc** script starts the windowing program. See Chapter 3 for information on **rc** scripts.

How an OS Client Boots

An OS client gets operating system service from a server on the local area network; we refer to this process of booting over a network as *netbooting*. Before a client can netboot, the client manager and server manager must talk and exchange information. Assuming that the OS server is already running TCP/IP and NFS, here is the sequence leading up to a client's successful netboot.

1. The OS server manager must get the Ethernet address and host name of the OS client wishing to boot. See Chapter 2 and your network documentation.

2. The OS server manager assigns an Internet address to the client host.

3. The OS server manager uses **sysadm** to assign the client to a specific OS release, using the client's Ethernet and Internet information to set the client's bootstrap files and make them accessible to the network.

4. The OS client host powers up and broadcasts its Ethernet address. The OS server responds by sending back the client's Internet address. The client host returns a request for a boot program, which is linked in the server's **/tftpboot** directory. The OS server sends the boot program to the client.

5. The OS client uses **rarp** to get its Internet address from the OS server.

6. The client uses the **bootparams** RPC service to locate the name of the NFS server that holds its **/** (root) file system. The **bootparams** database is stored in a YP map or a local file.

7. The client uses NFS to mount the remote root and **/usr** file systems. It then executes the kernel program, which starts a subprocess to bring up a default set of system services.

TCP/IP, NFS, and YP

A servnet relies on TCP/IP and NFS to provide operating system service to OS clients, so these products are shipped as part of the OS server's release package. This manual discusses TCP/IP and NFS parameters to some extent, but it does not cover the networking products completely. Refer to *Setting Up and Managing TCP/IP on the DG/UX™ System* and *Managing NFS® and Its Facilities on the DG/UX™ System* before you attempt to install either product.

Although the Yellow Pages (YP) are not absolutely necessary for managing a servnet, we recommend it because it provides an efficient database method for tracking and servicing users and systems. *Managing NFS® and Its Facilities on the DG/UX™ System* gives information on setting up YP on a server or client host. Note that the Yellow Pages can be managed from any host on the servnet.

End of Chapter

Chapter 2
Installing the DG/UX System

This chapter leads you through the initial installation of the DG/UX system. If you are installing the DG/UX system on an AViiON workstation that has its own hard disk, you may prefer to use a different Data General document, *Owner's Manual for Stand-Alone AViiON Workstation with a Hard Disk*. If you are installing the DG/UX system on a system that has a previous release of the DG/UX system already installed, 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.

The chapter first presents an installation checklist, followed by a discussion of device specifications, which you need to know at various phases of system installation.

Next, the chapter details four *sequential* installation phases. You should complete all steps in a phase, in order, before moving to the next phase. The four phases of the installation process are:

- **Phase One: Planning the Installation** -- sizing disks, naming file systems, listing devices, comparing DG/UX and foreign directory trees. You need to complete this or part of this phase *for all installations.*

- **Phase Two: Loading the Primary Release** -- using **diskman**(1M) to load the primary release and create the OS logical disks and file systems. If you have a preloaded physical disk containing the basic software configured to default values, you may skip Phase Two after completing Phase One. *Even if you have a preloaded disk, you still need to understand concepts and plan for your specific needs.*

- **Phase Three: Customizing the Primary Release** -- booting the starter system, loading software packages, building a custom kernel, setting a default boot path, creating other logical disks and file systems, adding user accounts, adding terminals, and adding printers. You need to complete this phase for all installations.

- **Phase Four: Adding OS Releases and Clients** -- adding secondary releases, adding diskless clients to a release, setting up diskless clients, and adding and deleting X terminal clients. If you will not use your system as an OS server or an X terminal server, you may skip this phase.

The last part of the chapter contains examples of installation sessions on three differently configured systems. You may refer to these examples as you perform the installation on your own system. Do not use the examples as guides; the configuration of your system may not be the same as any of the example configurations, and you may inadvertantly mis-install your system.

> Important: Read all of Phase One before starting the installation.

Before You Begin

The next section shows a 22-step installation checklist composed of the major section titles of this chapter. This is your path from start to finish. At this point you should have read Chapter 1 thoroughly; you are now familiar with the **sysadm** program, and you have a plan in mind for setting up a stand-alone host, an OS client, an OS server, or a combination of these.

Setting up a stand-alone machine (multiuser or workstation) and setting up a server machine are very similar operations. If you want to install a diskless machine on an existing server, space must be allocated on the server's disk.

The OS server host differs from the multiuser host in that it must be running networking software and must have extra logical disk space allocated for clients' root, swap, and dump purposes. The processes of setting up these different hosts have a common progression line; that is, if you take the setup for a multiuser machine and add extra logical disks plus extra networking software packages, you have an OS server. To an OS server you can add diskless clients that boot their kernels over the network. These clients can run the primary release (the same one that the server runs) or a secondary release. Clients can be Data General machines or foreign machines.

The checklist will give you some idea of the nature of the task ahead of you. Make a clear plan based on your changing future needs.

Checklist: Installing the DG/UX System

Phase One: Planning the Installation

- ☐ Step 1 Planning Resources and Using DG/UX Conventions
- ☐ Step 2 Planning Disk Usage for the Release
- ☐ Step 3 Listing the Devices on Your System
- ☐ Step 4 Assembling Network Information for Your System
- ☐ Step 5 Understanding the DG/UX Directory Tree

Phase Two: Loading the Primary Release

- ☐ Step 6 Booting diskman from Tape
- ☐ Step 7 Initializing Physical Disks with diskman
- ☐ Step 8 Creating System Logical Disks and File Systems
- ☐ Step 9 Loading DG/UX Software onto System Logical Disks
- ☐ Step 10 Updating System Software

Phase Three: Customizing the Primary Release

- ☐ Step 11 Booting the Starter Kernel
- ☐ Step 12 Creating Other Logical Disks and File Systems
- ☐ Step 13 Loading Software Packages with sysadm
- ☐ Step 14 Setting Up Software Packages with sysadm
- ☐ Step 15 Building a Custom Kernel
- ☐ Step 16 Setting Default Boot Characteristics
- ☐ Step 17 Starting System Administration

Phase Four: Adding OS Releases and Clients

- ☐ Step 18 Adding Secondary Releases
- ☐ Step 19 Building Kernels for Diskless Clients
- ☐ Step 20 Setting OS Client Defaults
- ☐ Step 21 Adding OS Clients
- ☐ Step 22 Booting and Setting Up an OS Client

Device Specifications: General Background

The information in this section is intended to introduce you to the methods for specifying devices for AViiON hardware running the DG/UX system. For the most complete details on setting up and selecting what device specifications to use, refer to your hardware manual for starting and setting up your machine, and to *Using the AViiON™ System Control Monitor (SCM)*. Also, see Appendix A in this manual, "Device Names and Codes."

Devices such as disk controllers and tape drives communicate with the operating system via device driver programs. To access a device, the operating system must be able to identify the device uniquely. The system does this with device codes. Device codes are derived from the hardware identifier that is used during an interrupt; therefore, device codes reflect the interrupt structure of the host system.

Device codes for devices on the VME bus can be jumpered as you desire. Instructions on jumpering are included as part of the documentation that is shipped with a particular board or device.

ESDI controllers support up to three ESDI disks, (units 0, 1, and 2). Data General has set two default device numbers (that represent base addresses) for ESDI controllers. This means that you can have up to two ESDI controllers with a total of six disks in the default situation. If you need to have more than six disks, you will have to select unused base addresses for the additional controllers needed. See Appendix A for a listing of default and base addresses for AViiON servers and workstations.

Syntax

A device specification consists of a mnemonic that identifies the type of device and some optional numerical parameters:

device_mnemonic [@device_code] ([parameters])

where the fields are as follows:

device_mnemonic	A two-to-four letter mnemonic used to identify the device.
device_code	If your driver is for a controller or adapter, you enter its device code preceded by an "at" sign (@); for example, **@18** (hexadecimal).
parameters	The parameters for the device specification depend on the type of device and whether the device is a controller, adapter, or a normal device unit.

The following are valid device specifications:

cied(0,1)	Drive 1 on Ciprico ESDI disk controller 0.

cied()	Ciprico ESDI disk controller with all parameters assuming their default values.
cied@72(fffff500,0)	Drive 0 on the Ciprico ESDI disk controller at the non-standard base address 0xfffff500, with the non-standard device code 72. This would be an example of adding another disk device after you had used up the default device codes.
sd(cisc(1),2)	The SCSI disk at SCSI id 2 reachable through the first SCSI adapter 1.
st(insc(),4)	The SCSI tape at SCSI id 4 reachable through the first integrated SCSI adapter.
sd(insc(),*)	All the SCSI disks reachable through the first integrated SCSI adapter.

Simplified Device Specifications

Although it is rare that you will need to specify a device specification in its long form (such as **sd(cisc(1),2)**), you do need to specify the long form when using the **diskman** program. Whenever you boot your system, a script called **chk.devlink** runs. This script reads the device files in **/dev** and sequentially maps each device to a number starting with 0 through *n* devices of a particular kind (where ***n*+1** is the number of devices of that kind). When the mapping is complete, you can refer to a device by its number instead of the long specification. For instance, **/dev/rmt/0** would represent your first tape drive, and **/dev/pdsk/1** would represent your second disk. Read your **/etc/devlinktab** file after booting to see how your devices are mapped. You can change the mappings in this file, and your changes will take affect the next time you boot.

To specify tape devices over the network, see the man pages **pmttapetab**(4), **rmt**(1), and **pmtd**(1M).

Phase One: Planning the Installation

You need to complete all the steps in Phase One whether you are installing a preloaded system or one that is not preloaded.

Planning the system involves:

- Planning Resources and Using DG/UX Conventions
- Planning Disk Usage for the Release
- Listing the Devices on Your System
- Assembling Network Information for Your System
- Understanding the DG/UX Directory Tree

Step 1: Planning Resources and Using DG/UX Conventions

Your interactions with the DG/UX operating system will go much more smoothly if you observe the following conventions:

hosts

A host name is a name that you assign to a system. Although host names may be anything you choose, it is wise to select names that relate to the use or location of their respective systems. For example, you could name machines according to owner, group, project, or department. A sales organization may name its system **sales**, for example, while a graphics application development group may name its system **grafdev**. Appropriate mnemonic names are particularly helpful in networked environments where systems may share file systems. An appropriate file system name tells you something of the nature of its contents as well as its home system. Do not use the capitalized names **MY_HOST** or **PRIMARY**; these names are reserved by **sysadm** for managing releases.

releases

A release is a complete operating system package. You will have at least one release, the 4.30 DG/UX system, which is the primary release. If you intend to support secondary releases for OS clients that will not run the 4.30 DG/UX system, you need to think up names for the releases. A release name must conform to DG/UX file name restrictions. 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 4.20 DG/UX system release **88k_dgux_420** because it is based on the Motorola 88000 chip architecture and the 4.20 DG/UX system.

logical disks

A logical disk is a fixed portion of reserved disk space. Creating a logical disk for a specific file system ensures that the file system will have that portion of space for its own use, and that no other file system can use its space. Conversely, a logical disk limits the size of the file system that it contains. The **swap** logical disk, which is for system use only, is the exception because it does not contain a file system.

The DG/UX system requires that system software be on logical disks named **root** and **usr**. As stated before, the system also needs a logical disk named **swap**. If you purchased the Client/Server User's Package for AViiON Systems, which includes the X11 and Framemaker packages, you also need to have a **usr_opt_X11** logical disk and a **usr_opt_frame1.3** logical disk.

If you have a factory-preloaded disk, the **root**, **usr**, and **swap** logical disks have already been created. If you have the X11 and Framemaker packages, the **usr_opt_X11** logical disk has already been created, but the **usr_opt_frame1.3** logical disk has not.

If you do not have a preloaded disk, you need to create the **root**, **usr**, and **swap** logical disks yourself. If you have X11 and Framemaker, you need to create **usr_opt_X11** and **usr_opt_frame1.3**.

The DG/UX system design philosophy dictates that you should not put any of your own software or files on the **root** or **usr** logical disks; many software packages have the same restriction for their own logical disks (like **usr_opt_X11**). Putting your own files in these file systems can cause problems when you update software. For your own programs and files, you should create separate logical disks. It is good practice to give these customized logical disks names relating to their contents or origin. Specifically, it is a good idea to name logical disks for the file systems that they will contain. Step 2 discusses logical disks and disk planning in more detail.

packages

A package is a set of files and/or programs that make up a software application or other utility, such as the Framemaker package. It is good practice to put each package on a logical disk tailored to fit the package's size requirements. For the Framemaker package, for instance, you may create a logical disk called **usr_opt_frame1.3**.

local software

For your site-specific shell scripts, programs, and so on, we suggest that you use the 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.

Step 2: Planning Disk Usage for the Release

If you have a preloaded disk, you do not need to create **root**, **usr**, or **swap** logical disks. If your DG/UX system package includes the X11 package, your preloaded disk will also have a **usr_opt_X11** logical disk. On non-preloaded systems, you need to create the **root**, **usr**, **swap**, and (if necessary) **usr_opt_X11** logical disks yourself. You should not add your own software or files to these logical disks—they are only for system use. This chapter tells how to create these logical disks in a later step.

Besides system logical disks, all systems need logical disks for local work. You need logical disks for your own programs, databases, personal directories, and so on. This section discusses these disks in more detail a little later.

You plan your disks based on your specific needs. You need to ask yourself questions like these:

- Will the system be an OS server? If so, for how many clients? Will it support multiple releases?
- How many users will have home directories on the system? How much space will each user need?
- Will the system be a database server? How big will the database file(s) be? Will the database design involve splitting databases over multiple physical disks?
- How much total disk space do you have? Can you acquire more if necessary?

By dividing your disk storage load into specialized logical disks, you have greater control over disk usage. You can tell easily where your needs and excesses are. Because logical disks are limited to the sizes you define for them, they allow you to contain accidental or occasional fluctuations in space consumption.

Logical disks also allow a certain amount of security and access control. For each file system on your system, you may specify which systems on your network may access the file system, which may write to (or just read) the file system, and which remote superusers may override permissions on the file system.

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

System Logical Disks

On preloaded systems, the system logical disks are created for you at the factory; therefore, you do not need to create the **root**, **usr**, **swap**, or (if you have the X11 package) **usr_opt_X11** logical disks.

Table 2-1 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 and mount points.

Table 2-1 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	**/**
swap	50,000*	–
usr	160,000	**/usr**
usr_opt_X11	105,000**	**/usr/opt/X11**

* 32,768 on preloaded 179MB disks.

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

The default **usr** logical disk size will accomodate the DG/UX operating system, TCP/IP (10,000 blocks), and ONC/NFS (20,000 blocks). If you do not intend to load TCP/IP and ONC/NFS, you can reduce the size of your **usr** logical disk.

Swap Space

The swap logical disk exists to ensure that memory is not in contention among processes waiting or ready to run. To keep idle (waiting) processes from taking up memory space, the operating system writes them out to the swap space until they are ready to run. Every system needs at least one swap logical disk.

If you have a preloaded disk, you do not need to create a **swap** logical disk. If you do not have a preloaded disk, you need to create the **swap** logical disk yourself. In both cases, the default size, 50,000 blocks (except on 179MB preloaded disks, where **swap** is 32,768 blocks), is probably sufficient. Read this section so you may evaluate your swap needs and determine if these default values are sufficient.

There is no reliable formula for determining how much swap space a system needs. Too little swap space can cause processes to fail and errors like "out of paging area" or "out of swap space" to appear on your system console. Too much swap area, on the other hand, can constitute a waste of valuable disk space. A general rule that works for most systems is that swap area should be 1.5 times the size of the computer's physical memory. The factors you need to consider are:

- The applications you intend to run. Applications that allocate lots of memory (whether they use it or not) will push your swap needs upward. Simulations and graphics applications, for example, tend to require a lot of memory.

- Amount of physical memory. On systems that have a generous supply of physical memory, swap space is less critical. Take, for example, a system with 128MB of memory functioning primarily as a file server (that is, not much CPU load)--such a system would probably not need much swap space at all. On the other hand, a system with very little memory will have to compensate with extra swap space.

- Number of users and processes. The more users and processes you have, the more swap space you need.

When creating the **swap** logical disk, you may specify swap space as small as 25,000 blocks. If you specify less, **diskman** returns you to the installation menu. If users encounter errors resulting in messages like "out of paging area," you need to create more swap area. You may create multiple swap logical disks to supplement existing swap logical space. Chapter 8, "File System Management," discusses adding more swap space.

Application Packages

You need to create logical disks for application package software shipped on the DG/UX system tape (like the Framemaker package) as well as any other packages you intend to install.

You may create one logical disk to contain all packages, or you may create a separate logical disk for each package. We recommend that you create a logical disk for each package. By tailoring the size of a logical disk to fit its software package, you save disk space.

The Framemaker package requires 30,000 blocks. If you have other packages to install, see the documentation that came with the package.

Customized Logical Disks

On both preloaded and non-preloaded systems, you need to create a number of logical disks whose sizes depend entirely on your plans for your system.

Home directories, for example, may require a lot of space or very little. Points to ponder: will your users produce (and accumulate) programs? academic papers? database reports? data files? In environments where most activity occurs in work directories rather than home directories, users may not need much home directory space. Remember to account for any OS clients' home directories as well as directories for local users.

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 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 directories (that is, **/var/tmp**) can fluctuate in size. You may find it convenient to regulate your temporary file directory size by making it its own logical disk and file system.

Client Logical Disks

If your system will be an OS server, you need to create logical disks for: the **/srv** directory, client swap space, client root space, client dump space, windowing/graphics software, and temporary file space. If you intend to support secondary releases, you need to create logical disks for them as well.

The /srv directory

You should make a logical disk for the **/srv** file system. The logical disk does not need to be large: 5000 blocks is sufficient.

Client Swap Space

A previous section, "System Logical Disks," defines swap space and discusses size recommendations. Whatever you decide for the swap needs of your OS clients, you should create a logical disk for their 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. You should name the client swap file system **/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 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

    50,000        *        5       * 1.17 =     292,500
```

Client Root Space

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. For five OS clients, this adds up to 200,000 blocks. Unlike the client swap area file system, you do not need to add file system overhead (the 40,000 default root size already includes file system overhead).

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 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 client root directories over different logical disks. Sharing kernels can, however, result in weakened security. See Step 19 for more information on kernel sharing.

Client Dump Space

You need to allocate space for 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 systems 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). Step 15, "Building a Custom Kernel," tells how to 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 is what determines where the client's system dump goes. Typically, the file is **/srv/dump/**client_name, where *client_name* is the host name of the client. At this point in installation, you need only 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.

The maximum amount of space you would need for **/srv/dump** is equal to the total size of physical memory on all of your client systems. In practice, however, you need less space than this for two reasons: a) clients will not need to make systems dumps all at the same time; and b) 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 your clients have no more than 16MB of physical memory each, 33,000 blocks is sufficient to hold one system dump. If you intend to make the dump directory its own logical disk and file system, remember to add in 17% file system overhead (example: 33,000 blocks + 17% = 38,600 blocks).

Windowing/Graphics Software

Whether or not local users on the server need windowing or graphics software, you should install such software it if you plan to support OS clients. As stated before, the X Window System™ (X11) takes up 105,000 blocks. You do not need to duplicate this package for every OS client that you support.

Temporary File Space

As stated previously, temporary file space may be something you want to govern by creating a separate logical disk. If you create a logical disk for temporary file space, mount it on the **/var/tmp** directory, not on **/tmp**.

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 install 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 3: Listing the Devices on Your System

A lot of the steps in the installation process require that you know the device specifications of your various devices. In fact, throughout the life of your system, you need to know device specifications in order to perform tasks like adding and removing logical disks, building kernels, and making tape archives. Make a list of your system's device specifications and keep it handy. You may need to review the section at the beginning of the chapter, "Device Specifications: General Background" and Appendix A, "Device Names and Codes." You may also need to review the documentation and notices that came with your hardware.

Step 4: Assembling Network Information for Your System

If your system will not be part of a network, you may skip this step.

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 more information, see Chapter 13, "Network Management," and the manuals *Setting Up and Managing TCP/IP on the DG/UX™ System* and *Managing NFS and Its Facilities on the DG/UX™ System*.

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**.

host name — This name could be whatever you intend to call your system within your network. Step 1, "Planning Resources and Using DG/UX Conventions" 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 a workstation 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 an extra 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

You do not need to prepare before setting up the ONC/NFS package. 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.

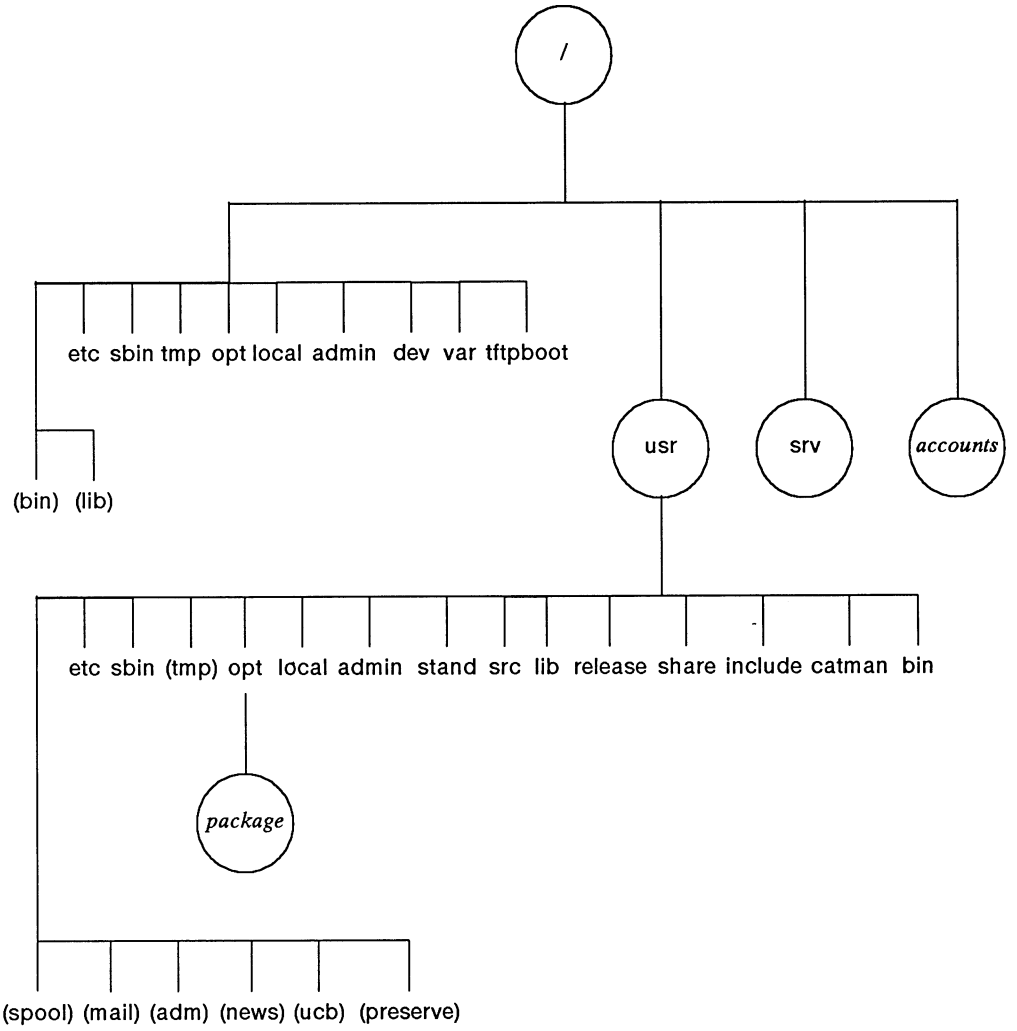
Compiling OS Client Information

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. The empty table below is provided for your host/client information.

Host	IP Address	Ethernet Address	Release

Step 5: Understanding the DG/UX Directory Tree

This section shows an overview of the DG/UX directory tree. In addition, this section shows the **/**, **/usr**, and **/srv** file systems.



Circles are logical disks (file systems) mounted on directories. In this directory structure:

accounts is a logical disk (which you would have to create yourself) intended to contain users' personal directories.
package is the mount point for **usr_opt_X11**

Names in parentheses are symbolic links to directories.

Figure 2-1 *An Example DG/UX Directory Tree*

The / Directory Tree

Here is the **/** file system. The `foo --> bar` notation indicates that foo is a symbolic link to bar. For example, commands traditionally found in **/bin** have been moved to **/usr/bin**. Note that **bin**, **dev**, **etc**, **lib**, **tmp**, and **sbin** must remain on the **root** logical disk. Do not move them elsewhere or use them as mount points.

```
/
|
|    |-- (bin)  --> /usr/bin
|    |-- (lib)  --> /usr/lib
|----|
|    |-- etc
|    |     |-- sysadm
|    |     |-- uucp
|    |
|    |-- dev
|    |-- sbin
|    |-- admin
|    |-- local
|    |-- tmp
|    |-- srv
|    |-- tftpboot
|    |
|    |-- var
|    |     |-- adm
|          |-- cron
|          |-- mail
|          |-- news
|          |-- preserve
|          |
|          |-- spool
|          |      |-- lp
|          |      |-- uucp
|          |      |-- uucppublic
|          |
|          |-- tmp
|          |-- uucp
|          |-- yp
```

The /srv Directory Tree

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.

```
/srv
 |
 |--- release
 |       |
 |       |---PRIMARY
 |       |        |
 |       |        |---(usr)   --> /usr
 |       |        |--- root
 |       |        |     |
 |       |              |--- (MY_HOST)   --> / (root)
 |       |              |
 |       |              |--- dg1
 |       |              |--- _Kernels
 |       |
 |       |--- 68k_sunos_4
 |                |--- usr
 |--- share       |--- root
 |                       |--- sun1
 |--- swap               |--- _Kernels
 |      |
 |      |--- dg1
 |      |--- sun1
 |
 |--- admin
        |
        |--- clients
        |--- defaults      (sysadm data files)
        |--- releases
```

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.

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.

```
/usr
 |
 |-- admin
 |-- bin
 |-- sbin
 |-- catman
 |-- etc
 |     |-- init.d
 |     |-- master.d
 |
 |-- include
 |-- lib
 |-- local *
 |-- (adm)    --> /var/adm
 |-- (mail)    --> /var/mail *
 |-- (news)    --> /var/news *
 |-- (spool)    --> /var/spool
 |-- (preserve) --> /var/preserve
 |-- (tmp)    --> /var/tmp *
 |-- (ucb)    --> /usr/bin
 |-- release
 |-- root.proto
 |-- src
 |     |-- lib
 |     |-- uts
 |           |--aviion
 |--- stand
 |--- share *
```

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**. Depending on your usage, you may want to consider making logical disks for the directories marked with asterisks (*) and mounting them under **/var**.

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.

```
/srv
  |
  |--- release
  |       |--- PRIMARY (88k_dgux_430)
  |       |        |
  |       |        |--- usr   --> /usr
  |       |        |--- root
  |       |             |--- MY_HOST   --> / (root)
  |       |             |--- dg1
  |       |
  |       |--- 68k_sunos_4
  |              |--- usr
  |              |--- root
  |-- share      |--- sun1
  |-- swap
  |      |
  |      |--- dg1
  |      |--- sun1
  |
  |--- admin            (sysadm data files)
```

A foreign system **/export** structure:

```
           |--- root
           |      |--- server  (primary release: symbolic link to /)
/export----|      |--- dg1
           |      |--- sun1
           |
           |--- exec
           |      |--- 88k_dgux_430
           |      |--- 68k_sunos_4
           |
           |--- swap
           |      |--- server
           |      |--- dg1
           |      |--- sun1
           |
           |--- share
```

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.

Phase Two: Loading the Primary Release

If you have a preloaded disk, skip this phase and proceed to Phase Three, "Customizing the Primary Release." If you are installing the DG/UX system on a system that has a previous release of the DG/UX system already installed, 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.

First, use stand-alone **diskman** to load the DG/UX system into the server's **root** and **usr** logical disks (do the same thing for stand-alone machines). The **diskman** program is a menu-driven disk management utility with a built-in installation procedure.

The steps in this phase are

- Booting **diskman** from Tape
- Initializing Physical Disks with **diskman**
- Creating System Logical Disks and File Systems
- Loading DG/UX Software onto System Logical Disks
- Updating System Software (if necessary)
- Booting the Starter Kernel
- Setting Up DG/UX: Initial Configuration

Step 6: Booting diskman from Tape

Before working with **diskman**, you may want to review the DG/UX common format for naming devices. See the section "Device Specifications: General Background" near the beginning of this chapter. Specifically, you need to know the specification for your tape device.

At this point, your hardware should be installed and powered on. Your console should show the SCM prompt. Mount your installation tape. Boot the tape using this command line (for most servers):

```
SCM> b st(cisc(),4) ↵
```

or this command line (for workstations and some servers):

```
SCM> b st(insc(),4) ↵
```

Once **diskman** is loaded and executing, the system comes up in a device menu for **diskman** and queries you about device names. The entry you should specify depends on the kind of terminal 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()**, like this:

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

After you complete these steps, **diskman** begins execution. Once you are in **diskman**, you will need to know the proper names for the devices on your server or workstation (particularly for the tape device and the disk devices).

Table 2-2 shows you how to use the **diskman** menus.

Table 2-2 Using Diskman Menus

User Input	Description
^	Return to previous menu.
?	Print HELP message, then redisplay menu.
number	Choose menu item by entering a number.
number?	Give information on the item number specified.
q	Exit from **diskman**. Enter this command from anywhere.
New Line	Same as entering the default response.

When the system has loaded **diskman**, 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
```

The fourth option, Initial Installation Menu, is the default. Press the New Line key. This option will lead you through all the steps necessary to get your system running in single-user mode.

You will see the **diskman** menu for initial installation:

```
            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** program will automatically take you through the six steps, beginning with number 1, initializing physical disks.

Step 7: Initializing Physical Disks with diskman

In this step, **diskman** will format the physical disks, perform read/write surface analysis to find and remap bad blocks, and install bootstraps.

Surface analysis is very time consuming at this point because you can do only one physical disk at a time.

After you choose number 7, the system proceeds to lead you through a dialog to initialize your physical disks. The system queries you about a number of things:

- First, the system queries you for the device specification for the first disk. Refer to the list of devices you compiled in Step 3.
- Reinstalling the disk label. You do not need to perform this step, so you may respond with **no**.
- Formatting the disk. Answer **yes**.
- Creating system areas on the disk. Answer **yes** (unless you are formatting a diskette; see Appendix E). This step includes a prompt for the bad block remap area size. Take the default.
- Installing bootstraps. Answer **yes** for disks that will contain **root**, usr, or any other logical disk that will contain an image you may want to boot.
- Performing surface analysis. You do not need to do this unless you suspect that your disk surface is flawed. Surface analysis step can take a long time.

If Your Disk Has a Built-in Bad Block Table

Some types of disks come from the manufacturer with a bad block table built into the hardware itself. If your physical disk comes with a built-in bad block table, you will see the following message:

```
Running surface analysis on this model of disk is not
required because the disk controller maintains a hardware
bad block table.  See the manual for more details.

You have the option of running either all test patterns
or a single test pattern.

Do you want to run all the test patterns? [y] ↵
```

Even if your disk has a built-in bad block table, you have the option of running surface analysis on it anyway to be sure that there are no problems. If you enter **n** to the test pattern prompt, **diskman** runs only one test pattern. The process takes about 20 minutes per 100MB, depending on your physical disk model and CPU. An average is about an hour per physical disk. During testing, **diskman** produces a status message on your console every five minutes. After all patterns have run, the bad block table is written to the disk. You'll see the following message:

```
Surface analysis finished.
n bad blocks were found and remapped.
Do you want to format another Physical Disk?  [n] y↵
```

If you need to format another physical disk, type **y**. The program for intializing physical disks will repeat from the beginning. After initializing your disks, proceed to the next step.

Step 8: Creating System Logical Disks and File Systems

Next you need to create the system logical disks, **root**, **usr**, and **swap**.

Creating the Root Logical Disk and File System

Here, **diskman** creates a **root** logical disk and makes the **/** (root) 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] ↵
```

Accept the default name (**root**). This is the node name that the operating system will use in **/dev/dsk** and **/dev/rdsk**. Again, our example dialog uses **cied(0,0)** as the disk specification.

```
Enter the Physical Disk specification in DG/UX common
format: [cied(0,0)]  ↵
The physical disk must be registered for this operation.
Do you want to register it? [y] ↵
Physical disk cied(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:  [default]  ↵
```

The default starting block, given as a hexadecimal location, begins at the first available space on the disk. Accept this default.

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

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 Logical Disk `root' has been created.

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

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

Creating the Swap Logical Disk

A swap area is a part of the disk where the operating system stores process information when memory is in contention.

```
      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: [cied(0,0)]  ↵
```

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 **diskman** 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. If you are installing your system on a 179MB disk, swap space should be 32,768 blocks. Step 2 discusses swap space in more detail.

```
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 dialog for creating the **/usr** file system continues much as in the previous dialogs. As before, you need to know the specification for the disk where you intend to install **/usr**. For the size of the logical disk, 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: cied(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: [default]  ↵
Enter the size in blocks of Logical Disk
Piece 1: [160000] ↵
```

Unlike the **swap** and **root** logical disks, the **usr** logical disk may consist of multiple pieces. The **diskman** program loops back for information on up to seven more logical disk pieces. If you decide not to 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.  ♪
```

Creating Custom Logical Disks and File Systems

Following the plans you made in Step 2, you may decide to create more logical disks and file systems at this time. You do not need to create them at this time, however, because Phase Three offers you the opportunity to do so.

Step 9: Loading DG/UX Software onto System Logical Disks

Now you need to load the **/** and **/usr** file systems.

Loading the / File System

In Step 5 of the initial installation, **diskman** loads the **/** file system's files from the release tape to the location you created for it. These files include a starter system that contains a minimum of information on disk and tape devices. The example tape specification in the following dialog represents the kind of tape that might be installed on a server.

```
        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(cisc(),4) ♪
  Ready to load the Root File System.

  Mount the first release tape on the tape drive st(cisc(),4)
  Press New Line when ready to continue... ♪
```

Make sure that your tape is mounted and online, then press New Line. Depending on how you answered the question about seeing the file names during the load, you may see them listed now. When the load is complete, you see:

```
  The Root File System has been loaded.

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

Loading the /usr File System

In this last **diskman** step, you'll load the **/usr** file system from the release tape to the location you created for it.

```
        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(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... ♪
```

When the system has loaded all tapes, you see these messages:

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

Your starter system has been installed.

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

Pressing New Line at this point returns you to the installation menu. If you have more update software to add, remain in **diskman** and continue to Step 10, "Updating System Software." If you do not have any more update software, exit **diskman**, skip Step 10, and proceed to Phase Three.

Step 10: Updating System Software

In addition to your release tape set, you may have also received one or more tapes containing incremental updates to the DG/UX system. If so, load those tapes now. If not, move on to the next phase.

At this point, you should have your update tape mounted (make sure the tape is online and at the beginning of the tape, BOT), and you should still be in stand-alone **diskman**.

If you are updating the DG/UX system software, you first need to reinstall the bootstrap on the physical disk(s) from which you boot. If you are installing any other software update, you may skip this paragraph. The disks on which you need to reinstall the bootstrap are the ones containing the **root** logical disk, the **usr** logical disk, and any other physical disk that contains a boot image. Return to the Diskman Main Menu and select number 1, "Physical Disk Management Menu." From this menu, select number 5, "Format a Physical Disk." From the Physical Disk Formatting Menu, select number 4, "Reinstall Bootstraps on a Physical Disk."

To install a software update, select number 5, "Update Installation Menu." Your system will display the following:

```
            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]
```

Mount your update tape, then select number 3. The ensuing dialog requires that you know your tape device specification. If the update is on more than one tape, you will be prompted for each tape. First, **diskman** updates the root file system, then the **usr** file system.

When the update is complete, you may exit **diskman** and proceed to Phase Three.

Phase Three: 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 (whether you loaded it yourself or whether you purchased a preloaded system). You are now ready to complete the setup and installation of your system. The steps in this phase are:

- Booting the Starter Kernel.
- 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 11: 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.

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. From this point onward, installation continues the same way for preloaded and non-preloaded systems alike.

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 2-3 shows the possible starter devices for AViiON systems.

Table 2-3 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 12: 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 eight 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_accounts** logical disk and file system on the **/sales/accounts** directory, the **addfsys** dialog 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 13: Loading Software Packages with sysadm

The **sysadm loadpackage** function allows you to load all software packages at once that are on a single tape. 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.

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** dialog.

```
# 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 14: 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 4, "Assembling Network Information for Your System."

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

Setting Up ONC/NFS

You do not need to prepare before setting up the ONC/NFS package with **sysdam setuppackage**. 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 15, and you boot it in Step 16. If you are installing a YP master or server, you finish YP setup at the end of Step 16.

Step 15: 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 2-4 shows the commands you can use in editing mode. These commands are case sensitive.

Table 2-4 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 Geneal 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
#  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
```

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 16: Setting Default Boot Characteristics

Now that your kernel is configured, there are two boot characteristics that you can set: a default boot path in the SCM and a default initial run level. If you intend to use your system as a YP master or server, there are also some things you need to do to complete YP setup.

The SCM Boot Path

You can set a default path for booting the DG/UX operating system. Take your system down, and use the SCM's **f** command:

```
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 dialog. The following example dialog 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]
```

You now have the option of booting the system. With the system boot path set correctly, you simply type the single letter **b** at the SCM prompt to boot the system.

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 logical disk called **alt_root** from which you want to boot, use a command line like this:

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

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 inital 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
```

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 17: 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 64, then use this procedure as described below. If you have more than 64, 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. If, for example, you have 84 ttys, you need to adjust the NPROC variable upward by 20 from its current default value. This will prevent 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 is for 24 ttys. 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 dialog 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**".

Phase Four: Adding OS Releases and Clients

You must add a release to the system before you can attach a client to that release. Phase Three 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 18: 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 dialog 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 19: 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 you 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 20: 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.
```

Step 21: 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 ♪
```

Server and Client /etc/fstab Files

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 wants all clients to mount upon booting. To set this up, the server administrator would edit the **/srv/release/PRIMARY/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 called **sales** and an OS client called **dg1**.

Server: **sales** --

```
/dev/dsk/root                                /        dgux
/dev/dsk/usr                                /usr      dgux
/dev/dsk/swap                               swap      swap
/dev/dsk/srv                                /srv      swap
/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       swap
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.

Setting Up Packages on the Client

On the OS client system, we must set up the TCP/IP, ONC/NFS, and YP packages. You may want to refer to Step 4, "Assembling Network Information for Your System," where it discusses the information you need before setting up TCP/IP, ONC/NFS, and YP.

On the OS client machine, type the following:

```
# sysadm setuppackage ↵
```

Answer the questions as before. Normally, you'll want to set the client host as a YP client. The YP setup script defaults to this state.

Step 22: 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

Administrators of client machines can now set a default boot path with the SCM **format** command as we did for the server in Step 16. Clients may also want to do the following:

- Set their **inittab** default boot run level; it is shipped at level s, single user, but you may prefer to have your system come up in level 3, multiuser. We did this earlier for the server at the end of Step 15.

- As prompted when the system comes up, use **sysadm setuppackage** on the client machine. Note that by default, OS clients use the same **tcpip.params** setting as the OS server.

- 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.

Examples

This section of the chapter (which continues to the end of the chapter) is completely new with release 4.30 of the DG/UX system. Although the rest of the section is not marked with revision bars in the margin, it is completely new.

This section reproduces example installation scenarios for specific system configurations. During installation, you may refer to these examples to see how installation proceeds for specific configurations, but do not refer to the examples as guides that show you what to do. 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 12MB of memory.
- One 322MB SCSI disk.
- One SCSI QIC-150 tape drive with SCSI adapter.
- One integrated Ethernet controller.
- One parallel printer.

Phase One: Planning the Installation

Step 1: Planning Resources and Using DG/UX Conventions

The host name for our AV310C is **aviion1**. This host name is unique to the network in which we are adding our machine.

We will load the Client/Server package (P001A) for release 4.30 of the DG/UX system and the Frame 1.3 package. Using Chapter 2 and the product release notices, we determine that we will need five logical disks including swap for loading these packages:

Table 2-5 Logical Disks and Software Packages

Logical Disk Name	Packages
root and **usr**	DG/UX system, Gnu C, DTK, DG/UX man pages
usr_opt_X11	X Windows, Looking Glass, OSF/Motif, man pages
usr_opt_frame1.3	Frame 1.3
swap	N/A

We will create another logical disk named **udd_aviion1** as the working directory for the user(s) of **aviion1**.

Step 2: Planning Disk Usage for the Release

Table 2-6 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 3: Listing the Devices on Your System

Table 2-7 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

Step 4: Assembling Information for Your System

Table 2-8 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**

Step 5: Understanding the DG/UX Directory Tree

No action is required for this step.

Phase Two: Loading the Primary Release

Step 6: Booting diskman from Tape

```
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 7: 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 8: 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 9: Loading DG/UX Software 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...
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 10: Updating System Software

This step is not necessary for our example system.

Phase Three: Customizing the Primary Release

Step 11: 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
```

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 12: 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 13: 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 14: 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 15: 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 16: 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 deault initial run level by editing the **/etc/inittab** file and changing this line:

```
def:s:initdefault:
```

To this line:

```
def:3:initdefault:
```

Step 17: 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 Four: 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 16MB of memory.
- One 662MB 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.

Phase One: Planning the Installation

Step 1: Planning Resources and Using DG/UX Conventions

The hostname for our system is **sales**. It will have a primary release for release 4.30 of the DG/UX system, and it will have a foreign release called **68k_sunos_4** for release 4.0 of the SunOS system. The system will have these logical disks:

Table 2-9 Logical Disks and Software Packages

Logical Disk Name	Packages
swap	N/A
root	DG/UX system, Gnu C, DTK
usr	DG/UX system, Gnu C, DTK, man pages
usr_opt_X11	X11, OSF/Motif, Looking Glass, man pages
sales_users	Home directories
sales_bin	Local executables
sales_data	Local databases
srv	N/A
srv_swap	N/A
srv_dgux430	DG/UX system for OS clients
srv_sunos4	Foreign system for OS clients
tmp	Temporary space

Step 2: Planning Disk Usage for the Release

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
		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 3: Listing the Devices on Your System

Table 2-11 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

Step 4: Assembling Information for Your System

Table 2-12 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**

Step 5: Understanding the DG/UX Directory Tree

No action is required for this step.

Phase Two: Loading the Primary Release

Step 6: Booting diskman from Tape

```
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 7: 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 8: 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 9: Loading DG/UX Software 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...
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...
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 10: Updating System Software

This step is not necessary for our example system.

Phase Three: Customizing the Primary Release

Step 11: 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
```

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/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 12: 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 One. 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 One.

Step 13: 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 14: 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:

            X11      nfs      tcpip         yp
            X11.lg

Package Name? [all]  ↵

Processing setup scripts for package X11.
Setup 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

Processing setup scripts for package X11.lg.
Setup 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 ...

Setup package X11.lg in MY_HOST root? [yes]  ↵

           Setting up package: X11.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 15: 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
# PERCENINFS 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()"

### PERCENINFS                  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 16: 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 deault initial run level by editing the /etc/inittab file and changing this line:

```
def:s:initdefault:
```

To this line:

```
def:3:initdefault:
```

Step 17: 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 Four: Adding OS Releases and Clients

Step 18: 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 19: 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 20: 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 21: 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 22: 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 32MB of memory.
- Two 1GB SMD disks.
- One SCSI QIC-150 150MB tape drive with SCSI adapter.
- One SCSI 2GB Maytag tape drive.
- One Systech asynchronous controller supporting 128 terminals.
- One serial printer.

Phase One: Planning the Installation

Step 1: Planning Resources and Using DG/UX Conventions

The host name for our AV6200 is **aviion1**. This host name is unique to the network in which we are adding our machine.

We will load the Operating Systems package (Q001A) for release 4.30 of the DG/UX system. We will also plan on loading one additional language compiler and a database system. Using Chapter 2 and the product release notices, we determine that we will need five logical disks including swap. Table 2-13 shows the logical disks and their software packages.

Table 2-13 Logical Disks and Software Packages

Logical Disk Name	Packages
root and **usr**	DG/UX, Gnu C, DTK, DG/UX man pages
usr_opt_lang1	Language compiler
usr_opt_db1	Database system
swap	N/A

We will create another logical disk named **udd_aviion1** as the working directory for the user(s) of **aviion1**. Because we expect a fair number of users to be compiling and editing files at one time, we will create a separate logical disk for **/var/tmp**.

Step 2: Planning Disk Usage for the Release

Table 2-14 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 3: Listing the Devices on Your System

Table 2-15 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

Step 4: Assembling Network Information for Your System

This step is not necessary for our example system.

Step 5: Understanding the DG/UX Directory Tree

No action is required for this step.

Phase Two: Loading the Primary Release

Step 6: Booting diskman from Tape

```
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 7: 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 8: 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 9: Loading DG/UX Software 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...
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 10: Updating System Software

This step is not necessary for our example system.

Phase Three: Customizing the Primary Release

Step 11: 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
```

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 protytype 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 12: 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 One. 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 One.

Step 13: 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 14: Setting Up Software Packages with sysadm

None of the DG/UX packages we loaded require setup via **sysadm setuppackage**.

Step 15: 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
# PERCENINFS 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 16: 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 17: 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 Four: Adding OS Releases and Clients

This phase is not necessary for our example.

End of Example

End of Chapter

Chapter 3
Operating the DG/UX System

This chapter shows you how to do system operations such as starting up, shutting down, recovering from trouble, collecting error messages, and dumping the system for analysis by Data General. Also, we explain how the **init**(1M) program interacts with **rc** scripts to set run levels and thus determine the process environment of your system.

The major sections of this chapter are as follows:

- Operational Terms
- DG/UX System Run Levels
- Operational Procedures
- How Run Levels Are Set
- Expert Run Level Information

Operational Terms

Read the following definitions before beginning the procedures in this chapter:

init The **init**(1M) program creates all other processes based on instructions in the file **/etc/inittab**. **Init** is invoked in two ways: inside the DG/UX system as the last step in the boot procedure, and from the command line with a run level as argument. When invoked during the boot procedure, its first function is normally to start a single superuser shell for the system console.

run level Also known as run state or run mode. Run levels are operating modes intended to provide varying degrees of service on the system. These services include network-related capabilities, accounting, **cron** batch job scheduling, lineprinter services, and so on. Typically, run level S (for single user mode) provides no services, and run levels 0 through 3 provide increasing levels of system functionality. As shipped, DG/UX provides full multiuser and network capabilities at run level 3.

You can define the run levels to provide whatever services you choose. The services themselves are controlled by the run command scripts (or **rc** scripts, described below) located in **/usr/sbin/init.d**. The mechanism that ties a given service to a run level is the **init** program and the link files located in the **/etc/rc***n***.d** directories (where *n* is S or one of the

numerals **0** through **6**). For more information on run levels, see **init**(1M), Table 3-1, and the section "Expert Run Level Information" in this chapter.

rc scripts The run command scripts are executed at every boot and every time you change run levels. At boot time or whenever run levels change, the **init** program executes scripts in a directory set up specifically for the intended run level. These scripts are the "run command," or **rc** scripts. They kill or start system services as directed by prefixes that consist of either a **K** (kill) or an S (start) followed by an ID number. Upon entering a run level (via the **init** command), all **rc** scripts designated in that run level are executed. Execution means that services are killed in order from highest ID number to lowest ID number. Next, processes are started in order from lowest ID number to highest ID number. The result is that only certain **rc** scripts, those that are started with the **S** switch, are active in any run level.

getty A program invoked by **init** that sets terminal type, mode, speed, and line discipline. The **getty**(1M) program reads a user's login name and invokes the **login**(1) command. While reading the user's login name, **getty** uses information from the **/etc/gettydefs** directory to adapt the system to the speed and type of terminal being used. It is the second process in the **init-getty-login-shell** sequence that ultimately connects a user with the DG/UX system.

SCM> The System Control Monitor prompt, which is displayed when no operating system is running on your computer. This prompt comes from your computer hardware. From the SCM, you use the **b** command to boot (begin execution) of your DG/UX kernel program, thereby starting the operating system. At the SCM prompt, type **h** for help, or see *Using the AViiON™ System Control Monitor (SCM)* for complete information.

single-user mode (S)
The operating system is running and under the control of a single superuser process from the system console. Only the **/** and **/usr** file systems are mounted.

administrative mode (1)
Also known as run level 1. All local file systems are mounted. All user processes are killed except those associated with the operator's console. This mode is for installing and removing software, checking file systems, backups, and other administrative duties.

multiuser mode (2 or 3)
The normal running mode for the DG/UX system. All system software is running. Run levels 2 and 3 differ from each other in that run level 2 allows TCP/IP transmissions outward only, while run level 3 allows TCP/IP transmissions in both directions.

DG/UX System Run Levels

The following table shows the default run levels for the DG/UX system.

Table 3-1 DG/UX Run Levels

Run Level	Description
0	Currently undefined. Reserved for future use.
S	Single-user is a low-level run mode that is the default level the system enters upon booting. The only process running is **init**. Only the **/** (root) and **/usr** file systems are available.
1	Administrative mode is used to install and remove software utilities, run file system backups and restores, and check file system integrity. All local file systems are mounted. Only processes associated with the system console may run.
2	All local file systems are mounted. Local users can log in at terminals and use local facilities, and some out-bound network services are available. Outside systems cannot contact this system over the network. NFS services are not available in this level.
3	Multiuser/remote file-sharing mode. Same as level 2, but with full TCP/IP and NFS services.
4	User-defined run level. Data General does not supply **inittab** definitions for this state.
5	Currently undefined. Reserved for future use.
6	Currently undefined. Reserved for future use.
a, b, c	Pseudo run levels. These can be specified without changing a run level. Typing **init a**, for instance, invokes those entries in **inittab** that have an **a** in the *level* field. See **init**(1M).

Default Multiuser Conditions

See the section "Run Command Level Scripts per Run Level," later in this chapter, for a complete list of the scripts that run when you go to multiuser states 2 or 3.

When you bring up the DG/UX system in multiuser state, the following things happen:

- Local file systems are mounted in run level 2 (as they are in run level 1).
- Remote file systems are mounted in run level 3.

- The error daemon (**syslogd**(8)), the batch job scheduler (**cron**(1M)), the file system synchronizer daemon (**update**(1M)), the network status monitor (**statd**(8C)), and the network lock daemon (**lockd**(8C) are started.

- The LP system and UUCP are ready to use. The **getty** processes are spawned on all connected terminal lines that specify **respawn** in their **/etc/inittab** entry. Users may log into multiuser systems at this point.

- If used, TCP/IP transmissions work outward in run level 2, and work in both directions in run level 3.

Operational Procedures

The following procedures are covered in this section:

- Starting Up the System
- Shutting Down the System
- Recovering from System Failure
- Repairing Damaged Root File System
- Logging System Errors

Starting Up the System

Purpose	To start the system from the SCM.
Starting Conditions	Processor running the SCM.
Commands	**reset, boot**
Note	If you set a default boot path in the SCM, you can boot your system by just typing **b**. Set the path with the SCM **format** command.

To start up your DG/UX operating system, make sure your processor is running and you have the SCM> prompt at the system console. Below, we'll enter the startup commands for our example system. Your command line will differ if you're booting from a workstation. To begin the start up process, type

```
SCM> reset ↵
SCM> b cied()root:/dgux ↵

Booting cied(0,0)root:/dgux loading...
        .
        .
INIT: New run level: S

INIT: SINGLE USER MODE
```

For future operations, note that you can use the **boot** command to load any executable file you choose. For the above example, we loaded our standard operating system. If, for instance, you had a diagnostics program, **/usr/stand/diags**, that you wanted to run, you could execute it with a command line like this:

```
SCM> boot cied()usr:/stand/diags ↵
```

Changing Run Levels

You must be superuser to change run levels. You switch run levels *upward* with the **init**(1M) command. To switch run levels *downward*, change to the root directory and use the **shutdown**(1M) command. Note that you can shut down only to run level S or 1; use **init** to go back up to the desired run level as follows. Type the **init** command with a run level argument:

init 1 Takes you to administrative mode.

init 2 Takes you to multiuser mode.

init 3 Takes you to multiuser mode with network services.

Rebooting the System

To reboot a system that is already running, first shut the system down to run level **s**, single-user mode. The following section tells how to shut down a system.

When the system is in single user mode, you can halt the CPU with the **halt**(1M) command:

```
# halt ↵
```

If you have set the SCM default boot path (use the SCM **f** command), you may boot like this at the SCM prompt:

```
SCM> b ↵
```

If you have not set the SCM default boot path, you will need to specify your complete boot path on the **b** command line. Chapter 2 discusses booting off of the various devices.

Booting brings the system up to the default run level set in **/etc/inittab**. If you want to boot to a run level other than the default, use a complete **b** command (one that includes the device and boot file specification) and append, as an option, the run level to which you want to boot. For example, to boot from an ESDI disk to run level 2, use a command line like this:

```
SCM> b cied(0,0)root:/dgux -2 ↵
```

After a power outage, the autoboot mechanism automatically brings your system back up to the default run level set in **/etc/inittab**. Initially, the default run level is **s** (single-user mode). To change it, edit your **/etc/inittab** file and change this line:

```
def:s:initdefault:
```

so that the second field contains the desired default run level. For example, the following line makes run level 3 the default:

```
def:3:initdefault:
```

Shutting Down the System

Purpose	To shut down to run level 1 or S. To shut down to power off.
Starting Conditions	Any run level above S, single-user mode.
Commands	**shutdown**(1M)
Note	Use **shutdown** to go down to run level S or 1.

There is no **sysadm** command to shut down the system. As superuser, you can shut down the system from the shell by changing to the root directory and using the **shutdown**(1M) command.

When we say shut down the system, we generally mean to go to a lower run level in order to perform some administrative task. Often, you will want to shut down to run level 1, the administrative state. Other times, you may want to go down to run level S, single-user mode, so that you can then halt the processor.

When you bring the system down, system buffers are flushed, open files are closed, user processes and daemons are stopped, file systems are unmounted, and superblocks are updated. See "How Run Levels Are Set" later in this chapter for details of what happens as the system comes down through various run levels.

You can only shut down to run levels S or 1 with the **shutdown** command. If you are in run level 3 and want to go to 2, you could shut down to 1 and go back up with the **init 2** command. This prevents remotely-mounted file systems from being lost.

With no options, **shutdown** defaults to run level S, single-user.

Shut Down to Administrative Mode: Run Level 1

Let's assume we're currently in run level 3 and we want to go down to run level 1. In the following example, we'll use the **–i** option to change run levels downward.

Options you can use are:

–y Answers the confirmation query so the command can be run without user intervention. A default of 60 seconds is allowed between the warning message and the final message. Another 60 seconds is allowed between the final message and the confirmation.

–i1 Go to run level 1, administrative mode.

–g0 Allow a grace period of 0 seconds between the warning message and the final message.

Type the following to change to the root directory and shut down the system:

```
# cd / ↵
# shutdown -y -i1 -g0 ↵

Shutdown started.

The system will be shutdown in 0 seconds.
The system is coming down. Please wait.

INIT: New Run Level: 1
#
```

Now you are in run level 1, administrative mode. Local file systems are the only ones mounted. If you want to shut down to power off, type the **shutdown** command again. You will go to run level S, single-user mode.

Shutting Down to Single-User Mode and Power Off

You can shut down to run level S from any other level. This example shows a shut down from run level 1. Type:

```
# cd / ↵
# shutdown -g0 ↵
```

After a few moments, you will be in single-user mode. You can change run levels *upward* at this point with the **init**(1M) command, or, you can use the **halt**(1M) command to stop the processor:

```
# halt -q ↵

CPU HALTED


SCM>
```

Once at the SCM prompt, you may turn off power to the computer.

Recovering from System Failure

Purpose	To return the system to a usable state. To dump memory and image to tape.
Starting Conditions	System halted abnormally: panic, hang, or power failure.
References	**bootparams**(4)

There are three kinds of trouble that can cause your system to stop operating: a panic, a hang, or a power failure.

Panics and Hangs

A panic or a hang means the system has halted because of a serious hardware or software error. If your system is experiencing a panic, a message like the following appears on your system console:

```
DG/UX System Panic.  Panic code  57000104
```

Write down the exact message and panic number. If your system is experiencing a hang, you will not get a message. Instead, you notice that the system does not respond to user input, the screens (including the system console) are all frozen, and so on—in short, the system is dead.

Performing a System Dump

The following procedure is valid whether your system is a stand-alone, a server, or a diskless host. The stand-alone and server hosts are assumed to have local tape drives. The administrators of stand-alone and server hosts control the dump destinations. For our example, we'll make that destination a tape drive. Note that it could also be a logical disk. Diskless systems dump to a file on the server (typically **/srv/dump/***client_hostname*) specified in **/etc/bootparams**.

A system dump actually consists of two kinds of dump: system memory and system image. The Data General Software Support Center needs both kinds of dump to analyze your system's problem adequately.

Completing a System Memory Dump

The way you begin dumping depends on what has happened to the system. If your system has panicked, the system prompts you to begin the dump procedure. If your system is hung, you need to initiate the dump procedure by pressing the "hot key" sequence, which is a series of control-bracket characters:

<Ctrl-]> <Ctrl-[> <Ctrl-]> <Ctrl-[> <Ctrl-]> <Ctrl-[>

To put it another way, hold down the **Ctrl** key while pressing the] [ ] [ ] [keys. This key sequence generates a panic code in the system, causing the system to begin the dump procedure (as with any other panic).

Workstations offer an alternative to the hot key method of interrupting a hung system: press the reset button on the hardware. Do not use this method unless the hot key sequence fails. See your hardware documentation to find the location of the reset button. Pressing the reset button will give you the SCM prompt.

When you get the SCM prompt, type **START 1000** to start the system dump procedure:

```
SCM> START 1000 ↵

DG/UX System aborted by operator.
```

Once the system dump procedure has started, whether you are on a workstation or server, whether your system hung or panicked, it proceeds like this:

```
Do you want to take a system dump? [Y] ↵
```

If you are on a diskless workstation configured to dump across the network to your server, check the size of the **/srv/dump** file system on the server, if possible, and make sure it has enough space for your memory image. Also make sure your workstation has a dump file in the dump directory.

If your system dumps to tape, check the density on your tape drive. Choose the highest density setting available. The higher the density number, the fewer tapes required on which to dump the system. You will be queried for another tape until the dump is complete.

```
Dump destination device? [0]  ↵

Mount tape.  Type the NEWLINE key when tape is ready. ↵

Tape volume 1 completed.

System Dump completed successfully.
```

After the dump, you will see an SCM prompt. You cannot initiate another system dump at this time by using the **START 1000** command.

Running fsck on the Root File System

As always, the **fsck** program will run automatically when the system is rebooted. You want to boot to single-user mode so you can perform the system image dump. If your default run level is not S, you will want to specify it on the **b** command line you use in the SCM. If you boot from an ESDI disk, for example, you use this command line:

```
SCM> boot cied()root:/dgux -s ↵
```

Once in single-user mode, you can take the system image dump.

Completing a System Image Dump

In all cases, when you do a memory dump, you should also dump the tailored system image (usually named **/dgux**) that was running at the time of the system failure. This image contains vital information necessary to interpret the memory dump. The memory dump is useless without the system image. If there is room, dump the system image on the same tape as you dumped the memory. The space you need for both dumps is the total of the memory dump image, which is the size of your computer's physical memory, and your kernel's image, which is the size of the file **/dgux** (or whatever image you had booted before the failure occurred). If you cannot fit both images on one tape, you may use two tapes. Use the following format for both reel and cartridge tapes:

Tape file 0: Memory contents in memory dump format.
Tape file 1: System image in **cpio** format.
Tape file 2: Other files, programs, and so on, in **cpio** format.

Do not use absolute path names, that is, path names starting with the **/** directory.

File 0

Dump the system memory as described earlier. The final tape volume will be rewound when the dump completes.

File 1

Before dumping file 1, you should have dumped the system memory as file 0, and the tape should be rewound to the beginning. To dump file 1, use the following command lines:

```
# mt -f /dev/rmt/0n fsf 1 ↵
# cat /dev/rmt/0n > /dev/null ↵
# cd / ↵
# echo dgux | cpio -ocv > /dev/rmt/0n ↵
```

When the file 1 dump completes, the tape will not rewind, but be positioned so you can dump file 2.

File 2

To dump file 2, use the following command line:

```
# echo file_name | cpio -ocv > /dev/rmt/0 ↵
```

The tape will rewind after this command. For problems that do not involve a system dump, put all files associated with the problem on tape file 0 in **cpio** format using the following command:

```
# ls file_names | cpio -ocv > /dev/rmt/0 ↵
```

When you have completed the memory and image dumps, be sure to label the tape. Your label might look like this:

```
BLACKJACK COMPUTER COMPANY
File 0: memory contents
File 1: system image
File 2: other files
Density: 6250
CPIO format: cpio -c
```

Power Failure

After any abnormal system halt, including a power failure, the file systems are automatically checked with **fsck** upon reboot.

Repairing Damaged Root File System

Purpose	To repair or restore root files damaged due to a system failure. To load a new root if the old one cannot be repaired.
Starting Conditions	The **fsck** program is unable to fix the root file system. The system will not boot.
Caution	A full restore erases everything on the **root** logical disk.
References	**fsck**(1M), Appendix D

You should have already tried running **fsck** before beginning this procedure. This procedure shows you how to get your system back up and running on a repaired root file system.

Since you cannot boot the damaged root, you need to run stand-alone **fsck** on it. If **fsck** cannot fix it, then you will have to load a new root file system onto the **swap** logical disk, and mount the damaged root under the temporary root file system. You can repair the damaged root if you first mount it on the new root. These steps (covered in more detail in the following sections) explain how:

1. Invoke stand-alone **diskman** from release media. On a typical workstation (and on some servers), use this command line:

   ```
   SCM> b st(insc(),4) ↵
   ```

   On other workstations and servers, use this command line:

   ```
   SCM> b st(cisc(),4) ↵
   ```

2. Load the root file system from release media onto logical disk **swap**; exit **diskman**.

3. From the SCM, boot the starter kernel on the logical disk **swap**.

4. Run **fsck** on **/dev/dsk/usr**.

5. Mount **/usr**.

6. Reload damaged files from previous day's backup.

7. Reboot the system as usual.

Repairing the Root Directory

Invoke stand-alone **diskman** from the release media. Select the Initial Installation Menu and choose number 5, "Loading the root file system." The **diskman** program begins its dialogue as follows:

```
5. Load the Root File System

Do you want to run this step? [yes] ↵

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

Enter the Logical Disk Name: [root] swap ↵

Enter tape drive specification in DG/UX system
common format: [st(cisc(0),4)]  ↵

Ready to load the Root File System.
Mount the first release on the tape drive st(cisc(0),4).

Press the NEWLINE key when ready to continue... ↵

Loading ...
```

When loading is complete, the system displays:

```
The Root File System has been loaded.

Press the NEWLINE key when ready to continue.
```

Now you may exit **diskman**. Type **q** and press New Line.

Booting the Starter Kernel

Now, we'll reboot:

```
SCM> reset ↵
SCM> boot cied(0,0)swap:/dgux ↵

Booting cied(0,0)swap:/dgux
          .
          .
          .

INIT: New run level: S

INIT: SINGLE USER MODE
#
```

Now run **fsck** on the **/usr** file system:

```
# /sbin/fsck –xp    /dev/dsk/usr ♪
```

When **fsck** completes, mount **/usr**:

```
# /sbin/mount  /dev/dsk/usr   /usr ♪

# mkdir  /dam_root ♪
# mount /dev/dsk/root   /dam_root ♪
```

Restoring Damaged Files

The following list contains the files that are subject to damage due to a system failure. You can either examine these files for damage or you can restore them from the previous day's backup tape.

- **/dam_root/dgux**
- **/dam_root/etc/passwd**
- **/dam_root/etc/group**
- **/dam_root/etc/init**
- **/dam_root/etc/inittab**
- **/dam_root/etc/ioctl.syscon**
- **/dam_root/bin/sh**
- **/dam_root/bin/su**
- **/dam_root/etc/init.d**

You have two methods for restoring these files:

1. To restore individual files (with **sysadm filerestore**), see Chapter 8, File System Management, and the section "Extracting Individual Files from fsdump Tapes"
2. To restore an entire file system (with **sysadm frestore**), see Chapter 8, File System Management,and the section "Restoring File Systems from fsdump Tapes"

We recommend that you restore individual files whenever possible so that you do not have to destroy your entire root file system with a full restore.

Rebooting with a Repaired Root Directory

After you have repaired or replaced the key files above, shut down and reboot your system.

If Your Repaired Root Directory Will Not Boot

If your repaired root still will not boot, you can try one other operation.

1. Invoke stand-alone **diskman** again from your release media.

   CAUTION: Step 2 will erase all data on the logical disk.

2. Go to the File System Management Menu and select number 1, "Make a File System." Specify **root** as the logical disk.

3. Go to the Initial Installation Menu and select "Loading the Root File System."

4. Load a new **/** (root) file system on the **root** logical disk.

5. Execute **shutdown −g0** and then reboot the system.

6. Use **sysadm filerestore** to restore the files listed earlier in "Restoring Damaged Files."

7. Execute **shutdown −g0** and then reboot the system again.

If your system still will not boot, contact Data General.

Logging System Errors

Purpose	To collect and record system error messages.
Starting Conditions	administrative or multiuser mode
References	**syslog.conf**(5), **syslog**(3), **syslogd**(8), **logger**(1)

Collecting and recording errors from various system facilities is a matter of setting some instructions in a file called **/etc/syslog.conf** and running the **/etc/syslogd** daemon. The daemon collects a variety of system error messages and either records them in a file, or forwards them to users; you determine where output will be directed in your **syslog.conf** file. Entries in this file are composed of two tab-separated fields:

```
selector      action
```

The `selector` field contains a semicolon-separated list of priority specifications of the form:

```
facility.level[;facility.level]
```

where `facility` is an origin such as a user or **mail**, and `level` is an indication of the severity of the error being logged. Values for facility are:

user Messages generated by user processes; this is the default.

kern Messages generated by the kernel.

mail The mail system.

daemon System daemons such as **routed** and **ftpd**.

auth The authorization system: **login, su,** and **getty**.

lpr The printer spooling system.

cron The **cron** or **at** facility.

local0-7 Reserved for local use.

mark For timestamp messages produced internally by **syslogd**.

***** An asterisk indicates all facilities except the mark facility.

news Reserved for the USENET network news system.

uucp Reserved for the UUCP system.

The second half of the selector field is the level. Recognized values for level, in descending order of severity, are as follows:

emerg For panic conditions that would normally be broadcast to all users.

alert For conditions that should be corrected immediately, such as a corrupted system database.

crit For warnings about critical conditions, such as hard device errors.

err For other errors.

warning For warning messages.

notice For conditions that are not error conditions, but may require special handling.

info Informational messages.

debug For messages that are normally used when debugging a program.

none Means do not send messages from the indicated facility to the selected file. For example, a selector of

```
*.debug;mail.none
```

will forward all messages except mail messages.

The `action` field indicates where to forward a message. It can be:

- A file_name beginning with a leading slash.
- A remote hostname prefixed with an @ indicates that messages are to be forwarded to the **syslogd** daemon on that host.
- A comma-separated list of usernames. Indicates that messages specified by the selector are to be written to the named users if they are logged in.
- An asterisk. Indicates that messages specified by the selector are to be written to all logged-in users.

The following is the default **syslog.conf** file:

```
*.err;kern.debug;auth.notice                    /dev/console
kern.debug;daemon,auth.notice;*.err;mail.crit   /usr/adm/messages
*.alert                                         root
*.emerg                                         *
```

The **syslogd** is started automatically by the **rc.syslogd** script.

How Run Levels Are Set

For a detailed discussion of run levels, see "Expert Run Level Information" later in this chapter.

Earlier, we said that a run level is a category of system processes that are activated by **rc** scripts, and that we use run levels to control exactly what processes are running. We will not go into all the details here of how various scripts, directories, and programs interact to specify what processes exist for a given run level. Instead, we will limit this explanation to a general discussion of three major components: the **init**(1M) program, the **inittab**(4) file, and **rc** (run command) scripts.

Below, we show you how these components act in sequence to "make" run level 2 as follows:

1. The administrator types the **init 2** command to change run levels upward from single-user mode S. The administrator thinks of this as switching levels upward to a state of having more services available.

2. The **init 2** command causes the **init** program to read the **inittab** file looking for all entries containing the number 2 in the *level* field. **Init** executes all lines that have 2 in the *level* field.

3. All **rc** scripts associated with run level 2 are started. These scripts result in certain actions, such as turning on accounting, starting scheduler programs, and starting daemons. Run level 2 is therefore defined as all those script-started processes running as a result of the **init 2** command.

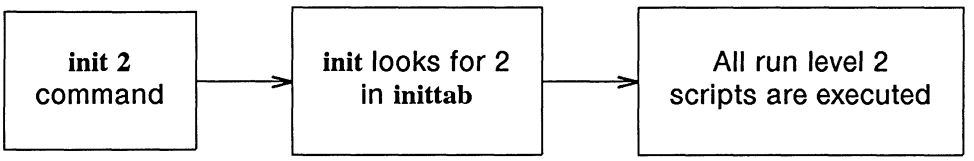


Figure 3-1 *An Example Run Level 2 Sequence*

Run Command Scripts Per Run Level

You can read the **rc** scripts to see exactly what they do. We recommend that you do not modify these scripts. You can add your own if needed. See "Adding Your Own RC Scripts" at the end of this chapter for directions.

For systems with the TCP/IP and NFS packages (in addition to the DG/UX system) loaded and set up, Table 3-2 shows which scripts are started per run level. Note the cumulative effect: the higher the run level, the more processes running. Blanks indicate that a script is not running. From the table, we see that zero scripts are executed in run level S, 0, 5, and 6. Five scripts are executed in run level 1, thirteen scripts in run level 2, and so on.

Table 3-2 RC Scripts Per Run Level

S	0	1	2	3	4	5	6
		rc.tcload	rc.tcload	rc.tcload	rc.tcload		
		rc.update	rc.update	rc.update	rc.update		
		rc.localfs	rc.localfs	rc.localfs	rc.localfs		
		rc.setup	rc.setup	rc.setup	rc.setup		
		rc.daemon	rc.daemon	rc.daemon	rc.daemon		
			rc.usrproc	rc.usrproc	rc.usrproc		
			rc.tcpipport	rc.tcpipport	rc.tcpipport		
			rc.syslogd	rc.syslogd	rc.syslogd		
			rc.ypserv	rc.ypserv	rc.ypserv		
			rc.account	rc.account	rc.account		
			rc.cron	rc.cron	rc.cron		
			rc.lpsched	rc.lpsched	rc.lpsched		
			rc.preserve	rc.preserve	rc.preserve		
				rc.nfsserv	rc.nfsserv		
				rc.tcpipserv	rc.tcpipserv		
				rc.nfsfs	rc.nfsfs		

The following section defines what the scripts in **/usr/sbin/init.d** do, and shows at which run levels they are in effect. See "Expert Run Level Information" later in this chapter for a complete grid layout showing the kill/start mechanism for all scripts active at all run levels.

RC Scripts in /usr/sbin/init.d

rc.tcload Loads the SYAC driver code once for run levels 1, 2, 3, and 4.

rc.update Starts the update daemon in run levels 1, 2, 3, and 4. The update daemon updates disks and preserves editor files left from the last boot.

rc.localfs Mounts local (type **dg/ux**) file systems listed in **/etc/fstab** in run levels 1, 2, 3, and 4; unmounts them in all other run levels.

rc.setup Displays packages that have not been set up at initial boot.

rc.daemon Starts miscellaneous daemons.

rc.links Create, list, or remove links in the **/etc/rc?.d** directories. This occurs only when you setup the DG/UX system. You can, if you wish, use **rc.links** to create, list, or remove your own links.

rc.usrproc Kills all user processes in run levels S, 0, 1, 5, and 6.

rc.tcpipport Sets hostname, host ID, network security, and initializes network I/O boards in run levels 2, 3, and 4. These are not set in any other run levels.

rc.syslogd Starts the **syslog** error logging program in run levels 2, 3, and 4; kills it in all other run levels.

rc.ypserv Starts the **yp** and **portmap** daemons, and sets the domain name in run levels 2 and 3; kills these in all other run levels.

rc.account Starts the **/usr/lib/acct/startup** services and processes in run levels 2, 3, and 4; stops those processes in all other run levels.

rc.cron Starts the **cron** daemon in run levels 2, 3, and 4; kills it in all other run levels.

rc.lpsched Starts the **lpsched** daemon in run levels 2, 3, and 4; kills it in all other run levels.

rc.preserve Invokes the **expreserve**(1M) command in run levels 2, 3, and 4 to recover editor files saved during a system crash.

rc.nfsserv Starts the **portmap, rwalld, mountd, ruserd, nfsd,** and **biod** daemons in run levels 3 and 4; kills them in all other run levels.

rc.tcpipserv Starts the **telnetd, ftpd, tfpd, smtpd, rlogind, rwhod, rshd,** and **rexecd** daemons in run levels 3 and 4; kills them in all other run levels.

rc.nfsfs Mounts all local and NFS file systems listed in **/etc/fstab** in run levels 3 and 4; unmounts them in all other run levels.

Check Scripts

In addition to the run command scripts, the DG/UX system uses four other scripts to set up a properly running environment. These are executed when the system is booted via the **bootwait** action in **/etc/inittab**. Each of these scripts is executed upon the first run level change to levels 1, 2, 3, or 4. For instance, if you boot the system and then go to run level 1, all check scripts are executed. If you then go to run level 2 (without rebooting), then the check scripts are not executed again.

The check scripts are:

chk.date Displays the current system date and allows the administrator to set the correct date. A correct date setting is vital to ensure file creation and modification dates are correct. Also sets time zone based on the **/etc/TIMEZONE** file.

chk.fsck Runs **fsck** on all file systems listed in **/etc/fstab**. The **fsck** program is called with the **−xp** switch which checks file systems in parallel, and only checks those file systems that need checking.

chk.system Performs the following system cleanup and initialization routines:

- Reinitializes the **/etc/ps_data** file.
- Cleans outs the **/var/spool/locks** used by the **uucp** program.
- Makes a **/tmp** directory if one doesn't exist.
- Runs the DG/UX setup scripts via the **init** command the first time the system is booted.
- Checks for accounts without passwords.

chk.devlink At the first run level change, this script automatically creates shortened names for devices in the sequence it finds them. For example, the first tape device will be device 0, the second will be device 1. These could then be specified as **/dev/rmt/0** and **/dev/rmt/1**. Device short names are taken from the **/etc/devlink** file.

Expert Run Level Information

You do not have to read this expert section to operate the DG/UX system. Information here is optional and is provided to enhance your understanding of how run levels work.

The following scripts, directories, and programs in the **/etc** directory interact to specify what processes are running at a given run level:

- The **rc** and **chk** scripts in **/usr/sbin/init.d**.
- The **/etc/rc***N***.d** directories, where *N* is run level S through 6.
- The **/sbin/rc.init** script.
- The **/etc/inittab** file.
- The **/sbin/init** program.

The Fundamentals

The **rc** and **chk** scripts start and stop system services required by all run levels S through 6. These are commonly called "run command" scripts and "check" scripts.

The **init.d** directory contains all of the run command scripts. Some are started at many run levels; some are started at one level and stopped at all other levels; some are started and never stopped until reboot time.

The **rc***N***.d** directories are used to organize and order the set of run command scripts associated with a particular run level. To avoid duplicate scripts and the problem of maintaining consistency among duplicate scripts, the entries in an **rc***N***.d** directory are links to a specific run command script in **init.d**.

The **rc.init** script, when called with an argument S through 6, executes the scripts in the given **rc***N***.d** directory. The processes are invoked according to **K** (kill) and **S** (start) switches.

The **inittab** file is a text file composed of entries that specify which processes will be invoked at which run level.

The **init** program reads the entries in **inittab**, and when they match the specified run level, **init** passes them to a shell for execution.

The Sequence

Let's follow the sequence that occurs when you invoke **init** to set a run level. Assume your system has been booted and is going to be changed from single-user mode, run level S, to multiuser mode, run level 3. We'll track one of the processes invoked, **syslog**.

1. Invoke the **init** program with the argument 3:

   # init 3 ↵

2. The **init** program scans the **inittab** file for all entries containing the run level number 3 in the *run level* field. Then, **init** invokes the `rc.init 3` instruction which is in the *process* field.

3. The **/sbin/rc.init** program uses the run level number 3 as a pointer to directory **/etc/rc3.d**, which contains links to the scripts in **/usr/sbin/init.d**. A script called **rc.syslogd** starts the **syslogd** program.

4. The **rc.init** program then executes all scripts for run level 3; among these is **syslogd**.

The /etc/inittab File

The **init** program relies on the information in the **/etc/inittab** file, whose entries have this format:

id:*level*:*action*:*process*

- *id* is one or two characters that uniquely identify an entry.
- *level* is a character (**s** or **1** through **6**) that determines at what run *level*(s) *action* is to take place. If the *level* field is empty, the action occurs at all run levels.
- *action* can be one of the following:

  bootwait Process the entry only at boot time. **Init** starts the process, waits for its termination and, when it dies, does not restart the process.

  wait When going to *level*, start *process* and wait until it's finished.

  initdefault When **init** starts, it will enter *level*. The *process* field for this *action* is not used.

  once Run *process* once and don't start it again if it finishes.

  respawn If *process* does not exist, start it, wait for it to finish, and then start another.

  ondemand Synonymous with **respawn**, but used only when **a**, **b**, or **c** is specified in *level*.

  off When in *level*, kill process or ignore it.

- *process* can be any executable program, including shell procedures.

You can add a comment to the end of a line by preceding the comment with a pound sign (**#**). The **init** program ignores everything appearing after a pound sign on a line.

Now let's look at lines in our sample **inittab** file and see how the structure makes sense to the DG/UX system:

```
def:S:initdefault:
#
fsc::bootwait:/sbin/chk.fsck    </dev/console 2>&1
dat::bootwait:/usr/sbin/init.d/chk.date    </dev/console 2>&1
set::bootwait:/usr/sbin/init.d/chk.system   </dev/console 2>&1
dev::bootwait:/usr/sbin/init.d/chk.devlink
#
rc0:0:wait:/sbin/rc.init 0  >/dev/console  2>&1
rc1:1:wait:/sbin/rc.init 1  >/dev/console  2>&1
rc2:2:wait:/sbin/rc.init 2  >/dev/console  2>&1
rc3:3:wait:/sbin/rc.init 3  >/dev/console  2>&1
rc4:4:wait:/sbin/rc.init 4  >/dev/console  2>&1
rc5:5:wait:/sbin/rc.init 5  >/dev/console  2>&1
rc6:6:wait:/sbin/rc.init 6  >/dev/console  2>&1
#
con::respawn: /etc/getty console console
00:123:off:/etc/getty tty00 9600
#
01:234:respawn:/etc/getty tty01 9600
02:234:respawn:/etc/getty tty02 9600
03:234:respawn:/etc/getty tty03 9600
```

Figure 3-2 *A Sample /etc/inittab File*

The first line in the sample file is:

```
def:S:initdefault:
```

It sets S, single-user, as the default initialization run level.

The next four lines `fsc`, `dat`, `set`, and `dev` start up four check scripts: **chk.fsck**, **chk.date**, **chk.system**, and **chk.devlink**. These scripts are executed at boot time according to the *bootwait* action of **inittab**(4).

The next seven lines, `rc0` through `rc6`, are instructions for setting run levels 0 through 6. For instance, at run level 3, the **init** program invokes the **rc** scripts in **/etc/init.d** via the links in **/etc/rc3.d** . These scripts perform the functions necessary to start system services for run level 3, and to stop services not associated with run level 3. Standard output and standard error are directed to **/dev/console** for all run levels.

The next two lines, `con` and `00`, identify two terminals to the system: the operator's console (**/dev/console**) and a CRT that can be activated to be an operator's console in run levels 1, 2, or 3. Note that it is turned "off" above.

The last lines are regular terminals enabled only in run levels 2, 3, and 4. The respawn action is set, and the **getty** *process* is invoked for all terminals.

RC Scripts and Check Scripts

When **rc.init** invokes a run level, the characteristics of that run level are produced by scripts in **/usr/sbin/init.d**. There are two types of scripts:

chk.* These scripts are usually run once, at boot time. An example is **chk.fsck** which runs the **fsck** program on file systems.

rc.* These scripts are invoked with either a start or stop argument. An example is **rc.tcload**, which starts or stops the asynchronous terminal I/O controllers.

Init.d Links

Typically, the above scripts exist in **/usr/sbin/init.d**. The **rc** and **chk** scripts are invoked via links in an **/etc/rc*N*.d** directory. Remember, there are eight **/etc/rc*N*.d** directories: **/etc/rcS.d**, **/etc/rc0.d**, **/etc/rc1.d**, **/etc/rc2.d**, **/etc/rc3.d**, **/etc/rc4.d**, **/etc/rc5.d**, and **/etc/rc6.d**. The *names* of the links are labeled as follows:

Snnn.name

or

Knnn.name

The entries have three parts:

S or **K** The first letter defines whether the process should be started (**S**) or stopped (**K**) upon entering the new run level.

nnn The next three characters are a number from 000 to 999. They indicate the order in which the files will be started (S111, S112, S113, and so on) or stopped (K231, K232, K233, and so on).

name The rest of the entry is the script name in **/usr/sbin/init.d**.

All process scripts are specified to be either killed or started when you change run levels. The **init.rc** program executes all **K** scripts first; they are executed from highest ID number to lowest ID number. When all **K** scripts have executed, **S** scripts begin executing from lowest ID number to highest ID number. The point here is that all scripts in **init.d** have links in all **/etc/rc.*N*d** directories; the switches determine what is on and what is off.

For example, the run level 3 link name for **rc.localfs** is **S114.localfs**. This link is in **/etc/rc3.d**.

Let's look at the rest of **/etc/rc3.d**. Type:

```
# cd /etc/rc3.d ↵
# ls -C ↵
K232.tcpipport  S130.daemon     S251.account    S334.tcpipserv
S112.tcload     S210.usrproc    S252.cron       S353.nfsfs
S113.update     S232.tcpipport  S253.lpsched
S114.localfs    S235.syslogd    S254.preserve
S119.setup      S237.ypserv     S315.nfsserv
```

The complete layout of how all **rc** scripts are started and killed is in **/etc/dgux.rclinktab.proto**. Figure 3-3 is a portion of that file:

```
#     run level          id   S   0   1   2   3   4   5   6
##  rc.usrfs             111  K   K   S   S   S   S   K   K
    rc.tcload            112  K   K   S   S   S   S   K   K
    rc.update            113  K   K   S   S   S   S   K   K
    rc.localfs           114  K   K   S   S   S   S   K   K
    rc.setup             119  K   K   S   S   S   S   K   K
    rc.daemon            130  K   K   S   S   S   S   K   K

    rc.usrproc           210  K   K   K   S   S   S   K   K
##  rc.x25port           217  -   -   -   S   S   S   -   -
##  rc.tcpipport         232  -   -   -   S   S   S   -   -
    rc.syslogd           235  K   K   K   S   S   S   K   K
##  rc.ypserv            237  -   -   -   S   S   S   -   -
    rc.account           251  K   K   K   S   S   S   K   K
    rc.cron              252  K   K   K   S   S   S   K   K
    rc.lpsched           253  K   K   K   S   S   S   K   K
    rc.preserve          254  K   K   -   S   S   S   K   K
##  rc.nw_tran           270  K   K   K   S   S   S   K   K

##  rc.nfsserv           315  -   -   -   K   S   S   -   -
##  rc.x25serv           317  -   -   -   K   S   S   -   -
##  rc.tcpipserv         334  -   -   -   K   S   S   -   -
##  rc.nfsfs             353  K   K   K   K   S   S   K   K
##  rc.nw_serv           370  K   K   K   K   S   S   K   K
##  rc.sna               371
##  rc.sna               372
##  rc.sna               373
##  rc.x11               391
```

Figure 3-3 *RC Scripts: the Kill and Start Mechanism*

You can think of all **rc** scripts as being either on (S) or off (K). Since TCP/IP and NFS are optional products, we show their links commented out above.

Changing the Behavior of RC Scripts

The behavior of all **rc** scripts is governed by data and arguments set in a parameters file. There are several such parameters files:

- **/etc/dgux.params** (shipped with the DG/UX system)
- **/etc/tcpip.params** (shipped with TCP/IP)
- **/etc/nfs.params** (shipped with NFS)

For example, the **rc.nfsserv** script starts and stops the **biod** daemon. The more network interaction you have, the more copies of the daemon you would want. The parameter you would change in **nfs.params** is **biod_ARG**. To run four copies of the daemon, for instance, set the parameter to:

```
biod_ARG="4"
```

Adding Your Own RC Scripts

If you add your own run command scripts, follow these rules:

- Place the script in **/usr/sbin/init.d**.
- Link the script to files in appropriate run state directories using the naming convention described above. You can link each one manually or you can write a script to make all the links for you. (For instance **/usr/sbin/rc2.d/S222.***name* should be linked to **/usr/sbin/init.d/rc.***name*.)
- To differentiate your scripts, use three-digit numbers with the *middle* number always being even. Data General uses only odd middle numbers. Attach **S** (start) and **K** (kill) to your links.
- Create a copy of **/etc/dgux.rclinktab.proto** called (for example) **rc.linktab**; this serves as a record of *additions* to the **rc** scripts. Put your new scripts in this file, indicating the K and S switches in each run level.

Example: Adding Two RC Scripts

Let's do an example. Let's say you have two scripts you'd like to add. They are **rc.turbo** and **rc.charger**. You'd like them activated in run levels 2 and 3. The link names in **/usr/sbin/rc2.d** and **/usr/sbin/rc3.d** might be **S220.turbo** and **S222.charger**. The link names in the remaining directories would all start with K. Your new scripts would be started in run levels 2 and 3 after **rc.usrproc**.

Create **/local/etc/rclinktab**. Add your new scripts and the corresponding K and S switches. Your file should look like the following.

```
#  run level          id  S  0  1  2  3  4  5  6

   rc.turbo           220 K  K  K  S  S  K  K  K
   rc.charger         222 K  K  K  S  S  K  K  K
```

Figure 3-4 *Your RC Script File*

Now link your new scripts to each entry in the **/usr/sbin** directories **rcS.d**, **rc1.d**, **rc2.d**, **rc3.d**, **rc4.d**, **rc5.d**, and **rc6.d**. Do this manually with the **ln**(1) command or write a script to make all the links for you. Assuming you wrote a script named **rc.mylinks**, you would run that script as follows:

```
# /usr/sbin/init.d/rc.mylinks -ln /local/etc/rclinktab ↵
```

End of Chapter

Chapter 4
System Configuration Management

The first part of this chapter lists the administrative logins for the DG/UX system and gives procedures for setting the system clock, setting defaults for making backup tapes, setting a new root password, and reconfiguring the system. Error messages from the **config**(1M) program are listed after the reconfiguration procedure.

The second part of this chapter offers suggestions on general operating policies, performance management, and on defining the best usage patterns for your system. This chapter concludes with listings and definitions of the tuneable parameters for the DG/UX system.

The major sections of this chapter are

- File Ownership and Access
- System Configuration Management Procedures
- Operating Policy
- Improving Performance
- Tuning System Parameters

File Ownership and Access

In the DG/UX system, a file's owner controls access to the file. The **root** login, which exists on all systems, can override any permission settings and can execute, open, read, delete, or change any file in the system. If you know the **root** password, you can become **root** by executing the **su**(1) command with no arguments, or by logging in as **root**. DG/UX systems also have an administrative login, **sysadm**, that has all the privileges of **root**. We recommend that you use the **sysadm** login instead of the **root** login. When you log in as **sysadm**, you're in the **/admin** home directory instead of in the **/**. Using this login keeps your personal administrative files out of **/**.

To avoid having **root** own too many files, the DG/UX system disperses file ownership over nine login names. Four of these login names function normally, that is, you can become these with **su**. The other five are for system use only, that is, you never actually log in with them. If you look at **/etc/passwd**, you'll see that the system logins have an asterisk (*) in the password field, meaning that no one can log in with these.

To perform administrative functions, you can log in as **root** or **sysadm**. Either way, you have the superuser privilege that allows you to override security features. As **root** or **sysadm**, you can use **su** to become **adm** or **nuucp**. The **uucp** login is used only by systems setting up UUCP file transfer connections. For more information on UUCP, including a discussion of the difference between the **uucp** and **nuucp** logins, see Chapter 12. Table 4-1 shows the administrative and system logins.

Table 4-1 Administrative and System Logins

Login	How It Is Used
root	This login has no restrictions. It overrides all process and file permissions. The **sysadm** login has the same unlimited access privileges as **root**.
sysadm	Same as **root**, except the login directory is **/admin**.
sys *	This login owns the files in **/usr/src**.
bin *	This login owns the files in **/usr/bin**.
adm	This login owns the files in **/var/adm**.
uucp *	This login owns the object and spooled data files in **/usr/lib/uucp**. To make **uucp** connections, systems log in to other systems with the **uucp** login and initiate file transfers via **/usr/lib/uucp/uucico**.
nuucp	The system administrator can log in to the system as **nuucp** and perform general administrative tasks.
lp *	This login owns the object and spooled data files in **/var/spool/lp**.
daemon *	This is the login of the system daemon, which controls background processing.
mail *	This is the login of the electronic mail facilities.

In Table 4-1 only those entries without asterisks may be used as actual logins; the others are for system use only.

System Configuration Management Procedures

The following procedures are covered in this chapter:

- Setting system time and date
- Setting tape archive defaults
- Recovering forgotten root password
- Reconfiguring the system

If you select the **sysmgmt** command from the **sysadm** Main Menu, the System Configuration Management Menu is displayed with the following choices:

```
                    System Configuration Management

1 datetime       Set date, time, time zone, daylight savings time
2 newdgux        Build and install a new DG/UX kernel
3 tapedefaults   Set defaults for tape use

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, or q to QUIT:
```

Setting Time and Date

Purpose	To synchronize system time with clock time or to reset the system time and date.
Starting Conditions	administrative or multiuser mode
sysadm menu	sysmgmt
Commands	**datetime**
References	**date**(1), **cron**(1M)

When you select **datetime**, the system responds as follows:

```
The current time zone is Eastern Standard Time (EST).

The current date and time are: 06/22/89 08:35.
```

Next, you will be asked what time zone you'd like. The default is the current time zone, in our case Eastern Standard Time. If you'd like to change to another time zone, type **?** to list the available choices. For now, let's say that you want the default. As shown below, you can simply press the New Line key to select the default value that appears in brackets.

```
Time Zone? [3] ♪
Does your area use Daylight Savings Time? [yes] ♪
Month? [06] ♪
Day of the month?[22] ♪
Year? [89] ♪
Hour? [08] ♪
Minute? [36] ♪

The date and time have not changed.
The time zone has not changed.


Press the NEWLINE key to see the sysmgmt menu [?, ^, q]:
```

Setting Tape Defaults

Purpose	To set the machine defaults for magnetic tape use.
Starting Conditions	administrative or multiuser state
Commands	**tapedefaults**
References	**dump2**(1M), **dumptab**(4)

The **tapedefaults** command sets the defaults for using magnetic tape. The default information that you supply here is used by other processes when you do things like modify dump cycles, dump files to tape, and restore files from backup tape, for example. You need to supply the default tape drive and tape medium. For a list of accepted values for tape media, see the first column of the **/etc/dumptab** file. If **dumptab** does not contain an entry for your tape media, refer to the **dumptab**(4) manual page to add an entry.

When you select **tapedefaults**, the system queries you for the tape drive and tape medium. The default is **cartridge**, which represents a QIC-150 150MB tape. Your dialog may look like this:

```
Default tape drive? 0 ↵

Default Tape Medium? cartridge ↵

The defaults have been added.
Press the NEWLINE key to see the sysmgmt menu [?, ^, q]:
```

Recovering Forgotten Root Password

Purpose	To recover from a forgotten **root** password.
Starting Conditions	Single-user state, S, or multiuser mode
Commands	**passwd, wall, sync, fsck**
References	**passwd**(1), **passwd**(4), **wall**(1M), **sync**(1M), **fsck**(1M)

If you should forget the **root** password, this procedure shows you how to set a new one. You may be in either of two situations when you realize that you have forgotten your **root** password:

- You are logged on as **root**, and you have the **#** prompt. Simply run the **passwd**(1) command and set a new **root** password.

- You are not currently logged on as **root**. For instance, you may be logged on as yourself and have the normal shell prompt. Because there is no way to access the **root** login, you will have to bring the system down in what is called an "unclean" halt.

If the latter condition is the case, go to the system console and do the following:

1. Use **wall**(1M) to warn users that the system is about to go down.

2. Issue the **sync**(1M) command.

3. Wait five seconds, and reset your system. You reset your system either by pressing the reset button or by entering the hot-key sequence at the system console. The hot-key sequence consists of three pairs of angle brackets (][][][) pressed while you hold down the control key:

   <Ctrl-]> <Ctrl-[> <Ctrl-]> <Ctrl-[> <Ctrl-]> <Ctrl-[>

   Resetting the system takes you to the SCM prompt.

4. Reboot your system.

5. Once you are in single-user mode, reset the root password with the **passwd**(1) command.

Reconfiguring the System

Purpose	To build and install a new **/dgux** kernel. To incorporate changes resulting from hardware and software changes to the system.
Note	Servers and clients do not run the same kernels. Most OS servers will configure kernels for themselves and for OS clients. You must reboot after configuring a kernel in order for the new kernel to go into effect.
sysadm menu	sysmgmt
Commands	**newdgux**
References	**config**(1M), **make**(1)

You need to rebuild the kernel when the physical or software configuration of your system changes. The only way the system can understand such changes is when you rebuild the kernel from its source files. This executable kernel depends on the information in text files in **/usr/etc/master.d**.

The **/usr/etc/master.d/dgux** file contains device information for all possible DG/UX system devices and default values for system parameters. If you have additional products, such as TCP/IP or NFS, then you will have configuration files for them in this directory also. For example, you might have **/usr/etc/master.d/tcpip**.

NOTE: You may look at the master files in **/usr/etc/master.d**, but do not edit them. To override any settings in a master file, add the appropriate entries to your system file (discussed later). Do not duplicate any of the master files in the master directory, or you will not be able to build your kernel.

Besides the files in directory **master.d**, another directory that is important for configuration is **/usr/src/uts/aviion/cf**. It contains prototype files from which your combined system file is assembled. Prototypes have the form **system.dgux.proto**. The first time you invoke **newdgux**, the prototype file is copied to **/usr/src/uts/aviion/Build/system** (a symbolic link to **/var/Build/system**).

The system file contains *your* changes to the system's parameter variables and a checklist of all devices on the system.

In this procedure, the **newdgux** command queries you for the name of the system file. Then **newdgux** requests the name of the editor you want to use. When you've finished editing, **config**(1M) runs on the system file and produces program code in a file named **conf.c**. Next, a build is invoked which compiles **conf.c** and links the libraries in **/usr/src/uts/aviion/lb** to build the new kernel image. Finally, the old kernel can be saved and the new one installed.

Configuring on a Stand-alone, Server, or Diskless Host

Configuring your system means building a kernel that includes parameters set to values that are appropriate for your environment. To build a kernel, use the **newdgux** command. The **newdgux** command first allows you to edit the system file.

The system file consists of prototype files (typically for the DG/UX system and the TCP/IP and ONC/NFS network products) concatenated together to form one large file that you can edit. The system file contains declarations for a number of tunable parameters, some of which you may need to change. The system file also contains text that explains the various parameters.

To build a kernel for a stand-alone system or a server, see Chapter 2. The section titled "Step 15: Building a Custom Kernel" in Chapter 2 discusses the changes you need to make in the system prototype files.

Building Client Kernels

Generally, the OS server manager will configure kernels for client hosts on the OS server host. Let's configure a kernel for diskless client **dg1**.

```
# 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, being sure to comment out the server-specific items, including entries for disks and other devices that your diskless clients do not have. For some **vi** editing hints and a discussion of the system file, see Step 17 (Building a Custom Kernel) in Chapter 2. When you have finished editing the file, save the file and exit from **vi**. Next, you will see the following:

```
Ready to Configure a Kernel? [yes]  ↵
sysadm will now run config on system
.
.
.
Config succeeded.
```

If **config** fails, see the following section, Configuration Error Messages. Correct the problem and execute **sysadm newdgux** again. After **config** succeeds and the build concludes, the new kernel is installed in a location accessible to the diskless client.

```
Install the New Kernel? [no] y ↵
For a diskless client of this host? [no] y ↵
Kernel Path Name?
     [/srv/release/PRIMARY/root/_Kernels/dgux.diskless] ↵
Save the old kernel? [y] ↵
Link all PRIMARY clients to the New Kernel? [y] ↵
```

Our `Kernel Path Name?` response placed the kernel for our diskless client in the same logical disk in which the client's root resides. If clients want to boot the same kernel, they must all reside in the same logical disk with the kernel they want to boot. We saved the old kernel, but you may want to delete it. Last, we took the default and linked all clients to the same kernel.

The diskless client's new kernel will take effect when the client reboots.

Configuration Error Messages

The following error messages are generated by the **config**(1M) program. Some errors originate in the master file, others in the system file. Errors in the system file are more common since you change it as a result of updating your configuration. Errors in the master file are less common; normally you don't alter the master file. You would alter the master file if you installed a new device driver.

Category	Message
Either you edited a master file and in doing so duplicated an entry in it, or you duplicated an entire master file. Make sure **/usr/etc/master.d** contains only the original, unchanged master files that you received with your software.	A master file entry for *entry* already exists.
Cannot open a file or directory. Make sure the master file is in the proper directory and that it is named correctly.	Cannot open the master file [*master_file_name*]. Cannot open master file directory.
The file in the master directory is not a legal master file.	No section definition found in master file [*master_file_name*]. This file will be ignored.

There may be incorrect information in your system file, i.e., you may have misspelled a device name. Check that entries in your system file match those in your master file. Keyword errors pertain to the system file. Device flag and flag errors pertain to the master file.	Unknown Keyword: [*keyword_name*] Unknown Device Flag: [*device_flag*] Unknown Flag: [*flag*]
Cannot allocate space for internal structures. This is the result of an error returned from **malloc**(3C). This is related to user logical address space. Check the master file directory for duplicate files.	Cannot Allocate Space. Allocate device entry: Out of memory. Allocate stream entry: Out of memory. Allocate protocol entry: Out of memory. Cannot allocate an alias structure. Cannot allocate a keyword structure. Error allocating Configured Device entry: Out of memory.
Illegal arguments sent to **config**.	Usage: config [-t] [-c] [-m master_file_directory] [-c conf_file] system_file
Illegal format for a master file or system file line. Device code errors and keyword errors are associated with the system file. The other errors in this category are associated with the master file.	Illegal Master file line: [*line*] Illegal protocol number: [*protocol number*] Illegal Domain number: [*domain number*] Illegal Socket number: [*socket number*] Illegal Device Code: [*device code*] No value associated with the keyword: [*keyword_name*]. Keyword will be ignored.
Two devices share the same major number. Change the incorrect one in the master file.	Warning: Device [*device_name*] and [*device_name*] have the same major number [*number*]. Warning: Device [*device_name*] on major number [*number*] configured.

Operating Policy

Sometimes situations arise that require you to shut down the system with little or no notice to users. Try to provide as much advance notice as possible about events affecting the use of the system. When you must take the system out of service, be sure and tell users and diskless client managers when to expect the system to be available again. Edit the Message Of The Day file (**/etc/motd**) to keep users informed about changes in hardware, software, policies, and procedures.

At your discretion, the following items should be done as prerequisites for any task that requires the system to leave the multiuser state.

- When possible, schedule service-affecting tasks to be done during periods of low system use. For scheduled actions, use the Message of the Day to inform users of future actions. Use **rwall** or **mailx** to inform diskless client managers.

- See if anyone is logged in before taking any actions that affect users. Use the **who**(1) command to display all system users. For immediate actions, use the **wall**(1M) and **rwall**(1M) commands to send broadcast messages announcing system down times. Always give users enough time to finish whatever they are doing and log off before taking stand-alone or server systems down.

Maintaining a System Log

We recommend that you maintain a complete set of records, both on paper and electronically. A system log book can be a valuable tool when trouble shooting transient problems or when trying to establish system operating characteristics over a period of time. Some of the things that you should consider entering into a log book are:

- What devices are configured into the current kernel

- Equipment and system configuration changes (dates and actions)

- A record of system panics and hangs

- Maintenance records (dates and actions)

- A record of recurring problems and fixes.

Whatever format you choose for your log, make sure that the system log and the types of items noted there follow a logical structure. Think of the log as a diary that you update on a periodic basis. To a large measure, how you use your system will dictate the form and importance of maintaining a system log.

Improving Performance

This section contains suggestions for improving the performance of your system through file system efficiency. You may want to reread the introduction and the planning sections in Chapter 2 again.

The last section gives tuneable parameter charts, definitions, and recommendations.

File System Organization

You can reduce the overhead of file access several different ways. As file systems are used, the blocks of individual member files tend to become physically scattered around the disk(s) and I/O becomes less efficient. This scattering yields poor ordering of blocks with files and poor directory structure. If you have more than one disk, balance heavily used file systems across your physical disks. For more information, see Chapter 2 for disk planning, Chapter 7 for disk management, Chapter 8 for file system management, and Chapter 9 for procedures on obtaining file information.

Maximizing System Usage

To ensure maximum system performance, you should check for

- Less important jobs interfering with more important jobs
- Unnecessary jobs
- Jobs running during peak hours that could just as easily run during off-peak hours
- The efficiency of user-defined features, such as **.profile** and PATH

Getting Process Information

Use the **ps −ef** command to obtain information about active processes. This command gives a "snapshot" of what is going on, which is useful when you are trying to identify what processes are loading the system. Things will probably change by the time the output appears; however, the entries that you should be interested in are TIME (minutes and seconds of CPU time used by processes) and STIME (time when process first started).

If you spot a "runaway" process, one that uses progressively more system resources over a period of time while you are monitoring it, you'll probably need to stop the process immediately via the **kill −9** command. When you have a real runaway, the process continues to eat up system resources until everything grinds to a halt.

When you spot processes that take a very long time to execute, you should consider using **cron**(1M) to execute the job during off-hours.

Checking User PATH Variables

This information applies only to **sh**(1) users.

PATH is searched upon each command execution. Before outputting "not found," the system must search every directory in PATH. These searches require both processor and disk time, thus changes here can help performance.

Some things that you should check for in user PATH variables are:

- Path Efficiency

  PATH is read left to right, so the most likely places to find the command should be first in the path (**/usr/bin**). Make sure that a directory is not searched more than once for a command.

- Convenience and Human Factors

  Users may prefer to have the current directory listed first in the path (**.:/usr/bin**), but note that putting the current directory first can lead to breaches in security. If a program having the same name as a system command, like **ls** or **pwd**, for example, is in the user's current directory, the user will inadvertently execute it while intending to execute the "real" system command. Depending on the nature of the "fake" system command that exists in the local directory, the results could be undesirable, or at best, unpredictable. It is particularly critical that the superuser profiles (**root** and **sysadm**) not have the current directory first on their path. In fact, it is safer to leave the current directory out of a superuser path altogether.

- Path Length

  In general, PATH should have the least number of required entries.

- Large Directory Searches

  Avoid searching large directories if possible. Put any large directories at the end of PATH.

Shift Workload to Off-Peak Hours

Examine users' crontab files in **/var/spool/crontab** to see if there are jobs scheduled for peak hours that could just as well run during off hours. You may find the **ps**(1) command and accounting reports (See Chapter 15) helpful in determining what processes have the greatest effect on system performance. Encourage users to run large, non-interactive commands (such as **nroff**(1) or **troff**(1)) at off-peak hours. You may also want to run such commands with a low priority by using the **nice**(1) or **batch**(1) commands.

Tuning System Parameters

Tuneable system parameters set various table sizes and system thresholds to handle the expected load on your system. You'll find the default tuneable parameter values are adequate for most configurations and applications. If your application has special performance needs, you may have to experiment with different combinations to find an optimal set. The only parameters you may have to adjust are NODE, MACH, TZ, DUMP, NPROC, and PERCENTNFS. See the definitions of these in this section for details. To set tuneable parameters, edit the values in your system file when building a new kernel with **sysadm newdgux**.

Uname Configuration Variables

The **uname**(1) and **uucp** programs use the Uname configuration variables. Each of these variables must be a character string no longer than eight characters, not including the trailing null character. These parameter variables are also listed in **/usr/etc/master.d/dgux**.

Table 4-2 shows the Uname configuration variables.

Table 4-2 Uname Configuration Variables

Parameter	Definition	Default Value
NODE	The UUCP node name of the system (sales, sys31, and so on).	"no_node"
MACH	The name of the system's underlying hardware.	"AViiON"
SYS	The name of the operating system.	"dgux"
REL	The number of the system's release.	"4.30"
VER	The version number of the operating system.	"00"

Setup and Initialization Configuration Variables

The setup and initialization configuration variables are also listed in **/usr/etc/master.d/dgux**. These variables set the system initialization parameters shown in Table 4-3 below.

Table 4-3 Setup and Initialization Configuration Variables

Parameter	Default Value
DST	1
TZ	300
DUMP	"st(insc(),4)"
DEBUGGER	&sc_null_debugger
DEBINITCMDS	"mode er on\n"
INIT	&init_run_sbin_init
STARTER	0

DST Specifies the type of Daylight Savings Time being used. The different types are defined in **/usr/include/sys/time.h**.

TZ Represents the time zone of your system in minutes west of Greenwich Mean Time (GMT). Set this according to your time zone. The default is USA Eastern time: 300 minutes west of GMT.

DUMP A string holding the name of the default system dump device in DG/UX common device specification format. This device is the default tape device used to do a system memory dump after a PANIC or a halt. Set this to your primary tape device.

DEBUGGER Represents the kernel debugger to be used by your system. The default is the null debugger stub. The DG/UX Kernel Debugger can be specified simply by listing the keyword **DEBUGGER** in your system configuration file, which causes the implied value (**&deb_debugger_request**) to be used. See *Using the DG/UX™ Kernel Debugger* for details.

DEBINITCMDS A string holding a list of zero or more commands to be executed by the DG/UX Kernel Debugger (if present) before displaying the first debugger prompt. If the null debugger is used, this variable has no effect. Note that debugger commands must be separated and terminated by the C language newline character, **\n**.

INIT Represents the internal function that is executed upon system booting. The default function calls **/sbin/init**. Do not change this parameter.

STARTER A Boolean variable indicating whether or not the system will ask to configure additional devices upon booting. The default is 0 (FALSE). Use 1 for TRUE. We recommend that you use the default.

CPU and Process Configuration Variables

The CPU and process configuration variables are also listed in **/usr/etc/master.d/dgux**. These variables set the parameters shown in Table 4-4 below.

Table 4-4 CPU and Process Configuration Variables

Parameter	Default Value
NPROC	256
NCPUS	0
MAXSLICE	500
MAXUP	25
MAXBUFAGE	30

NPROC Specifies the maximum number of processes the system can have at one time. For various sized systems use the following values: small 96; medium (default) 256; and large 512. The overall number of processes needed depends on the number of terminal lines available, the number of processes spawned by each user, and the number of system processes and network daemons. If the maximum number of processes is used up, the **fork**(2) system call will return the error EAGAIN.

NCPUS Specifies the number of processors to run. If set to 0 (the default), all available CPUs will be used. Any other value specifies that number of CPUs to run. If the value specified is more or less than the number of CPUs present, a message to that effect is printed when the kernel is booted. Note that on a uniprocessor system, this parameter has no real effect since the one processor will always be run.

MAXSLICE Specifies the maximum time in milliseconds a user process can run before being suspended. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it MAXSLICE number of milliseconds. MAXSLICE is normally 500 milliseconds (1/2 second).

MAXUP Specifies the maximum number of processes that a non-superuser can have in existence at one time. The entry is normally in the range of 15 to 25. This value should not exceed

the value of NPROC (NPROC should be at least 10% more than MAXUP). This value is per user identification number, not per terminal. For example, if ten people logged in with the same user ID, the default limit would be reached very quickly.

MAXBUFAGE Specifies the maximum ages in seconds that a modified buffer can reach before it is written to disk.

Pseudo-Device Unit Count Variable

The pseudo-device unit count variable sets the number of units a specified pseudo-device will have. (No count variables are needed for real devices; any units present are useable.)

PTYCOUNT The number of pseudo-terminal device pairs (**/dev/ttyp*** and **/dev/ptyp***) that will be created when the system is booted. The default value is 64. This parameter is used for **telnet**(1C), **rlogin**(1C), and **shl**(1).

PMTCOUNT The number of pseudo-magnetic tape devices used for access to remote magnetic tapes (**/dev/pmt/***). The default is 20.

File System Configuration Variables

The file system configuration variables are also listed in **/usr/etc/master.d/dgux**. These variables set the file system parameters shown in Table 4-5 below.

Table 4-5 File System Configuration Variables

Parameter	Default Value
ACCTON	5
ACCTOFF	2
PERCENTNFS	75
CDLIMIT	INT32_MAX
FREEINODE	4
FREERNODE	4

ACCTON, ACCTOFF

If the free space in the file system in which the accounting file resides becomes less than the percentage specified by **ACCTOFF**, then no further accounting records will be written. When the free space reaches **ACCTON** percent, then the writing of accounting records will resume. **ACCTOFF** should always be smaller than **ACCTON**.

PERCENTNFS Specifies the percentage of system buffers that are available for NFS clients. The percentage is a maximum, an upper limit on the amount of buffer space that NFS can use. Diskless workstations should set this parameter to 100.

CDLIMIT Specifies the maximum size in bytes that a non-superuser file may attain. The constant INT32_MAX is equal to 2,147,483,648.

FREEINODE Specifies the maximum ratio of in-use inodes to free inodes in the system. To improve performance on systems where you open a large number of files repeatedly, set this parameter to a higher value.

FREERNODE Specifies the maximum ratio of in-use rnodes to free rnodes in the system. To improve performance on systems where you open a large number of files repeatedly, set this parameter to a higher value.

STREAMS Configuration Variables

The STREAMS configuration variables are associated with STREAMS processing. We recommend that you use the default values supplied. These variables are also listed in **/usr/etc/master.d/dgux**. They set the STREAMS parameters shown in Table 4-6 below.

Table 4-6 STREAMS Configuration Variables

Parameter	Default Value
PERCENTSTR	20
STRLOFRAC	80
STRMEDFRAC	90
NQUEUE	2048
NSTRPUSH	9
STRMSGSZ	4096
STRMCTLSZ	1024

PERCENTSTR Specifies the percentage of system memory (after initialization) that is reserved for STREAMS buffers.

STRLOFRAC Specifies the threshold percentage of in-use STREAMS buffers beyond which low-priority requests for STREAMS buffers will be denied. This variable is included to help prevent deadlock by starving out low-priority activity. The recommend value of 80 works well for current applications. This parameter must always be in the range 0 <= **STRLOFRAC** <= **STRMEDFRAC**.

STRMEDFRAC Specifies the threshold percentage of in-use STREAMS buffers beyond which medium-priority requests for STREAMS buffers will be denied. This parameter must always be in the range **STRLOFRAC** <= **STRMEDFRAC** <= 100.

NQUEUE Specifies the maximum number of STREAMS queues (Streams plus instances of STREAMS modules) that may exist at any one time on the system. A minimal stream contains two queue pairs: one for the Stream head and one for the driver. Each instance of a module on a Stream requires an additional queue pair.

NSTRPUSH Specifies the maximum number of STREAMS modules that may be pushed on any one Stream. This is used to prevent an errant user process from consuming all the available queue pairs on a single STREAMS module.

STRMSGSZ Specifies the maximum number of bytes allowed in the data portion of a STREAMS message. A module maximum packet size of **INFPSZ** defaults the maximum packet size to this value. If it is larger than necessary, a single **write** or **putmsg** can consume an inordinate number of data blocks.

STRMCTLSZ Specifies the maximum number bytes allowed in the control portion of a STREAMS message. The control part of a message created with **putmsg** is not subject to the constraints of the minimum or maximum packet size, so this value is the only way of providing a limit for the control part of a message.

Semaphore Configuration Variables

These semaphore configuration variables are also listed in **/usr/etc/master.d/dgux**. The variables shown below in Table 4-7 are associated with interprocess communication (IPC) semaphores.

Table 4-7 Semaphore Configuration Variables

Parameter	Default Value
SEMMNI	10
SEMMSL	25
SEMOPM	10
SEMVMX	32767
SEMUME	10
SEMAEM	16384
SEMAPM	30

SEMMNI Specifies the maximum number of unique semaphore sets that may be active at any one time on the system.

SEMMSL Specifies the maximum number of semaphores that a semaphore set may contain.

SEMOPM Specifies the maximum number of semaphore operations that can be executed per **semop**(2) system call.

SEMVMX Specifies the maximum value a semaphore may have. The default is the maximum value for this parameter.

SEMUME Specifies the maximum number of undo entries per undo structure.

SEMAEM Specifies the maximum value of the adjustment for adjust-on-exit. The value is used whenever a semaphore value becomes greater than or equal to the absolute value of **semop**(2), unless the program has set its own value. The default value is the maximum value for this parameter.

SEMAPM The maximum number of processes that may specify adjust-on-exit at one time.

Shared Memory Configuration Variables

The tunable parameters shown below in Table 4-8 are associated with inter-process communication shared memory. These parameters are also defined in the **/usr/etc/master.d/dgux** file.

Table 4-8 Shared Memory Configuration Variables

Parameter	Definition	Default Value
SHMMNI	Specifies the maximum number of shared memory identifiers system wide. Each entry contains 52 bytes.	100
SHMSEG	Specifies the number of attached shared memory segments per process.	6
SHMMAX	Specifies the maximum shared memory segment size.	131072
SHMMIN	Specifies the minimum shared memory segment size.	1

Message Configuration Variables

These parameter variables are also listed in **/usr/etc/master.d/dgux**. They set the message parameters shown in Table 4-9 below.

Table 4-9 Message Configuration Variables

Parameter	Definition	Default Value
MSGMNI	Specifies the maximum number of message queues that may exist in the system at one time.	50
MSGTQL	Specifies the maximum number of outstanding messages that may exist in the system at one time.	1024
MSGMNB	Specifies the maximum number of bytes that a message queue may contain.	4096
MSGMAX	Specifies the maximum number of bytes that a message may contain.	2048

End of Chapter

Chapter 5
Release Management

A release is a directory environment, or *release area*, that contains software that provides operating system service. We distinguish between the *primary* release and *secondary* releases. The primary release (DG/UX 4.30), runs as the operating system on servers and stand-alone systems. Any other releases (such as the DG/UX 4.20 system) are secondary.

The primary release resides in **/** (root) and **/usr**. OS clients have access to the primary release via **/srv/release/PRIMARY**. Secondary releases reside in **/srv/release/***release_name*.

The **/usr** file system contains host-independent programs and data files that users typically do not change. The rationale for this organization is to put in one place all of the operating system components that do not vary among systems using the same release of DG/UX; consequently, an OS server and any of its OS clients attached to the primary release may save disk space by sharing the same **/usr** file system.

A system's root file system, on the other hand, is the directory that contains data files, configuration files, and programs (such as kernels) that may vary from system to system; therefore, each system needs to have its own root file system. On servers and stand-alone systems, the root is of course the **/** directory. OS clients, on the other hand, have root directories assigned to them under the **/srv/release/***release_name***/root** directory. OS client root directories are based on a prototype found in **/usr/root.proto**.

The **releasemgmt** menu of **sysadm** shows the procedures for managing releases:

```
            Software Release Management

 1 addrelease       Add a software release area
 2 delrelease       Delete a software release area
 3 lsrelease        List information about software releases
 4 loadpackage      Load software packages into a software release area
 5 setuppackage     Set up packages in a software release area
 6 makesrv          Create the initial /srv directory tree
 7 lstoc            List the table of contents from a release tape

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Adding a Software Release Area

Purpose	To set up the files and directories needed by a secondary release.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**addrelease**
Note	You must run **sysadm makesrv** before **addrelease**. The release name must not already be in use.

A release is a collection of software packages intended for a specific architecture and operating system. To add a release means to create the appropriate directories and files that will be used by the release. Once a release has been added, you can load software into it. Before creating a new release area, make sure you have the necessary disk space in the release's eventual destination. If the logical disk containing your **/srv/release** directory is not big enough, you need to recreate the logical disk with more space or mount another logical under **/srv/release** to hold the new release.

To add a new release, follow these steps:

1. Execute **sysadm makesrv** (if you have already run this command, you do not have to run it again; if you have not already run this command, **sysadm** will tell you to run it.).

2. Execute **sysadm addrelease**.

3. Execute **sysadm loadpackage**.

4. Execute **sysadm setuppackage**.

First, let's add a secondary release named **88k_myOS_2**. We start by typing this command:

```
# sysadm addrelease ♪
```

Our dialog proceeds as follows.

```
Running subcommand 'addrelease' from menu 'releasemgmt',
Software Release Management

Release Area? 88k_myOS_2 ♪
Usr Directory? /srv/release/88k_myOS_2/usr ♪
Share Directory? [/srv/share] ♪
```

```
Client Root Parent Directory? [/srv/release/88k_myOS_2/root] ♪
Client Swap Parent Directory? [/srv/swap] ♪
Release 88k_myOS_2 has been added.  You may now use loadpackage.
```

The last system response tells us that the appropriate **sysadm** and directory entries have been made. These are

/srv/release/*release_name***/usr**
For executables and data files that individual clients will not need to change.

/srv/release/*release_name***/usr/root.proto**
The prototype for the root directories of clients using this release. For the primary release, this is the same as **/usr/root.proto**. **addclient** makes a copy of the prototype root for new clients. Clients are free to customize their own root directory.

/srv/share
A directory that you can use at your own discretion, intended to contain whatever programs or files your clients may hold in common.

/srv/release/*release_name***/root**
The directory containing the root directories of OS clients. Client root directories are named the hostname of the client.

/srv/swap
The directory containing the swap areas of OS clients. Client swap areas are named the hostname of the client.

/srv/admin/releases/*release_name*
The file containing a list of directories associated with this release.

All we have done so far is to create a release area. Next, we need to load software into this newly created release area with **sysadm loadpackage**.

Deleting a Release Area

Purpose	To delete the files and directories used by a specific release.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**delrelease**

Deleting a release means deleting the release directory tree and erasing files used by **sysadm** for the given release. You can only delete releases that have no attached clients. Note that the primary release (**/usr**) cannot be deleted with this function.

Below, let's delete a release named **88k_foo_9.0**. When you select **delrelease**, the system responds as follows:

```
Release Area? 88k_foo_9.0 ↵
Do you really want to delete 88k_foo_9.0? [no] y ↵
Removing 88k_foo_9.0.  This may take a while.
Release 88k_foo_9.0 has been deleted.
```

Deleting this release removes the following directories from your system:

- **/srv/release/88k_foo_9.0/usr**
- **/srv/release/88k_foo_9.0/root**
- **/srv/admin/releases/88k_foo_9.0**

Listing Release Information

Purpose	To display information about all releases or about a specific release.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**lsrelease**

Listing releases consists of printing an entry for each release area with the release name, directory path names, installed packages, and attached clients. You can select information on all releases or on a specific release.

First, we'll display information on all releases. After that, we'll display information on a specific release, **88k_myOS_2**.

```
Release Area? [all] ♪

Release Area              Usr Path Name
------------              -------------
PRIMARY                  /usr
88k_myOS_2               /srv/release/88k_myOS_2
88k_foo_9.0              /srv/release/88k_foo_9.0
```

To display information about the specific release, enter the following at the prompt:

```
Release Area? [all] 88k_myOS_2 ♪

Release Area:         88k_myOS_2
Usr Directory:        /srv/release/88k_myOS_2/usr
Root Directory:       /srv/release/88k_myOS_2/root
Swap Directory:       /srv/swap

Packages:
V-windows 2.0
Zapfiles  1.0

Clients:
dg1
```

Loading Software into a Release Area

Purpose	To add software to a specific release.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**loadpackage**
Note	After loading software with **loadpackage**, use **setuppackage** to set up the software. There is no **sysadm** function for deleting software from a specific release area; you must do this by hand.

Use this function when you want to load software into a secondary release area or when you want to add software packages to an existing release. This function loads software into the **/usr** and **root** prototype directories for a given release.

Software is loaded into **/**, **/usr**, or a subdirectory of **/usr** such as **/usr/opt**. Be sure to follow software package instructions requiring you to mount file systems to receive software. When you add software to a release that is running on diskless clients, a copy goes into each diskless client root. This may include the server's root.

Below, we'll load an example package named **X11**. You don't need to specify the name of the package you want to load because **loadpackage** reads all package names on the tape you are loading, then asks you if you want to load each one. After you have determined which one you want to load, the actual load begins.

The location where a package will be loaded is specified on the tape table of contents. Use **sysadm lstoc** to read the tape table of contents. Before you load a package you should have already planned disk space for the package.

When you type the **sysadm loadpackage** command, the system responds as follows:

```
Release Area? [PRIMARY] ↵
Tape Drive? [0] ↵
Is the Tape Mounted and Ready? [yes] ↵
Load package X11? [yes] ↵
List file names while loading? [yes] ↵
Mount Volume 1.
Is the tape mounted and ready? [yes] ↵
```

File names in the package appear here.

```
loadpackage for X11 is finished.
```

Setting Up Software in a Release Area

Purpose	To supply information needed by software associated with a specific release.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**setuppackage**

This function runs setup scripts for a given package in a given release area. Use this function after you have loaded a package with **sysadm loadpackage**. The **setuppackage** function locates all setup scripts that have not been run from a software package and allows the user to execute them. If you had already loaded a package named **xray-vision**, you would set it up as follows:

```
# sysadm setuppackage ♪
```

The dialog, for setting up a package called **xray-vision**, continues as follows:

```
Release Area? [PRIMARY] ♪

The following packages have setup scripts that have not been run:

xray-vision   telepathy_1   fortune_teller

Package Name? [all] xray-vision ♪

Processing setup scripts for package xray-vision.
```

At this point, you would begin responding to the queries specific to the given setup script. When you finish setting up a package, **setuppackage** exits unless you had originally invoked **setuppackage** to set up all packages, in which case **setuppackage** continues.

Creating the srv Directory Tree

Purpose	To create the **/srv** directory tree for releases and clients.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**makesrv**

The **sysadm releasemgmt** and **clientmgmt** menus use the **/srv** directory tree for internal database storage for primary and secondary releases, and for clients. The first time you use **releasemgmt** or **clientmgmt**, you will be asked to run the **makesrv** command. You can run this command as many times as you want; it only creates a **srv** directory tree if none already exists. A typical interaction with this command is as follows.

```
# sysadm makesrv ↵

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

Making the PRIMARY release area.
Making the server client entry.
makesrv is finished.
```

Listing Tape Table of Contents

Purpose	To list tape contents.
Starting Conditions	administrative mode (run level 1) or higher
sysadm menu	releasemgmt
Commands	**lstoc**

The **lstoc** function decodes and prints the table of contents from a release tape or software package tape. Use this function to see where the **sysadm loadpackage** command will attempt to load a given package. After mounting the tape and typing the **sysadm lstoc** command, the system responds as follows:

```
Tape Drive? [0] ♪
Is the tape mounted and ready? y♪
```

File names on the tape appear here.

End of Chapter

Chapter 6
Client Management

This chapter tells how to manage OS clients. You may add, delete, and list clients, create sets of client defaults, and set default releases for clients attached to multiple releases. Chapter 1 introduces servers, clients, netbooting, and the servnet.

To add OS clients to a server, perform these steps:

1. Use **sysadm clientdefaults**.
2. Use **sysadm addhost** for each client (see Chapter 13, Network Management).
3. Use **sysadm addether** for each client (see Chapter 13, Network Management).
4. Use **sysadm addclient** for each client.
5. Set up software packages on the client system.

NOTE: Because of basic operating system differences, the **sysadm** procedures for adding DG/UX OS clients may not completely set up foreign OS clients. AViiON servers will support foreign OS clients, but Data General cannot supply the foreign system-specific information necessary for a complete setup. Consult the foreign system's documentation and/or your Data General representative.

The **clientmgmt** menu of **sysadm** shows the functions you use to manage clients:

```
                Diskless Client Management


 1 addclient        Add a diskless client entry
 2 clientdefaults   Create or modify a set of diskless client defaults
 3 delclient        Delete a diskless client entry
 4 lsclient         List information about diskless clients
 5 bootdefault      Change the default release for a diskless client
 6 addxterminal     Add an X terminal display bootstrap client
 7 delxterminal     Delete an X terminal display bootstrap client
 8 lsxterminal      List X terminals that are served by this system

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Creating or Modifying Client Defaults Sets

Purpose	This procedure allows you to create and modify sets of client defaults for the **addclient** function.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Note	The primary release is the one running on the OS server.
Commands	**clientdefaults**

A defaults set is a group of attributes that you choose. You may name this group anything you want. A defaults set is useful when adding OS clients because a set allows you to assign the same defaults to more than one client.

This function records defaults that will be used by the **addclient** function when you are adding diskless clients to your system. This function displays the names of any existing sets and allows you to define others. Assume we already have two sets defined, **68kset** and **dgset**.

Below, we show the defaults set we defined in installation Step 20. To set these defaults, type

```
# sysadm clientdefaults ♪
```

The system responds as follows:

```
The current set names are:

68kset    dgset
Defaults Set Name? dgset ♪
Release Area? [PRIMARY]  ♪
Default Swap Size? [16m]  ♪
Default Home Directory? [/home] /sales/accounts ♪
Default Kernel? [/srv/release/PRIMARY/_Kernels/dgux.diskless] ♪
Default Bootstrap File? [/usr/stand/boot.aviion] ♪

Defaults for set dgset have been assigned.
```

Adding a Client to a Release

Purpose	This function associates a client workstation to a specific operating system that resides on a remote server's physical disk.
Starting Conditions	run level 1 or higher You must perform **sysadm addhost** and **addether** (in that order) for the client before using **addclient.**
sysadm menu	clientmgmt
Commands	**addclient**

After a release has been set up on a server, you can add a diskless client to that release. Adding a client means creating a root directory for the client and then recording information about the client. You may want to use **sysadm clientdefaults** to set up defaults before adding any clients. You will be asked for a defaults set name and then asked if you want to use all of the defaults in that set. You can take all defaults, or change any entries as they are displayed.

Adding a client to a release does not change the client's default boot release. You change a client's default boot release with the **sysadm bootdefault** command.

For this example, let's assume that we have already created a defaults set named **dgset** (as we did in Chapter 2). Also, we must have already made entries for a given client with **sysadm addether**.

Let's add an example client, a Data General diskless workstation named **dg2** that will be running the primary OS. The dialog proceeds like this:

```
# sysadm addclient ♪

Server's Host Name on Client's Network? [sales] ♪
Client Host Name? dg2 ♪
Defaults set name? [dgset] ♪
Use all defaults from dgset? yes ♪
Creating client root.
Creating client swap file.
Creating client /etc/fstab.
Creating client /etc/hosts.
Creating client /etc/tcpip.params.
Creating client /etc/nfs.params.
Creating the kernel link.
Creating the bootstrap link.
Client dg2 has been added.
Do you wish to add another client? [yes]  no ♪
```

Files Created by sysadm addclient

When you add a client, that client inherits the server's environment. This means that the client receives copies of the server's various parameter files: **dgux.params**, **tcpip.params**, **nfs.params**, and so on. Clients can modify these as they want. The **sysadm addclient** command creates a number of directories and files for a new client: a root directory, a swap file, client parameter and data files, a link to a common diskless client kernel, a link to a common diskless client secondary bootstrap, an entry in the server's **bootparams** file, entries in the server's **exports** file, and client/release data files used by **sysadm**.

Client Root

A diskless client's root space is created on the server's disk when **sysadm** copies the files in **/srv/release/PRIMARY/usr/root.proto** to the client root area. The client root area is **/srv/release/PRIMARY/root/***client*, where *client* is the client's host name.

Client Swap File

The client swap file is **/srv/swap/***client*.

Client Parameter and Data Files

The following files are created in **/srv/release/PRIMARY/root/***client***/etc**.

fstab The **addclient** function adds the following entries to the client **/etc/fstab** file (*server* is the server's host name, and *client* is the client's host name):

```
server:/srv/release/PRIMARY/root/client   /     nfs  rw x 0
server:/usr                               /usr  nfs  ro x 0
server:/srv/swap/client                   swap  swap sw x 0
server:client_home_directory              swap  swap sw x 0
```

addclient adds the entry for the client's home directory only if the home directory is on the server. *fstab* should contain entries for all other file systems that a client needs to access. See **fstab**(4).

tcpip.params The **addclient** function copies the server's **tcpip.params** file to the client's area and changes server references to client references.

nfs.params The **addclient** function copies the server's **nfs.params** file to the client's area and changes the YP class to **client**.

Note that these changes to the **tcpip.params** and **nfs.params** files result in a minimal environment that allows a client to boot; the client is not fully set up. Running **sysadm setuppackage** on the client will complete the setup.

Client Kernel Link

A diskless client boots **/srv/release/PRIMARY/root/_Kernels/dgux.diskless** by default. The **addclient** function links this file to a file in the client's root space, **/dgux**.

Server Bootstrap Link

A diskless client uses a secondary bootstrap to load **dgux.diskless** over the network. This default bootstrap file is **/usr/stand/boot.aviion**. The **addclient** function makes a symbolic link from **/usr/stand/boot.aviion** to **/tftpboot/***client_ip_addr*, where *client_ip_addr* is the client's Internet address, expressed in hexadecimal. The client's Internet address comes from the **/etc/hosts** file, which you should have already updated with the **addhost** command.

Server bootparams File

The **addclient** function puts the following entry in **/etc/bootparams** for an OS client named **dg1** on OS server **sales**.

```
dg1  root=sales:/srv/release/PRIMARY/root/dg1 \
     swap=sales:/srv/swap/dg1 \
     dump=sales:/srv/dump/dg1
```

The entry is actually a single logical line. In the example entry above, backslashes at the ends of the first two lines escape out the New Line characters, effectively removing them and making the three physical lines one logical line.

Server exports File

The **addclient** function puts the following entries in **/etc/exports** for an OS client named **dg1** on OS server **sales**.

```
/srv/release/PRIMARY/root/dg1  -access=dg1,root=dg1
/srv/swap/dg1  -access=dg1,root=dg1
```

Client/Release Data Files for sysadm

The **addclient** function creates **/srv/admin/clients** and **/srv/admin/releases**. These directories contain files used by the **sysadm** program. Do not modify the contents of these directories or files yourself.

Deleting a Client from a Release

Purpose	This function removes a client system from a specific release.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Commands	**delclient**

Deleting a client means deleting a client's root directory tree for a given release. Also, **sysadm** information on a client/release pair is erased. This function displays each client/release pair and asks if it should be deleted. Below, let's delete a client named **dg2**. We begin by typing the following:

```
# sysadm delclient ↵
```

A sample dialog would continue like this:

```
Client Host Name? dg2 ↵
Release Area? PRIMARY ↵

Do you really want to delete dg2 from PRIMARY? [no] y ↵

Client dg2 has been deleted from PRIMARY.

Do you want to delete another client? [no] ↵
```

Listing Client Information

Purpose	List information on hosts on the servnet: release name, root directory, swap file, and kernel.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Commands	**lsclient**

You can use this function in two ways. You can display information about all clients or you can display detailed information about a single client. We'll do both. First, we'll do information for all clients. Note that the server is listed among the clients since it is running the primary operating system.

Begin by typing:

```
# sysadm lsclient ♪
```

A sample dialog would continue like this:

```
Client Host Name? all ♪

Client Name            Release Name
-----------            ------------
 server                PRIMARY
 dg1                   PRIMARY
 sun1                  68k_sunos_4.0
```

Or, you could display detailed information about a single client.

```
Client Host Name?  dg1 ♪

Client Name:      dg1
Release Area:     PRIMARY
Root Directory:   /srv/release/PRIMARY/root/dg1
Swap File:        /srv/swap/dg1
Swap Size:        16777216
Boot File:        /usr/stand/boot.aviion
Kernel File:      /srv/release/PRIMARY/root/_Kernels/dgux.diskless
```

Changing a Client's Default Boot Path

Purpose	Changes which OS release an OS client will boot by default.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Commands	**bootdefault**

In the case where a client is attached to more than one release, **bootdefault** allows you to change the default boot path. If we assume that diskless client **dg2** is attached to two releases, then we would want to set the default boot case. We begin by typing the following:

```
# sysadm bootdefault ♪
```

A sample dialog might proceed as follows:

```
Server's Host Name on Client's Network? [sales] ♪

Client Host Name? dg2 ♪

Release Area? 88k_foo_9.0 ♪
Updating the server's bootparams file.
Updating the server's exports file.
Creating the bootstrap link.

The default release for dg2 is 88k_foo_9.0.
```

Adding a client to a new release with **addclient** does not change the client's default boot path; you must change the default boot path explicitly with **bootdefault.**

Adding an X Terminal Display Bootstrap Client

Purpose	This function adds an X terminal bootstrap client.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Commands	**addxterminal**

Select option 6, **addxterminal**, to set your server up so that an AViiON AVX-30 network display station (or similar X terminal) can boot. This command sets up certain files so that when the X terminal boots, your server can send it the proper bootstrap file to get it started. For more information on the AVX-30 network display station, see the AVX-30 network display station documentation and software release notice.

Before executing **addxterminal**, you must add the X terminal's Internet address to the server's **/etc/hosts** file (with **sysadm addhost**), and you must add the X terminal's Ethernet address to the **/etc/ethers** file (with **sysadm addether**). The X terminal displays its Ethernet address when you turn on power. Get the Internet address from your network administrator.

After you have run **addhost** and **addether**, use **addxterminal** to set the server up so that it will boot the X terminal. We begin by typing the following:

```
# sysadm addxterminal ♪
```

When **addxterminal** asks you for the host name, supply the host name of the X terminal. When prompted for the X terminal's bootstrap file, take the default. A sample dialog follows:

```
Client Host Name? meatloaf ♪
Bootstrap File? [/usr/opt/X11/xtd/avx30boot]  ♪
Creating the bootstrap link.

Client meatloaf has been added.
Do you wish to add another X terminal client? [yes]
```

At this last prompt, you have the liberty of continuing to add more X terminal clients.

Deleting an X Terminal Display Bootstrap Client

Purpose	This function deletes an X terminal bootstrap client.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Commands	**delxterminal**

Select option 7, **delxterminal**, to delete the files on your server that allow an X terminal client (like the Data General AVX-30 network display station) to boot. We begin by typing the following:

```
# sysadm delxterminal ♪
```

A sample dialog might proceed as follows:

```
Client Host Name?  meatloaf ♪
Do you really wish to stop serving meatloaf? [no]  y ♪

The boot file for client meatloaf, /tftpboot/80DE2101,
has been removed.
Do you wish to stop serving another X terminal client? [yes]
```

At this point, you may elect to remove another X terminal client. After deleting an X terminal client, you may wish to remove the client's **/etc/hosts** entry (with **delhost**) and **/etc/ethers** entry (with **delether**).

Listing X Terminal Clients

Purpose	This function lists the X terminal bootstrap clients served by your system.
Starting Conditions	run level 1 or higher
sysadm menu	clientmgmt
Commands	**lsxterminal**

To list the X terminals set up to boot from your server, select option 8, **lsxterminal**:

```
# sysadm lsxterminal ♪
```

The system responds by asking which X terminal host you want to list. As in the following example, you may take the default to list all X terminal clients:

```
X Terminal Host Name? [all]  ♪

Client    Address         Bootstrap  Linked to
------    -------         ---------  ---------
victor    128.223.14.c8   80DE0E42   /usr/opt/X11/xtd/avx30boot
anubis    128.223.14.02   80DE0EBA   /usr/opt/X11/xtd/avx30boot
meatloaf  128.223.33.33   80DE2101   /usr/opt/X11/xtd/avx30boot
```

End of Chapter

Chapter 7
Disk Management

The **diskman** program manages your physical and logical disks. It is composed of multi-layered menus which call up DG/UX programs to perform various disk tasks.

The major sections of this chapter are as follows:

- Invoking the Diskman Program
- Using Diskman Menus
- Disk Management Procedures
- Command Line Options

Invoking the Diskman Program

The DG/UX system comes with two versions of **diskman**:

- **Stand-alone**: Invoke this version directly from tape when you are installing the DG/UX system, or boot the disk file **stand/diskman** from your **usr** logical disk.
- **Stand-among**: Invoke this version as superuser through the **sysadm diskmgmt** command or directly from the shell. Stand-among **diskman** is **/usr/sbin/diskman**. See "Command Line Options" at the end of this chapter.

Chapter 2 contains definitions and explanations of physical and logical disks. The chapter details the association of file systems with logical disks. Be sure that you are thoroughly familiar with these concepts before you alter disks.

Some of the major tasks you can do with **diskman** are

- Format physical disks.
- Create, delete, and copy logical disks.
- Display physical and logical disk information.
- Create, check, and repair file systems.

Using Diskman Menus

The **diskman** program works very much like **sysadm**; **diskman** menus lead you through the functions, query for information, respond to selections, and display error messages. At the end of a function, you return to the previous menu. When you are offered a default, simply press the New Line key to choose it. Table 7-1 shows how to use **diskman**. Remember: HELP is available by typing the question mark, **?**. Use it freely.

Table 7-1 Using Diskman Menus

User Input	Description
^	Return to previous menu.
?	Print HELP message, then redisplay menu.
number	Choose menu item by entering a number.
number?	Give information on the item number specified.
q	Exit from **diskman**. Enter this command from anywhere.
New Line	Same as entering the default response.

If **diskman** receives invalid input, it displays a message indicating what features *valid* input should have, and then displays the menu or question again. If an operation fails, **diskman** prints an error message prefaced with:

```
**** Error:
```

If you type ? after receiving an error message, **diskman** displays a HELP screen.

The following sections discuss the operations you can select from the **diskman** Main Menu.

Disk Management Procedures

```
            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:
```

NOTE: This chapter does not give you the information you will need for initial installation. See Chapter 2 for instructions on using the stand-alone version of **diskman** to install the DG/UX system.

To make a physical disk accessible to users on your system, follow these steps:

1. Format the physical disk.
2. Register the physical disk.
3. Create a logical disk on the physical disk.
4. Create a file system on the logical disk. At this point, you are finished with **diskman**.
5. Using **sysadm addfsys**, add the file system to **/etc/fstab** and create a mount directory for the file system.
6. To make the file system accessible to users, mount it on its mount directory.

This chapter covers steps 1 through 4. Chapter 8 covers steps 5 and 6.

Physical Disk Management

As always, if you are not sure how to respond to any query, type ?. Let's begin looking at the **diskman** menus, starting with selection 1:

```
            Physical Disk Management Menu

1. Register, Deregister, or List Registered Physical Disks
2. Add, Recover, or Display Bad Blocks on a Physical Disk
3. Display a Physical Disk's Layout
4. Display a Physical Disk's Label
5. Format a Physical Disk

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

Enter Choice:
```

After you select and complete one of the procedures in this menu, you may return to this menu and either quit or select another option.

Registering, Deregistering, or Listing Registered Physical Disks

Purpose	To register a physical disk so that logical disks associated with that physical disk are available for use. To deregister a physical disk no longer in use. To list the physical disks that are already registered.
diskman menu	Physical Disk Management Menu
References	**diskman**(1M)

You should register any disk you add and deregister any that you intend to remove. If you have a CDROM, diskette, or magneto-optical (optical SCSI) device on your system, you should unmount the file system and deregister the device before removing the disk media. When you register a physical disk, you are ensuring that all of the logical disks on that physical disk are known to the system. The following menu is displayed when you select choice 1 from the Physical Disk Management Menu, "Register, Deregister, or List Registered Physical Disks."

```
        Physical Disk Registration Menu

1. Register a Physical Disk
2. Deregister a Physical Disk
3. List Registered Physical Disks

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

Enter choice:
```

Register a Physical Disk

To register a physical disk, select option 1. You need to know the disk's device specification in order to register it. If the disk is already registered when you try to register it, **diskman** responds with:

```
Physical Disk cied(0,0) is already registered.
```

If the physical disk is not formatted, or if it is open by another user, registration will fail, and the following will be displayed:

```
Physical Disk cied(0,0) could not be registered.
```

Deregister a Physical Disk

To deregister a physical disk, select option 2. You need to know the disk's device specification in order to deregister it. If the disk is not registered, the system displays:

```
Physical Disk cied(0,1) is not registered.
```

If a logical disk on the physical disk is in use (that is, open by another user), deregistration fails, and your screen will display the following:

```
Physical Disk cied(0,1) could not be deregistered.
```

If you have deregistered the disk in order to disconnect the device from the system, make sure you shut the system down and turn off power to the system and disk before removing it. If you have deregistered a CD disk, optical disk, or diskette, you may remove the disk media from the device.

List Registered Physical Disks

To list registered physical disks, select option 3. This option displays the long names of the currently registered physical disks on the system.

```
Currently registered physical disks are:

    cied@18(FFFFEF00,0)
    cied@19(FFFFEF00,1)

Press the NEWLINE key when ready to continue.
```

If no disks are registered:

```
There are currently no Registered Physical Disks.
```

Adding, Recovering, or Displaying Bad Blocks on a Physical Disk

Purpose	To locate bad blocks and remap them to a replacement good block in the bad block table. To unmap blocks from their assigned replacement block in the bad block table.
diskman menu	Bad Block Management Menu
References	**diskman**(1M)

Disk units occasionally develop flaws in the disk itself. Most disk units keep track of these bad parts without depending on the operating system to do it for them. When the DG/UX systems detects a flaw on a disk, it flags the block (512-byte portion of disk space) as bad and finds a good block to replace it. The operating system takes care of remapping reads and writes intended for the bad block so that they go to the good replacement block instead. A part of the disk called the bad block remap area contains good blocks reserved specifically for this purpose, to replace blocks that go bad elsewhere on the disk.

The **diskman** utility creates the bad block remap area when it creates system areas. When **diskman** creates the bad block remap area, it offers a default remap area size that is based on the size of the disk. You may specify another size if you like, but the default size should be sufficient for any but the poorest quality disks.

When you select number 2, "Add, Recover, or Display Bad Blocks on a Physical Disk," from the Physical Disk Management menu, the following menu is displayed:

```
        Bad Block Management Menu

1. Add Bad Blocks to a Physical Disk's Bad Block Table
2. Recover Bad Blocks from a Physical Disk's Bad Block Table
3. Display a Physical Disk's Bad Block Table

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

Enter Choice:
```

These selections are used as explained in the following sections.

Adding Bad Blocks to a Physical Disk's Bad Block Table

To add bad blocks to the bad block table, select option 1. Bad blocks are parts of the physical disk that may be unreliable for storage and retrieval of information. The operating system keeps track of bad blocks by listing them in its bad block table.

There may be other bad blocks besides the ones listed in the bad block table. If you suspect that a particular block is unreliable or diagnostics have shown a block to be unreliable, you can add that block to the bad block table explicitly with this selection.

When you select the Add Bad Blocks option, **diskman** asks you to specify the physical disk and the physical disk address of the bad block that you want to add to the table. You do not have to provide the address of a good block for the remapping. A sample dialog follows:

```
Enter the Physical Disk specification in DG/UX common
format: cied(0,1) ↵

The Physical Disk must be registered for this operation.
Do you want to register it? [y] ↵

Enter the Physical Disk addresses of bad blocks, one per line.
Use the NEWLINE key or 0 to terminate the list.
```

Press the New Line key after each address you enter. The second statement is repeated until you press New Line without entering a bad block address. Below, let's enter two addresses:

```
Enter the Physical Disk Address: 201104 ↵
Enter the Physical Disk Address: 376104 ↵
Enter the Physical Disk Address: ↵
```

If any of the bad blocks are readable (not bad according to the system), you are informed about them and asked if you still want to add those blocks to the Bad Block Table.

```
The following blocks are readable:

201104

Do you want to add them to the bad block table anyway? [y] n ↵

The following blocks were added to the Bad Block Table:

376104

Press the NEWLINE key when ready to continue.
```

If the bad block remap area does not have any more good blocks (which is highly unlikely), **diskman** returns an error message telling you that it cannot add the bad blocks to the table. When this happens, dump the disk contents to tape and use **diskman** to recreate the system areas on the disk, being sure to specify a larger bad block remap area size. Then reload the disk from tape.

Recovering Bad Blocks from a Physical Disk's Bad Block Table

To recover bad blocks from the bad block table, select option 2. This option recovers the remapped blocks that no longer need to be remapped. To remap a block means to designate another block to use in place of the bad or unreliable block. For example, block 100000 might be remapped to another block. After getting your disk serviced, however, you want to remove block 100000 and all other formerly-bad blocks from the bad block table. For this option, a sample dialog could proceed like this:

```
Enter the Physical Disk specification in DG/UX common
format:  cied(0,1) ♪

The Physical Disk must be registered for this operation.
Do you want to register it? [y] ♪

Beginning Bad Block recovery...
```

As each block is recovered, a message is printed:

```
Recovered block at Physical Disk Address xxxxxx.
Recovered block at Physical Disk Address xxxxxx.

xx bad blocks were recovered on Physical Disk cied(0,1).

Press the NEWLINE key when ready to continue.
```

Displaying Physical Disk's Bad Block Table

To display a physical disk's bad block table, select option 3. After you have formatted a disk, use this option to display and remap bad blocks (if any). If you supply the disk drive specification, the **diskman** program will list the addresses (in decimal form) where bad blocks are located.

An example dialog for this selection follows.

```
Enter the Physical Disk specification in DG/UX common
format: cied(0,1) ♪

The Physical Disk must be registered for this operation.
Do you want to register it? [y] ♪

Bad Blocks exist at the following Physical Disk Addresses:

433560
466000
466660

Press the NEWLINE key when ready to continue.
```

Displaying a Physical Disk's Layout

Purpose	To display a table showing how areas on a given physical disk are being used.
diskman menu	Physical Disk Management Menu
References	**diskman**(1M)

Select option 3 to display information on logical disk pieces, sizes, physical addresses, disk capacity, and free space on a physical disk.

The Primary System Area (PSA), created when you format the physical disk, contains the initial bootstrap program and information describing the layout of the physical disk. You will be asked for the disk device specification in DG/UX common format. You may create logical disks in multiple pieces (discussed later), so the layout display includes logical disk piece (LDP) numbers for logical disks. The following example display shows that piece 1 of **thor** and piece 2 of **comm** are located on physical disk **cied(0,0)**.

```
System Areas on Physical Disk cied(0,0):

Area Name                 LD Piece    Physical Disk   Size
                          Number      Address
                                      of Area

Primary System Area       ....        0               8
System Bootstrap Area     ....        8               500
Secondary System Area     ....        508             8
Primary Bad Block Table   ....        516             5
Primary LDP Table         ....        521             9
Bad Block Remap Area      ....        530             315
Secondary Bad Block Table ....        845             5
Secondary LDP Table       ....        850             9
thor                      1 of 1      859             20000
comm                      2 of 2      20859           10000
(Free Space)              ....        30859           331472

Press the NEWLINE key when ready to continue...

Total Physical Disk Size: 362331 blocks.
Unallocated Space: 331472 blocks.

Press the NEWLINE key when ready to continue...
```

Displaying a Physical Disk's Label

Purpose	To display a physical disk's space allocation.
diskman menu	Physical Disk Management Menu
References	**diskman**(1M)

Disk labels contain information that the system requires to write to the disk, read from it, and keep track of damaged disk blocks. The system does not use the labels on SCSI disks, but you must label them anyway when you initialize the disk. When you select the "Display a Physical Disk's Label" option from the Physical Disk Management menu, you enter a dialog like the one that follows.

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

The Physical Disk must be registered for this operation.
Do you want to register it? [y]

Cycles per drive:              1632  interleave:              1
Visible cycles per drive:      1626  head skew:               4
Tracks per cylinder:             15  Cylinder skew:          20
Sectors per track:               52  Head group skew:         0
Bytes per logical sector:       512  Spares per track:        1
Bytes per unformatted sector:   512  Byte per data preamble:  0
Mgs's defect area start sector:   0  Bytes per id preamble:   0
Bytes in mfg. defect info:        0  Base head for volume:    0
Number of relocation areas:       0  Bytes in gap 1:          0
Sectors per relocation area:      0  Bytes in gap 2:          0
Next relocation sector:           0
A final short sector does not exist.
SMD-Extended addressing is not used.
```

Formatting a Physical Disk

Purpose	To format a physical disk.
diskman menu	Physical Disk Management Menu
References	**diskman**(1M)

When you select the "Format a Physical Disk" option from the Physical Disk Management menu, the following menu is displayed:

```
        Physical Disk Formatting Menu

1. Install a Disk Label on a Physical Disk
2. Perform Hardware Formatting on a Physical Disk
3. Create DG/UX System Areas on a Physical Disk
4. Install Bootstraps on a Physical Disk
5. Perform Surface Analysis on a Physical Disk
6. All of the above

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

Enter Choice:
```

We recommend that you select item 6 to perform all steps.

Installing a Disk Label on a Physical Disk

Use choice 1, "Install a Disk Label on a Physical Disk," to install a particular disk label on a physical disk. A disk label contains the disk geometry (e.g., tracks per cylinder, bytes per sector) Every disk must contain this information so that it can be accessed by the operating system. This function is used in the default case, choice 6. The interaction is shown below:

```
Enter Choice: 6 ♪
Enter the Physical Disk specification in DG/UX
common format: cied(0,2) ♪

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

If the disk already has a label, you will be asked if you want to reinstall it.

```
A label already exists on this physical disk.
Do you want to reinstall the label? [n]
```

If a label exists and you decide to reinstall it, answer **y** to the query. You then see the following display:

```
   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:
```

If you select one of the numbers above, the disk label will be installed. If you select "None of the Above," you will need to enter 22 parameters for your disk. You then receive the following series of prompts:

```
Enter the total cylinders per drive:
Enter the OS visible cylinders per drive:
Enter the tracks per cylinder:
Enter the sectors per track:
Enter the bytes per logical sector:
Enter the bytes in mfg defect information:
Enter the bytes per unformatted sector:
Enter mfg defect information start sector:
Enter the number of relocation areas:
Enter the sectors per relocation area:
Enter the interleave:
Enter the head skew:
Enter the cylinder skew:
Enter the head group skew:
Enter the spares per track:
Enter the bytes per data preamble:
Enter the bytes per id preamble:
Enter the base head for volume:
Enter the bytes in gap 1:
Enter the bytes in gap 2:
Does the drive use SMD extended addressing?
Does the drive have a final short sector?
```

If the install was successful, the following message appears:

```
Disk Label has been installed.
```

Performing Hardware Formatting on a Physical Disk

Option 2, "Perform Hardware Formatting on a Physical Disk," is not yet implemented.

Creating DG/UX System Areas on a Physical Disk

If you want to create a system area on a physical disk, choose option 3, "Create DG/UX System Areas on a Physical Disk." The DG/UX operating system needs system areas to describe the file systems on a disk. There is a Primary System Area (PSA) and a Logical Disk Piece Table that describes the logical disks that are on the physical disk. Another system area is the Bad Block Table which the OS uses to remap bad blocks.

The interaction is:

```
The Physical Disk must be deregistered for this operation.
Do you want to deregister it? [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 cied(0,2).

Do you want to continue? [y] ↵

The physical disk cied(0,2) is nnnnnnn blocks in size.
Enter the number of blocks to allocate for the remap
area: [189] ↵
```

When you create system areas on a disk, **diskman** calculates the size of the remap area based on the overall size of the disk.

Installing Bootstraps on a Physical Disk

The **diskman** program contains low-level bootstrap programs used to boot the DG/UX system image. These programs are written to disk by **diskman** when you format the physical disk. If you are adding a new release of the DG/UX system or the contents of your disk have been destroyed, you will need to reinstall the bootstraps. To do so, choose option 4, "Install Bootstraps on a Physical Disk." The interaction is shown below:

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

The Physical Disk must be deregistered for this operation.
Do you want to deregister it? [y] ↵

Installed Bootstraps on the Physical Disk cied(0,2).
```

If you are not going to perform other operations on the disk at this time, you need to reregister it to use it.

Performing Surface Analysis on a Physical Disk

If this disk is on a controller that performs hardware bad block remapping, you will be informed and asked whether surface analysis is still desired. If it is still desired, select option 5, "Perform Surface Analysis on a Physical Disk." The resulting interaction will look like the following:

```
Running surface analysis on this model of disk is not
required because the disk controller maintains a hardware
bad block table.  See the manual for more details.

Do you want to perform surface analysis on this Physical
Disk? [y] ♪
```

Generally, it is not necessary to run surface analysis. If you decide to run surface analysis, we recommend that you run *all* test patterns. The process takes about 20 minutes duration per 100 Mbytes, depending on your physical disk model and CPU. An average is about an hour. Before surface analysis begins, you are queried:

```
You have the option of running all test patterns or a
single test pattern.

Do you want to run all the test patterns? [y] ♪
The Physical Disk cied(0,0) is 100000 blocks in size.
```

As surface analysis proceeds, the system displays:

```
Beginning Surface Analysis...
Surface analysis is 27 percent complete; finished in 13 minutes.
Surface analysis is 54 percent complete; finished in  8 minutes.
Surface analysis is 81 percent complete; finished in  3 minutes.

Surface analysis finished
xx bad blocks were found and remapped.

The Physical Disk cied(0,2) has been formatted.
Do you want to register it? [y] ♪
The Physical Disk cied(0,2) has been registered.
```

Performing All Physical Disk Formatting Steps

If you select option 6, "All of the above," **diskman** performs all five of the Physical Disk Formatting Menu options in order.

Managing a Logical Disk

Logical disks are formatted pieces or sections of physical disks. You might think of a logical disk as a virtual disk because it can have pieces on more than one physical disk, but it functions as a whole.

A logical disk name may be no more than 31 characters in length. Legal characters are a-z, A-Z, 0-9, - (hyphen), _ (underscore), and . (period).

Name your logical disks according to a function or classification that fits your application, such as a logical disk named **tax_86** that contains tax records for 1986.

When you create a logical disk, you must create all pieces of that logical disk. This means it is important to plan ahead. For instance, if you intend to have a logical disk that spans three physical disks, when you create the logical disk, you must create each piece on a specified physical disk. For instance, you could create logical disk **tax_86** with pieces on disk 0, disk 1, and disk 3, only if you do it all at once at creation time.

You cannot add pieces after you've created a logical disk. If one piece is damaged or inaccessible, you must delete all remaining pieces before you can re-use the associated physical disk space.

The Logical Disk Management menu is displayed when you select option 2, "Logical Disk Management Menu," from the Diskman Main Menu:

```
        Logical Disk Management Menu

1. Create a Logical Disk
2. Delete a Logical Disk
3. Display Information about a Logical Disk
4. Copy a Logical Disk
5. Display Information about a Logical Disk Piece
6. Delete a Piece of a Damaged Logical Disk

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

Enter Choice:
```

Creating a Logical Disk

Purpose	To create logical disks.
diskman menu	Logical Disk Management Menu
Note	You must create all pieces of a logical disk when you create the logical disk. You cannot add pieces later.
References	**diskman**(1M)

If you have finished formatting your physical disk, you are ready to create one or more logical disks. If you have not formatted your physical disk, return to the Physical Disk Management Menu and select option 5, Format a Physical Disk.

To create a logical disk, select option 1. When you create a logical disk, the first part you create is piece 1. You have the option of creating up to seven more pieces, to a maximum of eight per logical piece. The **diskman** program considers a one-piece logical disk to be piece one of one.

```
Enter the Logical Disk name: disk ↵
```

You will be prompted for information on each piece (up to the maximum of 8 pieces) that will be part of this logical disk. Below, we create a logical disk that consists of one piece on physical disk 0.

```
Logical Disk Piece 1:

Enter Physical Disk specification in DG/UX common
format: cied(0,0) ↵

Do you want to display the layout of this Physical Disk? [n]
```

Respond with **y** if you want to see a table showing the sizes and starting addresses of the physical disk's internal data areas and logical disks. The layout information is useful because it shows the remaining areas of free space on the disk.

The next prompt asks at what address you want your new logical disk to begin. The default is the first available location on the disk.

```
Enter the Physical Disk Address of the starting block
of Logical Disk Piece 1: [location] ↵
```

Unless you are creating a logical disk for which there is an assigned default size (like **root** or **swap**), the default size for the logical disk is the maximum possible size, given the current layout of the physical disk.

```
Enter the size in blocks of Logical Disk Piece 1: [size] ♪
You have allocated n blocks so far for this Logical Disk.

Do you want to specify any more Logical Disk Pieces? [n] ♪
```

If you specify more pieces, **diskman** begins the create loop again. When you have finished creating pieces, you get either a successful or an unsuccessful report:

```
The Logical Disk "disk" has been created.
```

or

```
Could not create the Logical Disk "disk".
```

If the creation was successful, you will be asked if you want to make a file system on the new logical disk. Respond as follows:

```
Do you want to make a file system on this logical disk? [y] ♪
```

This procedure uses the **mkfs** program to make a file system. You have the option of specifying whatever **mkfs** options or arguments you want **diskman** to use when it makes the file system. See the **mkfs**(1M) manual page for more information. The procedure continues:

```
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 the Logical Disk "disk"...
```

When **diskman** has finished, it displays:

```
Made a File System on the Logical Disk "disk".
```

If **mkfs** fails, it returns an appropriate error message.

Deleting a Logical Disk

Purpose	To delete all pieces that make up a currently usable logical disk. All pieces of the disk must be intact to use this procedure.
diskman menu	Logical Disk Management Menu
References	**diskman**(1M)

To delete a logical disk, select option 2. As you organize and reorganize your disk usage over time, you'll probably find it necessary to delete a logical disk. Below, we'll delete a logical disk that is spread over three physical disks. Note that you can only delete logical disk **accts_86** if all pieces are intact. This means that all pieces of a logical disk must be on physical disks that are in service. If one of those physical disks has suffered a failure, then any logical disks associated with that disk cannot be deleted with this procedure. The section "Delete a Piece of a Damaged Logical Disk" handles the case of recovering physical disk space by deleting pieces of logical disks that are on damaged physical disks.

Below, let's assume that we want to delete a functional logical disk:

```
   Enter the Logical Disk name: accts_86 ↵

 The Logical Disk accts_86 consists of the following pieces:


Piece     Physical Disk       Starting Physical     Size in
                              Disk Address          Blocks

1          cied(0,0)               3565              2000
2          cied(0,2)               48876             1500
3          cied(0,3)               716               1000

   Do you want to delete the Logical Disk "accts_86"? [y] ↵

   The Logical Disk "accts_86" has been deleted.

   Press the NEWLINE key when ready to continue...
```

If you try to delete a logical disk when one of its pieces is on a damaged physical disk, the system displays:

```
Could not remove the Logical Disk "accts_86".
```

Displaying Information About a Logical Disk

Purpose	To get an overall view of all logical disks in service.
diskman menu	Logical Disk Management Menu
References	**diskman**(1M)

To display information about a logical disk, select option 3. This option lists everything that the system knows about a logical disk. If you intend to delete or change pieces of logical disks, you need to know where all the pieces are located and how much space, in blocks, is being used by each logical disk piece. In the following dialog, **comm** is an example logical disk.

```
Enter the Logical Disk name: comm ↵
```

After you give the name of the logical disk, the system responds as follows:

```
The Logical Disk `comm' consists of the following pieces:

Piece  Physical Disk          Starting Physical      Size in
                              Disk Address           Blocks

1.     cied@18(FFFFEE00,0)       376542              3000
2.     cied@18(FFFFEE00,1)       218576              2000

        Press the NEWLINE key when ready to continue...
```

Copying a Logical Disk

Purpose	To make a backup copy of a logical disk on another logical disk.
diskman menu	Logical Disk Management Menu
References	**diskman**(1M)

To copy a logical disk, select option 4. Both versions of **diskman** offer this option. Before you can copy a logical disk, you must create a logical disk of exactly the same size as the one that you intend to copy. In the following example dialog, the source logical disk is the logical disk you wish to duplicate. The destination logical disk the new logical disk created to hold the copy. The following sample dialog shows how you may copy a logical disk after selecting option 4.

```
Enter the Source Logical Disk Name: assets ↵

Enter the Destination Logical Disk Name: bkup_assets ↵

WARNING: this operation will DESTROY any data currently
on the Logical Disk "bkup_assets"

Do you want to continue? [y] ↵

The Logical Disk "assets" has been copied to the Logical
Disk "bkup_assets".

Press the NEWLINE key when ready to continue...
```

Displaying Information About a Logical Disk Piece

Purpose	To list the addresses and sizes for specified logical disk pieces.
diskman menu	Logical Disk Management Menu
References	**diskman**(1M)

If you select option 5, "Display Information About a Logical Disk Piece," the system displays information, one piece at a time, for the pieces making up a logical disk. This option is available in both versions of **diskman**.

Below, we seek information on piece number 2 of logical disk **comm**.

```
Enter the Logical Disk Name: comm ↵

Enter the Logical Disk Piece Number: 2 ↵

This Logical Disk Piece is on the physical disk cied(0,1).
Its starting Physical Disk Address is block 218576.
Its size is 2000 blocks.

Press the NEWLINE key when ready to continue.
```

Deleting a Piece of a Damaged Logical Disk

Purpose	To recover physical disk space associated with out-of-service logical disk pieces.
diskman menu	Logical Disk Management Menu
Caution	Use this procedure only for deleting pieces associated with damaged physical disks. If you try to delete a piece of a usable logical disk, you will make that entire disk inaccessible. If you wish to delete an entire usable logical disk, see the section, "Deleting a Logical Disk," earlier in the chapter.
References	**diskman**(1M)

You can delete a piece of a damaged logical disk by selecting option 6, "Delete a Piece of a Damaged Logical Disk." Whenever one piece of a logical disk becomes inaccessible, all other pieces become inaccessible, and thus the *entire* logical disk is inaccessible. To re-use physical disk space, you must delete all logical disk pieces.

Deleting only a portion of a damaged logical disk allows you to recover physical disk space so that you can reformat it and use it for other logical disks. This procedure is useful when you have a logical disk spanning two or more physical disks. For instance, say you have a two-piece logical disk named **trimble**. One piece of **trimble** is on unit 0 and the second piece is on unit 1. Suppose the head scratches unit 1, putting it out of service. You can no longer use **trimble** because one of its pieces is on damaged physical disk 1. All other space associated with **trimble** on any other physical disks is also unusable. If you want to use that space on physical disk 1, you must first recover the space by deleting the associated piece or pieces of **trimble**.

After selecting option 6, we delete the surviving piece of **trimble** on unit 0:

```
WARNING: this operation will DESTROY the contents of a
Logical Disk.  Use this ONLY to recover the space being
used by the surviving pieces of an already-damaged
Logical Disk.

Do you want to continue? [y] ↵

Enter the Physical Disk specification in DG/UX common
format: cied(0,0)

Enter the Logical Disk Name: trimble ↵

Enter the Logical Disk Piece Number: 0 ↵

This Logical Disk Piece is on the Physical Disk cied(0,0).
Its starting Physical Disk Address is block 218576.
```

```
Its size is 20000 blocks.

Do you want to delete this Logical Disk Piece? [y] ↵

Piece 1 of the Logical Disk "trimble" has been deleted.

Press the NEWLINE key when ready to continue...
```

Now we can re-use the vacant space on physical disk **cied(0,0)** and send physical disk **cied(0,1)** off for repairs.

Managing a File System

Use the selections from the File System Management Menu to create and check file systems. File systems are created via the **mkfs**(1M) program and checked via the **fsck**(1M) program.

The following menu is displayed when you select option 3 from the **diskman** Main Menu:

```
          File System Management Menu

  1. Make a File System
  2. Check a File System

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

Enter Choice:
```

Making a File System

Purpose	To make a file system on a logical disk.
diskman menu	File System Management Menu
Caution	This command erases all information on the logical disk.
References	**mkfs**(1M)

Select option 1, "Make a File System," to create a new, empty file system on a logical disk. After creating the file system, you will have to mount it on a directory using the **sysadm mountfsys** command before users can access it. We recommend that you use the logical disk name associated with the file system when you mount the file system on a directory. For instance, if you have a file system on logical disk **comm**, you should create a mount directory named **/comm**. Then you mount the file system as file system **/comm**.

Type **?** for HELP if you are unsure how to use this option. Refer to **mkfs**(1M) for more information.

```
Enter the Logical Disk Name: comm ♪

WARNING: this operation will DESTROY any data on the
Logical Disk "comm".
Do you want to continue? [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 the Logical Disk "disk"...

Made a File System on Logical Disk "disk".

Press the NEWLINE key when ready to continue...
```

Notice that above we typed only a logical disk name, no options. For most cases, this will suffice for creating your file systems. That is, file system characteristics such as inode density (potential number of files), region size, free space, etc., are automatically set by **mkfs**. If you have special file system requirements, see **mkfs**(1M) for details on changing defaults.

Checking a File System

Purpose	To run the **fsck** check and repair program on a file system.
diskman menu	File System Management Menu
References	**fsck**(1M), Appendix D

Select option 2, "Check a File System," to check a file system for inconsistencies. The **fsck** program checks blocks and file sizes, directory contents, connectivity, link counts and resource allocation, and disk allocation region information. The **fsck** program reports any inconsistencies within the file system; it is your option to fix or ignore them. For a detailed discussion of **fsck**, see Appendix D in this manual.

The following example demonstrates how to use this option.

```
Enter the Logical Disk Name: comm ↵

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: -p ↵
```

The example above runs **fsck** with the **-p** option on file system **/comm**. The **fsck** program will display messages about the success or failure of the check.

Command Line Options

You may execute some **diskman** functions from the shell. You can invoke stand- among **diskman** with command line options that perform functions without going through the menus, or you invoke it via **sysadm diskmgmt**. Here are some options you may find useful. We list the functions provided as command line options below. The examples use **sd(insc(),0)** as an example physical disk specification and **usr_opt_X11** as the example logical disk name. Note that because the physical disk specification includes parentheses, which are shell metacharacters, the examples show the specification enclosed in single quotes.

- Display all registered disks with the following:

  # diskman display_registered_disks ↵

- Register physical disk **sd(insc(),0)** as shown below:

  # diskman register_disk 'sd(insc(),0)' ↵

- Deregister physical disk **sd(insc(),0)** with the following:

  # diskman deregister_disk 'sd(insc(),0)' ↵

- Display information on logical disk **usr_opt_X11** as shown below:

  # diskman display_ld_info usr_opt_X11 ↵

For a complete list of **diskman** functions, use the following:

 # diskman −l ↵

End of Chapter

Chapter 8
File System Management

Operations involving creating file systems and logical disks are done on systems that have their own physical disks. Other operations, such as mounting, unmounting, and modifying file system tables are done on systems with or without physical disks.

This chapter gives brief explanations of logical disks, mounting file systems, dump cycles, and shows you how to manage your DG/UX file systems. You'll use **sysadm** to add, delete, mount, unmount, restore file systems, and to make backup tapes. To create file systems, use **sysadm diskmgmt**. The major sections of this chapter are:

- File System Terms
- File System Perspectives
- Making File Systems Accessible
- File System Management Procedures
- Expert File System Information

File System Terms

You may want to review the information on file systems and logical disks in Chapter 2 (particularly Phase One, Planning the Installation, which introduces logical disks and tells how they can help you on your system) and Chapter 7 (which discusses disk management procedures). Chapter 4 contains file system performance information. We use the following terms in this chapter:

/etc/fstab — The **fstab**(4) file describes local and remote file systems that you want accessible on the local system. File systems listed in **fstab** become accessible automatically at boot time. The **fstab** file contains information for commands that mount, unmount, dump, restore, and check file systems.

mount point directory — After you name a logical disk (for example, **thor**), you create a file system on it. Next you mount the file system on a directory named **/usr/thor**. From now on, the new file system carries the name of the directory on which it was mounted. In **sysadm**, mount directory name is the same as file system name.

mount — To attach a file system to a directory, making it accessible to users.

unmount	To detach a file system from a directory, making it inaccessible to users.
initialize	To make a file system on a logical disk. Initialization clears or erases anything in the specified disk region. You can initialize with **diskman** or the **mkfs**(1M) command.
dump cycle list	A plan for doing daily, weekly, and monthly backups on tape. A default dump cycle list is supplied with the DG/UX system.
pass number	A number from a file system's **fstab** entry that indicates the order in which the **fsck** program will check the file system for corrupted and damaged files.
read-write mode	Access permission that allows general use of a file system.
read-only mode	Access permission that allows only reading of a file system.
NFS	Network File System. A separate software package that allows you to access remote file systems as though the remote file systems existed on your system. For more information, see *Managing NFS® and its Facilities on the DG/UX™ System.*

File System Perspectives

As discussed in Chapter 7, the DG/UX system uses logical disks, a feature that provides more flexibility for organizing file systems. The operating system treats a logical disk as if it were a real physical disk.

You can view file systems in two ways. From the operating system's perspective, a file system is associated with a logical disk, which in turn is associated with sections of physical disks accessed through a device node. From the user's perspective, a file system is a hierarchical directory structure. Logical disk names assigned by a user are incorporated into the node names used by the operating system.

The kernel creates device nodes at boot time. These device nodes provide the operating system with access to logical disks. For each device and for each logical disk, the kernel creates a device node automatically each time you boot the system.

The Operating System's View of File Systems

From the operating system's point of view, a file system is associated with sections of one or more physical disks. Logical disks form the bridge between file systems and physical disks. The file system associated with the logical disk is mounted (made available to users) in the directory structure.

You can think of the relationship between file systems, logical disks, and physical disks as a three-level hierarchy.

- **The file system** is the level at which the user interacts with the system.
- **The logical disk** is the intermediate level that associates the file system with the physical disk. The file system and the rest of the operating system interact at this level.
- **The physical disk** is the level at which the operating system interacts with the hardware.

Logical disks have other functions besides acting as bridges between physical disks and file systems. You received a preloaded **swap** logical disk or you created one during installation. Unlike other logical disks, the **swap** logical disk does not have a file system on it. The **swap** logical disk is an area of a disk that is accessed directly only by the operating system.

The User's View of File Systems

From a user's viewpoint, there appears to be one file system: the single hierarchy consisting of all files and directories on the system. Because all file systems are mounted under the **/** (root) file system, a system's entire directory tree appears to be a single hierarchical directory structure. If you mount a new file system, the directory tree (as seen by the user) expands, starting at the point where the new file system is mounted. The user sees a group of new files under a new directory name. If you unmount a file system, part of the directory tree disappears.

The operating system prevents you from inadvertently unmounting a file system that someone is using. For example, if a user changed directory to **/usr/opt/X11/catman**, you would not be able to unmount the **/usr/opt/X11** file system. Similarly, if a user were executing a program in **/usr/opt/X11/bin** (regardless of the user's working directory), you would not be able to unmount **/usr/opt/X11**.

Creating a File System

You can access **diskman** to create file systems in two ways. One way involves invoking stand-alone **diskman** from the SCM by typing

```
SCM> b cied()usr:/stand/diskman ↵
```

To create file systems after the operating system is running, use the stand-among version of **diskman** by invoking **sysadm diskmgmt**:

```
# sysadm diskmgmt ↵
```

Making File Systems Accessible

Once you have created a file system with **diskman**, you make the file system accessible to users by attaching or *mounting* it on a directory of your choice called a *mount point directory*, or just a *mount directory*. The mount directory then becomes the name of the file system. It is good practive to give the mount directory the same name as the logical disk where the file system resides. For example, for a logical disk named **thor**, we have mount directory (and file system name) **/usr/thor**.

To make a file system accessible, use **sysadm addfsys**. The **addfsys** command puts an entry for the file system in the **/etc/fstab** file. When the system changes to certain run levels, the system mounts file systems with entries in **fstab**. You may mount a file system yourself at any time with **sysadm mountfsys** or the **mount**(1M) command.

Every time you boot your system, the kernel creates new device nodes; therefore, in our example above, the node name **/dev/dsk/thor** will be created. Figure 8-1 shows mount directory **thor** branching from **/usr**. We now have a new file system named **/usr/thor**.

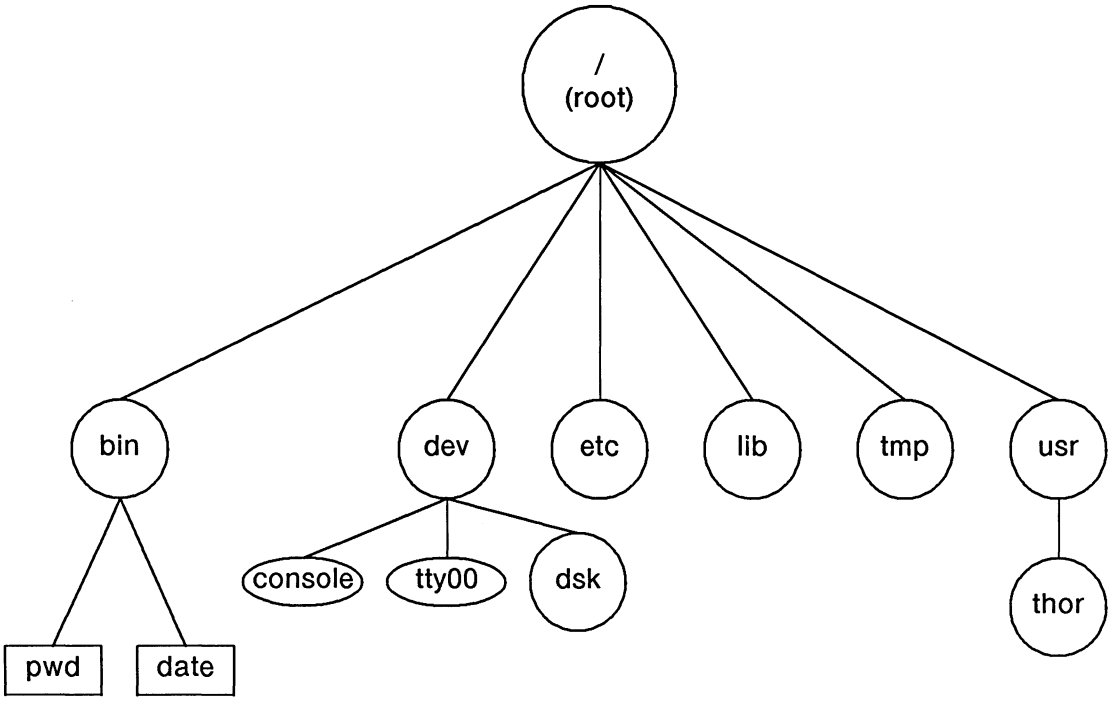


Figure 8-1 *Mounting a File System*

You can think of each file system as an independent directory tree. The **mount** command "grafts" a file system tree onto a larger tree. The top of the total tree is the **/** (root) directory, which is also the top directory of the root file system.

So, after you have used **diskman** to create a logical disk and make a file system on the logical disk, you must create a mount directory, and mount the file system on that mount directory. You can do this by selecting 1, **addfsys**, from the File System Management menu. When you conclude with **addfsys**, the file system will be accessible to users.

As a part of routine management, you or someone else will make backup tapes of your file systems. We have included a default dump cycle method designed around daily, weekly, and monthly backups. With this method, you schedule according to the dump cycle list. The **sysadm** utility's **fsdump** command reads the information from this list and interacts with you to perform each dump. See the section "Changing the Dump Cycle List" to see how to change the dump cycle to fit your needs.

File System Management Procedures

This section describes the procedures you will use to manage your file systems. When you select **fsmgmt** from the **sysadm** Main Menu, the following is displayed:

```
                    File System Management

 1 addfsys          Add an entry to the list of file systems
 2 delfsys          Delete an entry from the list of file systems
 3 lsfsys           List the available file systems
 4 modfsys          Modify an entry in the list of file systems
 5 addswap          Add a swap entry to the list of file systems
 6 delswap          Delete a swap entry from the list of file systems
 7 mountfsys        Mount a file system
 8 unmountfsys      Unmount a file system
 9 modcycle         Modify the current position in the dump cycle list
10 fsdump           Make backup tapes using dump
11 fsrestore        Restore a complete file system from fsdump tapes
12 filerestore      Extract a few files from fsdump tapes

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Adding a File System Entry

Purpose	To add an entry to the table of file systems in **/etc/fstab**. Also creates mount directory and mounts the file system if desired.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**addfsys**
Note	The **sysadm** program knows only about file systems that are listed in **/etc/fstab**.
References	**fstab**(4)

If you choose 1, **addfsys**, you will need to supply the name of the mount directory and name of the logical disk associated with the file system. If these are already in **/etc/fstab**, an error message is printed, and **addfsys** exits. If not, then you will be queried for the file system type, the read-write mode, the NFS mount mode (hard or soft), the dump cycle, and the **fsck** pass number. An example **addfsys** dialog follows:

```
Mount directory name? /usr/thor ♪

Is this a local file system? [yes]
```

At this point, we will split the **addfsys** dialogue into two examples: one for local file systems, and another for remote file systems, those accessed via NFS.

Adding Local File Systems

If you answer yes to the last query, the dialog with the system continues as follows:

```
Is this a local file system? [yes] ♪

Logical disk name? thor ♪

Writeable? [yes] ♪
```

Answering **yes** to the `Writeable?` query allows users to create or modify files on this file system. If you answer **no**, users will have only read and execute permissions for this file system, including all the files and directories in it.

Next, you are queried about the dump cycle you want for the file system:

```
Dump Cycle? [d] ↵
```

The dump cycle determines how often this file system should be backed up on tape, or archived. Indicating a dump cycle choice does not mean that archives (making backup tapes) will occur automatically. The information from this query is collected and used later by **fsdump** when you are ready to do an archive. Valid responses to this query are as follows:

dwm Archive daily, weekly, and monthly.

wm Archive weekly and monthly.

d Archive daily only.

w Archive weekly only.

m Archive monthly only.

x Do not archive at all.

See the section, "Changing the Dump Cycle List," for more information on the use of **d**, **w**, and **m**. The **addfsys** example dialog continues like this:

```
fsck Pass Number? [1] ↵
```

The **fsck** pass number indicates the pass on which an **fsck −p** process should check this file system for damaged or corrupted files. This number is a digit between 0 and 9. File systems with pass numbers between (and including) 1 and 9 will be checked in order. That is, all file systems with number 1 are checked first, then those with number 2, and so on. The digit 0 indicates that a file system should never be checked. For more information on **fsck**, see Appendix D.

The dialog continues:

```
Export? [no] ↵
```

Exporting a file system means that other systems on your network can mount it. If you do not want users on other systems to have access to the file system, or if you do not think that the file system is of interest to other users, take the default response to the `Export?` query; otherwise, answer **yes**. At this point, the system responds:

```
The entry for /usr/thor has been added.
```

If the directory you specified does not already exist, you may create it:

```
The directory /usr/thor does not exist.
Create /usr/thor? [yes]
```

You may then create the directory. If you do, **addfsys** continues:

```
Mount the file system? [yes] ♪

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

Adding Remote File Systems

Let's assume we have NFS and want to add a remote file system. The remote system is **sys5**. The mount directory is **/comm/prog**. We got this information from the system administrator of **sys5**. The following example shows how the **addfsys** dialog would differ if we were adding a remote file system.

```
Is this a local file system? [yes] n ♪

Remote host name? sys5 ♪
Remote mount directory? /comm/prog ♪
Writeable? [yes] ♪
Hard Mount? [yes] ♪

The entry for /comm/prog has been added.
```

If an NFS file system is hard mounted, user processes will wait indefinitely for file system accesses to be completed. This means that if the remote system on which the file system resides is not responding (because it is down, for example), user programs will appear to hang. The benefit of the hard mount option is that it provides more reliability when writing to a remote file system. If an NFS file system is soft mounted, user processes will receive an I/O error if the remote file system does not respond. The benefit of a soft mount is that the user process does not wait for the remote file system to complete the requested disk I/O. Answering **no** to the `Hard Mount?` query gives a soft mount.

If the directory you specified as a mount directory does not exist, the system displays:

```
The directory /comm/prog does not exist.
Create /comm/prog? [yes]
```

You can create the directory now if needed by accepting the default value. If you type **no**, **addfsys** exits without creating the directory or mounting the file system. The dialog continues as follows:

```
Mount the file system? [yes] ♪

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

Users cannot access the file system unless it is mounted. If you do not mount the file system now, it will be automatically mounted when you reboot.

Deleting a File System

Purpose	To delete an entry from the table of available file systems in **/etc/fstab**.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**delfsys**
Note	You may delete only one file system at a time.
References	**fstab**(4)

Select option 2, **delfsys**, to delete a file system. If you simply want to disable a file system, select option 4, **modfsys**, instead. The **delfsys** command requires that you specify the mount directory name, then it deletes the entry in **/etc/fstab** for the specified file system. With the entry deleted, that file system can no longer be mounted, and therefore is no longer accessible to users. All information is still on disk, so if you re-add the entry, it becomes accessible again.

An example **delfsys** dialog follows:

```
Mount directory name? /usr/thor ↵

Do you really wish to delete this file system entry? [no] y ↵

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

Listing File Systems

Purpose	To list the entries in **/etc/fstab**.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**lsfsys**
References	**fstab**(4)

For each file system in **/etc/fstab**, this command shows the logical disk name, the mount point directory, the file system type, the access mode, the NFS mount type, the dump cycle, and the **fsck** pass number. If you select 3, **lsfsys**, from the File System Management menu, the system responds as follows:

```
Logical             Mount       FS          NFS    Dump   Fsck
Disk                Directory   Type    RW  Mount  Cycle  Pass
---------------     ----------  -----   --  -----  -----  -----
/dev/dsk/root       /           local   rw           w      1
/dev/dsk/swap       swap_area   swap                 x      0
/dev/dsk/dbase      /dbase      local   rw           d      3
/dev/dsk/thor       /usr/thor   local   rw           d      2
sys5:/dev/dsk/sys   /comm/prog  remote  rw  soft     x      0

        Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

The discussion of the **addfsys** command, earlier in the chapter, explains the fields.

Modifying a File System Entry

Purpose	To modify an entry in the **/etc/fstab** file system table. Change logical disk name, file system type, access mode, dump cycle, and **fsck** pass number.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**modfsys**
Note	A file system must be unmounted before you can modify it.
References	**fstab**(4)

Option 4 of the File System Management menu, **modfsys**, locates the specified file system entry and uses its values as defaults for the queries below. If no entry exists, **modfsys** exits with an error message. Your interaction with **modfsys** proceeds as in the following example:

```
Mount directory name? /systems

Is this a local file system? [yes] ♪
```

Above, we accepted the default, **yes**, because the file system is local, that is, it exists on a physical disk connected to our processor. For a remote file system accessed via NFS, enter **n** for no. At this point, we will split the **modfsys** dialogue into two examples: one for local file systems, and another for remote file systems.

Modifying Local File Systems

We'll begin with the logical disk name:

```
Is this a local file system? [yes] ♪

Logical Disk Name? [systems] ♪

Writeable? [yes] n
```

Above, we changed the permissions on **/systems** to be "read only" by entering **n** after the `writeable?` query. After the `writeable?` query, continue to press New Line to select the defaults. Notice that we are exporting this local file system, which makes it available to be mounted and used by other systems.

The dialog continues:

```
Dump cycle? [m] ↵
fsck Pass Number? [1] ↵
Export? [yes] ↵

The entry for /systems has been modified.

Mount the file system? [yes] ↵

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

If you do not mount the file system, no one can access it until you mount it or until you take your system down and then bring it up to the proper run level with the **init**(1M) command. Bringing your system up to run level 1 or higher mounts local file systems, and bringing your system up to run level 3 mounts remote file systems. Chapter 3, Operating the DG/UX System, discusses run levels.

Modifying Remote File Systems

The **modfsys** command asks you if the file system is local. If the file system **/comm/prog** resides on remote system **sys5**, the example **modfsys** dialog continues like this:

```
Is this a local file system? [yes] n ↵

Remote host name? sys5 ↵
Remote Mount Directory? /comm/prog ↵
Writeable? [yes] ↵
Hard Mount? [yes] ↵

The entry for sys5:/comm/prog has been modified.

Mount the file system? [yes] ↵

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

In this example, we changed the file system **/comm/prog** so it is hard mounted instead of soft mounted.

Adding a Swap Entry to /etc/fstab

Purpose	To add swap entries to the **/etc/fstab** file.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**addswap**
References	**swapon**(1M)

Select option 5, **addswap**, to add an entry for a swap logical disk to your **fstab** file. A swap entry has the form:

```
/dev/dsk/swap   swap   swap   sw  x  0
```

The name of the logical disk is user-defined (the first field). The other fields are set by **sysadm**. Swap entries are used by **swapon**(1M) and by the accounting system. These apply only to local logical disks or NFS-mounted swap files.

Below, we'll add **comm** with **addswap**. We begin by typing:

```
# sysadm addswap ♪
```

The interaction continues as follows:

```
Is this a local swap area? y ♪

Logical Disk Name? comm ♪

The entry for /dev/dsk/comm has been added.
```

Deleting a Swap Entry from /etc/fstab

Purpose	To delete swap entries from the **/etc/fstab** file.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**delswap**
References	**swapon**(1M)

Select option 5, **delswap**, to remove a swap area entry from your **/etc/fstab** file. If the file system has been enabled for swapping by the **swapon** command, the file system will be used until the system is shut down. The DG/UX system does not permit a swap area to be removed from activity. Below, we'll delete **comm**. We begin by typing:

```
# sysadm delswap ♪
```

The interaction continues as follows:

```
Is this a local swap area? y ♪

Logical Disk Name? comm ♪

Swap entry for comm has been deleted.
```

Mounting a File System

Purpose	To mount a file system on a directory so that users may access the file system.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**mountfsys**
References	**fstab**(4), **mount**(1M)

Select option 7, **mountfsys**, to mount a file system that already has an entry in your **/etc/fstab** file. You will normally mount file systems for the first time when you add them with the **addfsys** command. File systems are automatically mounted when the system changes to run levels 2 or 3.

In this procedure, you must supply the mount directory name. That name must exist in **/etc/fstab**. If it does not, exit with **q** and use **addfsys** to make the entry. (For more information, refer to the section "Adding a File System Entry" earlier in this chapter. When you select **mountfsys**, the dialog continues as follows:

```
Mount directory Name? /usr/thor ↵
```

You may respond with a directory name or one of the following:

list Lists all mount directory names.

all Mounts all file systems.

remote Mounts all remote file systems.

local Mounts all local file systems.

If the mount directory does not exist, you may create it now:

```
Create /usr/thor? [yes]
```

If you respond with **no**, **mountfsys** exits without mounting the file system. If you answer **yes** (the default) and the command succeeds, you see this:

```
File system /usr/thor has been mounted.

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

Unmounting a File System

Purpose	To unmount a file system from a directory, making it inaccessible to users.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**unmountfsys**
References	**fstab**(4), **mount**(1M)

Select option 8, **unmountfsys**, to unmount a file system. The example below unmounts a specific file system. Another way to use this command is by responding to the first query, `Mount directory Name?`, with one of the following:

list Lists names of all mount directories (whether currently mounted or not).

all Unmounts all file systems.

remote Unmounts all remote file systems.

local Unmounts all local file systems.

Below, we'll unmount a file system called **/accounts** (which is also the name of the file system's mount directory).

When you select **unmountfsys**, the system responds as follows:

```
Mount directory Name? /accounts ↵
```

If the file system is not mounted, the following is displayed:

```
File system /accounts is not mounted.
```

If the unmount succeeds, the following is displayed:

```
File system /accounts has been unmounted.
```

Or, if the unmount fails:

```
An error has occurred using the unmount command.  The error
message is: error_message

Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

Modifying the Dump Cycle

Purpose	To modify the current position in the dump cycle.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**modcycle**
References	**dump2**(1M)

Select option 9, **modcycle**, to change the current position of the pointer in the dump cycle list. This list contains one entry for each day of the month. The pointer indicates the current day's entry to **fsdump**, which performs the dump. Each entry shows the dump cycle letters that match the dump for the day (this selects the file systems to be dumped) and the tape label information to be used for the day's dump tapes. When a dump is completed, the pointer is automatically moved to the next day's entry. On the first day of the month, the pointer is reset to the top of the list to restart the dump cycle.

It's possible for the current pointer position in the list to be wrong if dumps are skipped for a day or more. With **modcycle**, you can manually move the pointer to the desired line in the list.

When you select the **modcycle** command, you enter into the following dialog:

```
Dump Cycle List Operation? [list] ♪
```

Press New Line to select the default, **list**. Possible responses to this query are as follows:

list Print the dump cycle list.

forward Move the dump cycle pointer forward.

backward Move the dump cycle pointer backward.

reset Move the pointer to the top of the list.

end Stop editing the dump cycle list.

When you select the default **list** option, the system displays the following list:

```
Dump Cycle List

Cycle    Level   Multi        Label
-----    -----   -----  --------------------
dwm        0       n      Monthly
--> d      6       y      Week 1 - Monday Set
d          7       y      Week 1 - Tuesday Set
d          8       y      Week 1 - Wednesday Set
d          9       y      Week 1 - Thursday Set
dw         1       y      Week 1 - Weekly Set
d          6       y      Week 2 - Monday Set
d          7       y      Week 2 - Tuesday Set
d          8       y      Week 2 - Wednesday Set
d          9       y      Week 2 - Thursday Set
dw         2       y      Week 2 - Weekly Set
d          6       y      Week 3 - Monday Set
.          .       y        .             .
.          .       y        .             .
.          .       y        .             .
dw         5       y      Week 5 - Weekly Set


The current position is marked by -->.
```

The entries in the list have this significance:

cycle Lists the cycle letters that correspond to those in **/etc/fstab**, which indicate when the file systems will be dumped. For instance, file system **/comm** might be set to **w**, so it is only dumped once per week. You set the cycle letter in **/etc/fstab** when you add the file system with **addfsys**. Here, again, are the cycle letters and there meanings:

dwm Archive daily, weekly, and monthly.

wm Archive weekly and monthly.

d Archive daily only.

w Archive weekly only.

m Archive monthly only.

x Do not archive at all.

Level These numbers are used internally by the **fsdump** program. The ones we supply need not be changed for normal system operation.

Multi Multi-dumps. **fsdump** normally dumps one file system per tape. If you specify **y**, multi-dumping occurs. This means as many file systems as there is room for will be written to tape. An **n** entry means write just one file system per tape, as is shown for the monthly line below.

Label We recommend that you label your tapes so that they match the entries in the dump cycle list. `Monthly` means dump all file systems marked with **d**, **w**, or **m** (daily, weekly, or monthly). `Monday Set` means dump all file systems marked with **d** (daily), and so on with the other weekdays. Friday's dump becomes part of the `Weekly Set` of dump tapes.

If you specify the dump cycle list operation **forward**, the system responds:

```
How far to move forward? [1] ↵
```

Take the default or enter the number of positions you wish to move forward. This number must be greater than 0. If your number moves the pointer off the list, the pointer is positioned at the top of the list again.

```
    Dump Cycle List

    Cycle   Level  Multi        Label
    -----   -----  -----  --------------------
    dwm       0      n     Monthly
    d         6      y     Week 1 - Monday Set
--> d         7      y     Week 1 - Tuesday Set
    d         8      y     Week 1 - Wednesday Set
    d         9      y     Week 1 - Thursday Set
    dw        1      y     Week 1 - Weekly Set
    d         6      y     Week 2 - Monday Set
    d         7      y     Week 2 - Tuesday Set
    d         8      y     Week 2 - Wednesday Set
    d         9      y     Week 2 - Thursday Set
    dw        2      y     Week 2 - Weekly Set
    d         6      y     Week 3 - Monday Set
    .         .      y       .            .
    .         .      y       .            .
    .         .      y       .            .
    dw        5      y     Week 5 - Weekly Set


    The current position is marked by -->.

    Dump Cycle List Operation? [list] ↵

    Press the NEWLINE key to see the fsmgmt menu [?, ^, q]:
```

The **reset** choice displays the top entry in the dump cycle list.

Making File System Backup Tapes

Purpose	To dump file systems to tape as indicated by the dump cycle list described in the section, "Modifying the Dump Cycle."
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**fsdump**
References	**dump2**(1M), **dumptab**(4), **dump2label**(1M)

Select option 10, **fsdump**, to create dump tapes of file systems. The **fsdump** command uses the dump cycle list and determines the correct entry for the current day. Then, it looks through the file system table (**/etc/fstab**) and locates those active file systems that match the current cycle entry; when a match is found, the file system is scheduled to be dumped. The **fsdump** command calls the **dump2**(1M) program for each file system that is scheduled to be dumped.

When you select **fsdump**, you enter the following dialog:

```
Tape Drive? [0] ↵

Tape Medium? [default] ↵
```

The default tape medium is a QIC-150 cartridge tape. Note that we accepted the defaults in both cases above.

Notice that the multi-dump field of the monthly line is set to **n**. This is to ensure that the monthly dumps are separated in case of loss or damage to the tape. Remember, the **n** setting means that **fsdump** will list each file system that is scheduled to be dumped, one at a time, and give you the option of dumping it or not dumping it. When the multi-dump field is set to **y**, then all file systems would be listed.

Assume that **/usr/opt** is the first file system to be dumped and the dump cycle pointer is on `Monthly`.

```
Label the next tape set:

Monthly Set
/usr/opt
08/01/90
Tape #1, #2, ...
```

Then **fsdump** prompts you to mount the first tape of the set:

```
Mount the first tape of the set and place the tape drive online.
Enter yes when the tape is ready.
Is the tape ready? yes ↵
```

Possible responses to the last query are **yes**, **no**, or **skip**. You should skip a file system only if there is an error that makes it impossible to back it up or if the system crashed while **fsdump** was running. For instance, if you are dumping five file systems and the system crashes while you're on the third one, you would skip numbers one and two and resume dumping with the third file system.

When you answer yes, the dump will begin:

```
Dumping: /usr/opt  Monthly Set  08/01/90
```

When you are finished dumping, the pointer on the dump cycle list advances.

Restoring File Systems from fsdump Tables

Purpose	To restore file systems to disk from backup tapes that were made with the **fsdump**(1M) command.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**fsrestore**
Note	The directory in which the restore is to take place must be empty.
References	**dump2**(1M), **restore**(1M)

Use the **fsrestore** command, option 11, when you want to load files or file systems from tape to a directory on disk. You might want to do this when you're recovering from a disk failure or moving an entire file system from one disk to another. You are asked to supply the directory name where the restore will be done. A full restore works by restoring a full (monthly) dump onto an empty file system. Then, incremental (weekly and daily) dumps are restored on top of that until the most recent tape set has been loaded. You should locate the following tapes associated with the desired file system:

- The most recent monthly dump.
- All weekly dumps since the most recent monthly dump.
- All daily dumps since the most recent weekly dump.

When **fsrestore** asks you for tape sets, you should restore them in the order shown above (a monthly followed by weeklies followed by dailies). You will be asked to mount the first tape of a set and then execute the restore program which will ask for further tapes from the set. This process will be repeated until you indicate that there are no more tapes.

Mount the tape and select **fsrestore**; the system responds as follows:

```
Tape Drive? [0]
```

Press the New Line key to accept the default.

In the following interaction you are asked for the name of the empty directory to which you will restore the file system and whether the first tape of the set is ready:

```
Mount Directory Name? /foobar ↵

Is the first tape of the set ready? y ↵
```

A **yes** answer to the second query means that you have mounted the tape and the drive is online. The restore process begins.

If **/foobar** is not mounted, you will be instructed to use **mountfsys** to mount it. If **/foobar** is not empty, you will be instructed either to choose another directory or to use **makefsys** to clear the file system.

Next, the available file systems on the tape are displayed, then you are queried about the file system you typed in:

```
The file systems on this tape are:

/foo
/foobar
/usr/newdir

File Systems from Tape? [/foobar]  ↵
```

If there are problems, an error message will be printed, otherwise you will be asked for another tape:

```
Is there another tape set? [yes] ↵
```

Pressing the New Line key enters the **yes** default and the restore process continues. When you have loaded all tape sets, answer **no** when you are asked for another tape set, and you will exit from **fsrestore**.

There will probably be times when there is only one file system on the tape and you want to write that file system to another location other than its current path name. For instance, you might want to put **/foo** (which is on the tape) into **/new_location/newfoo**. The system would respond as follows:

```
Mount Directory Name? /new_location/newfoo ↵

The file system on the tape /foo does not match the
mount directory /new_location/newfoo.

Restore this tape set? [no] y ↵
```

This interaction writes the file system on the tape to the new location.

Extracting Individual Files from fsdump Tapes

Purpose	To restore individual files that have been lost or deleted.
Starting Conditions	administrative or multiuser state
sysadm menu	fsmgmt
Commands	**filerestore**
Note	This command restores files that were previously dumped with **fsdump**.
References	**restore**(1M)

Use the **filerestore** command, option 12, for small restore tasks, such as when a user has accidentally deleted files or directories. For major file system recovery, such as after a disk failure, use **sysadm fsrestore**.

The **filerestore** command asks for the directory where the restored files should be placed. If you specify an invalid directory, the system asks if you want the directory to be created. If you don't create the directory, you will be asked to select another directory.

In the following example, we restore a file named **redeye** that resided in directory **/sales/accounts/smith**. We put the restored file under the **/tmp** directory. When you select **filerestore**, a dialog like the following occurs:

```
Tape Drive? [0] ↵

Restore directory? /tmp ↵

sysadm will now run the restore program in interactive mode
on repeated sets of dump tapes until you tell it to stop.
If you haven't run filerestore before, enter ? when the restore
prompt 'restore>' appears.

Please mount the first tape of the first tape set.
Is the first tape of the set ready? y ↵

The file systems on this tape are:

/sales/specs
/sales/accounts

File system from Tape?  /sales/accounts ↵
```

When **filerestore** is finished with an operation, it prompts you for another command. You can use the following commands at the `restore>` prompt:

ls List directory contents (**ls**(1) options are invalid).

cd Change directory.

pwd Print working directory.

add Add file name to the list of files to be extracted.

delete Delete file name from the list of files to be extracted.

extract Extract requested files.

quit Exit program.

help Print this list.

To restore the file **/sales/accounts/smith/redeye**, we use some of the commands from the list:

```
restore> cd /sales/accounts/smith ↵

restore> ls redeye
redeye

restore> add redeye

restore> ls redeye
*redeye

restore> extract

You have not read any tapes yet.  Unless you know which
volume your files are on, you should start with the last
volume and work towards the first.

Specify next volume #: 1 ↵
Set owner/mode for '.'? [yn] n ↵

restore> quit ↵
```

Above, we changed directory to the working directory, listed the file, added the file to the list to be extracted, and extracted the file. It was written to **/tmp** so that you can inspect the file before installing it in its original directory.

Making Tapes

Purpose	To move on-line data to a magnetic tape.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**cpio**(1), **find**(1)

This section does not discuss a **sysadm** menu procedure; it simply offers some suggestions for making tapes. When you have a small-scale dumping task, as when you are making a personal tape for a user, you don't need to use the **dump2** and **restore** operations. You can use **cpio**(1) instead. See the **cpio** manual page for a complete listing of options and further instructions. To dump a directory named **/sales/smitht** (and all subdirectories and files under it), do the following:

1. Mount a tape and put the drive on-line.

2. Go to the directory you wish to dump:

   # cd /sales/smitht ↵

3. Dump everything in the directory to tape:

   # find . −print | cpio −ocvB > /dev/rmt/0 ↵

   The contents of **/sales/smitht** have been dumped to tape. The **cpio** options we used are:

   o Copy files to standard output.

   c Use ASCII headers for portability.

   v Be verbose: print a list of file names.

   B Use large block size: 5120 bytes instead of 512.

   To write individual files to tape, go to the directory where the files are located and type:

   # echo fileA fileR fileZ | cpio −ocvB > /dev/rmt/0 ↵

   This command dumps to tape the contents of all three files.

   We directed the output of the dump to raw magnetic tape (**rmt**), device 0.

Expert File System Information

Information in this expert section is optional. You do not need to read this section to manage your file systems; however, you may find this information useful if you have to diagnose file system-related problems.

Mounting and Unmounting a File System Without sysadm

As we stated earlier, to access the file system on logical disk **thor**, it must be *mounted* on a directory. You can mount a file system from the command line with the **mount**(1M) command. The following command line shows the file system on logical disk **thor** being mounted on a directory called **/usr/thor**. Notice that the **mount** command identifies the logical disk by its node name.

```
# cd /usr ↵
# mkdir thor ↵
# mount /dev/dsk/thor /usr/thor ↵
```

Above, we created mount directory **/usr/thor** before we mounted the file system on it. You can do the same thing with the **mountfsys** command in this chapter. To unmount a file system, you can use the **umount**(1M) command.

Dumping and Restoring Without sysadm

Refer to the **dump2**(1M) and **restore**(1M) manual pages for detailed information.

Using dump2(1M)

The **dump2** program copies some or all files on a logical disk to the backup medium based on the dump "level". There are 10 levels: 0 through 9. Execute the **dump2** program by specifying a logical disk and a dump level as in the following:

```
# /usr/sbin/dump2 −0 −u −f /dev/rmt/0 /dev/dsk/root ↵
```

The above command line uses the following options:

0 Dump level.

u Update the **/etc/dumpdates** file.

f Dump file name.

The **dump2**(1M) manual page lists all available options.

The dump level number instructs **dump2** to make a copy of each file that has been modified since the most recent dump at any lower dump level number. For example, if the dump level is 3, **dump2** will make a copy of any file that has been modified

since the most recent level 0, 1, or 2 dumps. Level 0 dumps *every* file in the file system because there is no lower dump level. A level 0 dump is often called a "full" dump.

The **dump2** command knows that a file has been modified by examining the inode change time (or "ctime") and the file modification time (or "mtime") for each file (see **stat**(2) for detailed information). If either of these is later than the dump time for the file system at the appropriate dump levels, then the file has been "modified" since the previous lower level dumps. The **dump2** command knows when the file system was last dumped at any given level because it keeps this information in the file **/etc/dumpdates**. This file contains lines of the form:

```
/dev/dsk/root          3 Fri Jan  6 15:45:15 1989
```

In this case, the most recent level 3 dump for **/dev/dsk/root** was made at 3:45 p.m. on January 6, 1989. An entry is added to **/etc/dumpdates** only after the dump completes the successfully. This prevents it from inserting a date for a dump that later aborts. Also, duplicate entries are deleted. In the example above, any other level 3 entries for **/dev/dsk/root** would be deleted when adding the new one.

Making Backup Tapes

To make backup tapes, you must first set up a dump schedule and determine which dump level should be used on each dump to implement that schedule. For example, the default dump cycle list shipped with **sysadm** is based on the following dump schedule:

- A "monthly" dump is performed once a month (probably on the weekend). This records *every* file on the file system.
- Each Friday a weekly dump is performed. This records every file modified since the most recent weekly or monthly dump.
- On Monday, Tuesday, Wednesday, and Thursday, a daily dump is performed. This records every file modified since the most recent daily, weekly, or monthly dump.

These dumps would generally be done after prime working hours (say at 6 p.m.).

Dump levels that would implement this schedule are shown below.

DUMP	LEVEL
Monthly	0
Weekly 1	1
Weekly 2	2

Weekly 3	3
Weekly 4	4
Weekly 5	5
Monday	6
Tuesday	7
Wednesday	8
Thursday	9

The first weekly dump is the one following the monthly dump. This schedule would be carried out (say, for the root file system) by performing dumps in the following order:

DUMP	COMMANDS
Monthly	**/usr/sbin/dump2 −0uf /dev/rmt/0 /dev/dsk/root**
Monday (1)	**/usr/sbin/dump2 −6uf /dev/rmt/0 /dev/dsk/root**
Tuesday (1)	**/usr/sbin/dump2 −7uf /dev/rmt/0 /dev/dsk/root**
Wednesday (1)	**/usr/sbin/dump2 −8uf /dev/rmt/0 /dev/dsk/root**
Thursday (1)	**/usr/sbin/dump2 −9uf /dev/rmt/0 /dev/dsk/root**
Weekly (1)	**/usr/sbin/dump2 −1uf /dev/rmt/0 /dev/dsk/root**
Monday (2)	**/usr/sbin/dump2 −6uf /dev/rmt/0 /dev/dsk/root**
Tuesday (2)	**/usr/sbin/dump2 −7uf /dev/rmt/0 /dev/dsk/root**
Wednesday (2)	**/usr/sbin/dump2 −8uf /dev/rmt/0 /dev/dsk/root**
Thursday (2)	**/usr/sbin/dump2 −9uf /dev/rmt/0 /dev/dsk/root**
Weekly (2)	**/usr/sbin/dump2 −2uf /dev/rmt/0 /dev/dsk/root**

and so on.

Week 5 will not always be needed, depending on the month.

Changing the Dump Cycle List

You should read the previous section to understand how **dump2**(1M) works. The dump cycle list is kept in the file **/etc/sysadm/dumpcycle**. It contains lines of the following form:

cycle_pattern dumplevel multiflag tape_label

where the fields are separated by tabs.

The first field, *cycle_pattern*, is a **grep**(1) pattern that selects one or more letters from the dump cycle field in **/etc/fstab**. The permissible letters are as follows:

dwm Archive daily, weekly, and monthly.

wm Archive weekly and monthly.

d Archive daily only.

w Archive weekly only.

m Archive monthly only.

x Do not archive at all.

The second field, *dump_level*, is a dump level number, which is a single digit between 0 and 9. The third field indicates whether or not all file systems should be written onto the same tape. The last field is an arbitrary text string which is given to the operator as the desired tape label for this dump.

One line in the **dumpcycle** file is marked as the "current" dump by placing an "at" sign (**@**) in front of it:

```
@[dwm]   0   n    Monthly Set
```

When **sysadm fsdump** is used, it locates the current entry. It then locates every line in **/etc/fstab** whose dump cycle field matches `cycle_pattern`. In this example, the pattern `[dwm]` would match daily, weekly, and monthly entries. The **sysadm fsdump** program then executes the **dump2**(1M) command once for each matching line in **/etc/fstab**. The dump level is taken from the second field in the dump cycle and the tape label given to the operator is taken from the last field.

The **dumpcycle** file (and thus **fsdump**) can be set up to use any dump schedule. For example, if a site wishes to do a full dump every Monday and do incremental daily dumps on Tuesday through Friday, the following **dumpcycle** file would work:

```
@[dwm]   0  n     Monday (Full) Set
d        1  n     Tuesday Set
d        2  n     Wednesday Set
d        3  n     Thursday Set
d        4  n     Friday Set
```

Using restore(1M)

Restore file systems using the **restore**(1M) command with the **r** option. If a file system is completely destroyed, it can be restored by first remaking the file system and then using the **restore** command on the following tape sets:

1. The most recent monthly dump.

2. All weekly dumps made since the most recent monthly dump.

3. All daily dumps made since the most recent weekly dump.

Consider an example environment where we do our weekly dumps on Friday. If the file system is lost on Wednesday of the second week (before the Wednesday dump), we need the following tapes:

Monthly
Weekly (1)
Weekly (2)
Monday (2)
Tuesday (2)

The following are sample command sequences:

Unmount the file system:

 # /etc/unmount /dev/dsk/foo ↵

Remake the file system (be aware that this command destroys all data on the logical disk):

 # /etc/mkfs /dev/dsk/foo ↵

Check the file system:

 # /etc/fsck /dev/dsk/foo ↵

Mount and restore the monthly tape set:

```
# /etc/mount /dev/dsk/foo /mount_name ↵
# cd /mount_name ↵
# /usr/sbin/restore rf /dev/rmt/0 ↵
```

Restore the weekly backups and daily backups, one at a time::

```
# /usr/sbin/restore rf /dev/rmt/0 ↵
```

The **restore r** command restores all files in the current directory.

End of Chapter

Chapter 9
File Information

This chapter shows you how to find and display information about files in your file systems.

File Terms

We use the following terms in this chapter:

inode — Data structure containing information about a file such as file type, size, date of creation, owner ID, and group ID. The number of inodes represents the total number of files that can exist on the system. The total number of inodes is set in the **diskman** program. An inode is 126 bytes long, and there are 4 inodes to a disk block.

disk block — A unit of data as it is actually stored and manipulated. It consists of 512 bytes.

setuid — A mode bit that can be specified for any executable file. When a user runs an executable file that has the **setuid** bit set, the system gives the user the permissions of the owner of the executable file for the duration of the command. See **chmod**(1).

/dev — Administrative directory containing entries for all devices on the system.

File Information Procedures

When you select **fileinfo** from the **sysadm** Main Menu, the following choices are displayed:

```
                    File Information

 1 diskuse          Show free blocks on mounted file systems
 2 fileage          Locate idle files in a directory tree
 3 filename         Locates files by name in a directory tree
 4 filescan         Locate files with possible permission errors
 5 filesize         Locate very large files in a directory tree

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, or q to QUIT:
```

Remember that help is always available for any of the commands in a **sysadm** menu. For example, you can get help on **diskuse** by typing **1?**. Also, type **?** for any query you don't understand how to answer.

Listing Disk Information

Purpose	To obtain usage information on mounted file systems.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**diskuse**
References	**df**(1M), **du**(1M)

Select option 1, **diskuse**, to display a table showing the number of blocks and inodes in use on each mounted file system. Here's an example of a display that **diskuse** provides:

```
                Free    Total    Pct      Free     Total    Pct
File System    Inodes  Inodes   Used    Blocks    Blocks   Used
-----------    ------  ------   ----   -------   -------   ----
/               3738    4094     8%      2204     18480    88%
/usr           19534   22526    14%     28012    152240    81%
/udd/sys5      17772   22526    21%     33100    152240    78%
/comm              0       0     0%    116926    584110    79%
/usr/local      1475    7650    98%      9614     88000    89%
/spare         20474   20474     0%    135272    140800     3%
/acme          46231    7500    38%     27433     58700    53%

Press the New Line key to see the fileinfo menu [?, ^, q]:
```

Searching Files by Age

Purpose	To list all files in a directory by specified age.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**fileage**
References	**ls**(1)

Select option 2, **fileage**, to search a specified directory and lists all files that have not been accessed within a given number of days. You will find it necessary to search your directories from time to time to see what files are taking up space, but receiving little use. For instance, you might want to remove a file that has not been accessed in the last 90 days. When you select **fileage**, the interaction continues as follows:

```
Search Directory?  /comm/smith ♪
```

You must enter a path name starting with **/**. The entire tree starting with **/** will be searched. There is no default directory. If there is a problem with your entry, you will get one of the following messages:

```
/comm/smith does not exist.
```

or

```
/comm/smith is not a directory.
```

Next, you are queried:

```
File age (in days)? [90]  ♪
```

You can enter any number of days or choose the default. All files that have not been accessed in the period you specify will be printed.

```
Owner       Size (bytes)     Last Access          File Name
-------     -----------      -----------       -----------------
smith        150227          May 14 1989      /comm/smith/a.out*
smith           400          Nov 03 1987      /comm/smith/emacs
smith          2000          Aug 21 1986      /comm/smith/nfs.1
smith        765555          Jun 11 1986      /comm/smith/tcpip
smith       9999999          Dec 29 1985      /comm/smith/xwindow

Press the New Line key to see the fileinfo menu [?, ^, q]:
```

Listing Files by Name

Purpose	To locate and list specific files in a directory.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**filename**
References	**ls**(1)

Select option 3, **filename**, to search a directory and list all files which have a given basename. If the desired name is "foo", **filename** will print the path names of files that match it, such as **/usr/foo** and **/usr/lib/foo**. This is useful when you need to locate a file in a directory tree. You can specify the wildcard search and locate all files beginning with the same string. Below, we are looking for all file names beginning with `lost.file` so we attach an asterisk (*) to the file name.

When you select the **filename** command, the system queries as follows:

```
Search Directory?  /comm ♪
File Name?  lost.file* ♪

Owner     Size (bytes)  Last Access          File Name
----      -----------   -----------     ----------------------
smith     155           Apr  2 1988     /comm/docs/lost.file1
smith     3410          May  9 1989     /comm/docs/lost.file2
smith     1550          May 10 1989     /comm/docs/lost.file3

Press the New Line key to see the fileinfo menu [?, ^, q]:
```

If you specify a file name that is not in a given directory, a message is printed telling you the file was not found.

Listing Files with Permission Errors

Purpose	To search for and list files in a directory that are owned by root and have the **setuid** bit set. To search for and list device files that should be in **/dev**.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**filescan**
References	**setuid**(2), **chmod**(1)

Select option 4, **filescan**, to search a directory tree for files that have suspicious ownership and permission settings. These files may indicate that an error or a security breach has occurred. This command searches the directory you specify and reports files that have the following problems:

- Device files that exist outside of **/dev**

  No device files should reside outside **/dev** unless you, as system administrator, have created or moved device files for a special purpose, such as a test.

- Non-system files that are owned by root and have the **setuid** bit set

The **setuid** bit is a special kind of permission attribute that all files have but which is only useful for executable files (like programs and shell scripts—not directories or text files). When a user runs an executable that has the **setuid** bit set, the process runs with the effective user ID of the executable's owner—not, as is normally the case, with the user ID of the person running the executable.

For example, if user **fred** executes a program owned by user **bob**, and this program *does not* have the **setuid** bit set, **fred**'s process runs with the effective user ID of **fred** (as normal). On the other hand, if user **fred** executes a program owned by user **bob**, and this program *does* have the **setuid** bit set, **fred**'s process runs with the effective user ID of **bob**. This means that the program, if it is so written, can access files to which **fred** may not normally have access, but to which **bob** does. User **fred** does not even need to know **bob**'s password for these accesses to occur.

The rationale behind the **setuid** bit is to allow users to perform some action that they should not be able to perform under normal circumstances. The **lp** command, for example, has the **setuid** bit set so that any user who invokes **lp** to queue up a print job can, through the agency of the **lp** program, do things like copy files to the LP system's directories and add requests to the LP scheduler's queue file.

When you execute a program that has the **setuid** bit set, your actions are determined by the scope of the program and the user ID of the program's owner. Thus, we arrive at the danger inherent in the **setuid** bit feature: the combination of a permissive program that has the **setuid** bit set, owned by a privileged user ID, may give a user too much freedom on the system. This situation can result in a breach of security. The extreme case would be a shell program owned by **root** that has its setuid bit set; any user could execute the program and enjoy the use of a superuser shell throughout the system.

Among its various functions, the **filescan** program includes a search for insecure programs, ones owned by **root** that have the **setuid** bit set.

The following example file listing shows **ls −l** output for several files that have the **setuid** bit set (which we know because of the **s** flag in the user-execute permission and/or owner-execute permission places in the permissions line). The file **/usr/tom/nasty** is suspicious because it is owned by **root** but is obviously not a normal system program. It appears to belong to user **tom**, who no doubt has evil on his mind. Depending on what Tom's program does, it may constitute a security breach.

```
-rwsr-sr-x  1 root  bin   120792 Mar 28 18:16 /usr/bin/at
-rwsr-sr-x  1 root  bin    98520 Mar 28 18:16 /usr/bin/crontab
---s--x---  1 root  users  95376 Aug 18 11:08 /usr/tom/nasty
```

On a secure system, Tom would never have been able to create such a program without superuser access. Thus, the program may not only constitute a security breach itself, but it also indicates the presence of a breach elsewhere, the one that allowed Tom (or some good friend) to log in as superuser and produce the suspicious executable.

To protect your system, never leave your logged-in terminal unattended (particularly if you are logged in as **root** or **sysadm**). Another user could move or copy files, or commit any manner of destructive acts, all with your user ID.

When you select **filescan**, an interaction like the following will occur:

```
Search Directory? /crock ↵

The following files look suspicious.
The problem codes are:

device  This device file was found outside of /dev.
setuid  This file is owned by root and has setuid mode on.


Problem          File Name
-------          --------------
setuid           /crock/sh
setuid           /crock/ps
device           /crock/tty14
```

When you find a **setuid** bit set, investigate further. You may need to correct **setuid** permissions with the **chmod**(1) command. In general, you will not be creating device files, so there shouldn't be any ouside of **/dev**. There might, however, be the case when you or someone else creates a *test* device file outside of **/dev**. If necessary, you may either move or delete the device file.

Listing Files by Size

Purpose	To search for and list files in a directory by specified size.
Starting Conditions	administrative or multiuser state
sysadm menu	fileinfo
Commands	**filesize**
References	**df**(1M), **ls**(1)

Use this option 5, the **filesize** command, mainly for cleanup purposes. If, for instance, you notice a drastic reduction in available disk space, you could check to see what large files might be using up the space. In a specified directory, the **filesize** command lists the largest files. You supply the directory name and the number of files you want displayed. The default number of files is 10, meaning that the 10 largest files will be displayed.

When you select the **filesize** command, you will be queried for information as shown in the following interaction:

```
Search Directory? /comm/smith ♪

How many files? [10] 3 ♪
```

We entered 3 to this query. The system displays the 3 largest files in the **/comm/smith** directory.

```
Owner    Size (bytes)      Last Access     File Name
-------  ------------      -----------     -----------------
smith          456000      Dec 29 1987     /comm/smith/tokyo
smith          200998      Oct 10 1988     /comm/smith/new_york
smith          100001      Sep 21 1988     /comm/smith/mayberry

Press the New Line key to see the File Information menu [?, ^, q]:
```

End of Chapter

Chapter 10
TTY Management

This chapter shows you how to set up and manage terminals (ttys) on the DG/UX system with **sysadm ttymgmt**. If you are not already familiar with **inittab**(4) and **gettydefs**(4), you may want to consult these manual pages as you use the information in this chapter.

The major sections of this chapter are:

- TTY Terms
- TTY Management Procedures
- Expert TTY Information

TTY Terms

We use the following terms in this chapter:

tty — Derived from the abbreviation for teletypewriter, the term covers the whole area of access between the DG/UX system and peripheral devices, including the system console. A tty is commonly known as a terminal.

tty line — The physical equipment through which access to the computer is made. Also known as a terminal line or port.

linesets — A group of settings (modes) in **/etc/gettydefs** that govern baud rate, special characters, echoing, etc., as defined in **termio**(7). The tty line and the terminal must be in the same mode before communication can take place. Lineset names usually refer to baud rate, such as 9600.

baud rate — The speed at which data is transmitted, measured in state changes per second.

hunt sequence — A lineset structure that allows a user to try more than one lineset until a match is found between a tty line and a terminal. Used mainly for dial-up lines. The DG/UX system ships with a default hunt sequence.

login state — Either **off** or **on**. If a line's login state is **off**, no one can log in on the line. Some printers require that their line's login state be set to **off**. If a line's login state is **on**, users can log in over that line.

TTY Management Procedures

If you select **ttymgmt** from the Main Menu, the system displays the following menu:

```
                TTY Management

1 ttydefaults   Define tty default settings
2 addtty        Add a single tty entry
3 deltty        Delete a tty entry
4 modtty        Modify a tty entry
5 lstty         List tty entries
6 installtty    Add multiple tty entries

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, or q to QUIT:
```

The following sections discuss the options you may select.

Defining TTY Default Settings

Purpose	To set terminal defaults.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**ttydefaults**
References	**tty**(7), **inittab**(4)

Select option 1, **ttydefaults**, to define the default values for other **ttymgmt** functions. These default values include login state, lineset name, hangup delay, TERM variable, and an optional description of every terminal. These defaults come with values already set. You should verify that each of these defaults is correct for your system; if so, press the New Line key. If the preset value is incorrect for your system, set the value you want. When you select **ttydefaults**, the system will engage you in an interaction like the following:

```
Default Login State? [on] ♪

Default Lineset Name? [9600]  ♪

Default Hangup Delay (in seconds)? [0] ♪

Default TERM Variable?  [vt100]  ♪

Default Description? ♪

The tty defaults have been set.

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Adding a Single TTY to the System

Purpose	To add one terminal at a time to the system.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**addtty**
References	**tty**(7), **inittab**(4)

Select option 2, **addtty**, to add a tty line entry to your **/etc/inittab** table. The tty name must refer to an existing device file in the **/dev** directory. The terminal controllers, such as a Systech Asynchronous Controller (**syac**), on your CPU will determine how many tty entries are in **/dev**. These device files are created automatically by the kernel at boot time. Files are of the form **/dev/tty***N* where *N* is a number consisting of two or more digits; for example, **tty01**, **tty02**, **tty03**, or **tty100**.

When you make a tty available for login use, you are putting a tty name that already exists in **/dev** into the tty table in **/etc/inittab**.

To add a tty to your system, select the **addtty** command. The system queries you, and you may respond, as follows:

```
TTY Number?  ? ♪
```

Typing ? displays the following message (*N* is the highest numbered tty on your system):

```
Enter the number of an existing tty device.  For example,
to add the device '/dev/tty01', enter '01'.  The tty numbers
are in the range: 00    N.
```

Once you have decided what tty you want to add, type it at the `TTY Number?` prompt and press New Line.

```
TTY Number?  07 ♪
```

Then you see this prompt:

```
Login state? [on]  ♪
```

By pressing New Line, we got the default state for logins, `on`. To use this tty for some other purpose (such as a laser printer) answer `off`. The example dialog with **addtty** continues:

```
Lineset Name? [9600] ?↵

   Enter the lineset name.  The lineset defines the terminal
   modes and login banner used for logging in.  The available
   linesets are:

   console         19200           M2400           M9600
   9600            19200EP         M4800           M300
   9600EP          M1200

Lineset Name? [9600] ↵
Hangup Delay (in seconds)? [0] ↵
```

The hangup delay specifies how many seconds the **getty**(1M) program will wait for the user to enter the login name. Delays are often used on dial-up lines to avoid wasting telephone connect time; the delay is generally 60 to 120 seconds. A delay of 0 is often used on direct tty lines. The maximum delay is 600 seconds (ten minutes). Next, **addtty** prompts you for a TERM variable value:

```
TERM Variable? [vt100] ↵
```

You may define the initial value of the TERM variable for this tty. We selected the default TERM variable value. This environment variable is used by programs like **vi** to determine the type of terminal that is on this tty. The value of TERM will be exported to whatever login program is used by the person logging on to tty01. If you add a TERM value that does not correspond to an entry in **/usr/lib/terminfo**, you will receive an error message.

The next prompt you see asks if you want the terminal to be available when the system is at run level 1 (**init** administrative state).

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

Finally, you have the option of writing a description for the tty line. This comment may be whatever you choose, as long as it is only one line in length.

```
Description? Terminal at end of hallway 8 ↵

TTY 01 has been added.

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Deleting TTYs from the System

Purpose	To delete a tty entry from the **inittab** table.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**deltty**
References	**inittab**(4)

Select option 3, **deltty**, to delete a tty entry from your **/etc/inittab** table. You might want to delete a tty if you have removed a terminal controller board, thus reducing your tty capacity. But generally, you will not delete tty entries. To do so removes them entirely from **/etc/inittab**. If you simply want to disable a tty, use the **modtty** command and turn the login state to `off`.

To delete a tty, select the **deltty** subcommand. Shown below is a sample interaction:

```
TTY Number? 12 ♪

Do you really want to delete TTY 12? [no] y ♪

TTY 12 has been deleted.

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Modifying TTYs

Purpose	To modify an entry in the **inittab** table. To set an administrative state CRT.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**modtty**
References	**inittab**(4)

Select option 4, **modtty**, to modify a tty entry. Use this function to change settings on ttys. Settings you may want to change from time to time are a tty's description or its Administrative State privilege, which determines if the tty is available when the system is at run level 1.

The tty to be modified must already have an entry in **/etc/inittab**. The default for each query in this function is whatever value you have set previously.

To modify a tty, select the **modtty** command. In the following example, we modify the entry for line 14 so that no one can log into the system over line 14.

```
TTY Number?    14 ♪

Login state? [on]   off ♪

Lineset Name? [9600] ♪

Hangup delay (in seconds)? [60] ♪

TERM variable? [vt100] ♪

Available in Init Administrative State? [no] ♪

Description? Line disabled 8/21/90 ♪

TTY 14 has been modified.

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

We chose the "off" login state, the default lineset, 60 seconds for the hangup delay, and the default TERM variable. The `Description` query is there for your benefit; you can leave it blank by pressing New Line or you can enter a description as we did above.

Listing TTYs

Purpose	To list tty entries in the **inittab** table. To give tty information.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**lstty**
References	**inittab**(4)

Select option *5*, **lstty**, to list ttys and get information about them. Here is an example listing:

```
TTY Numbers? [all] ♪

                    Hangup      Line
 TTY       State    Delay       Set       Description
-----      ----     -----     --------    -----------
 00         on         0         9600     Admin Terminal
 01         on         0         9600     C. Roach
 02         off        0         9600     C. Robin

Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Adding Multiple TTYs

Purpose	To add more than one terminal at a time to the system.
Starting Conditions	administrative or multiuser state
sysadm menu	ttymgmt
Commands	**installtty**
References	**tty**(7), **inittab**(4)

Select option 6, **installtty**, to install a tty entry into **/etc/inittab**. A tty name must refer to an existing device file in the **/dev** directory. The terminal controllers on your CPU will determine how many tty device files are in **/dev**. For instance, if you have 32 lines, then you will have 32 tty device files in **/dev**. These device files are created automatically by the kernel at boot time. Files are of the form **/dev/tty***N*, where *N* is a number consisting of two or more digits. Some example tty file names are **tty00**, **tty01**, **tty100**, and so on.

Making a tty available for login use means adding an entry for the tty (which must already have exist in **/dev**) to the **/etc/inittab** file.

The procedure for adding tty lines depends on the number of tty device files that you have in **/dev**. If you have fewer than 128, use the procedure described below. If you have 128 or more, run **sysadm newdgux** and add a line in which you set the NPROC parameter to some appropriate value. NPROC determines the maximum number of processes that can be active on your system at one time. The following example line, which you could put anywhere in your system file, sets NPROC to a figure higher than its default of 256:

```
NPROC      300
```

This will prevent 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.

When you select **installtty**, the system will prompt you for information as shown in the following interaction:

```
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 can 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? [yes] ♪
Sysadm will now create the tty entries...
Press the New Line key to see the ttymgmt menu [?, ^, q]:
```

Expert TTY Information

The remaining sections contain expert-level information that describes alternative methods for tty administration. You do not need to read these sections to manage your system; we provide them for readers who want more details.

How the TTY System Works

A series of four processes connects a user to the DG/UX system: **init**(1M), **getty**(1M), **login**(1), and a shell (**sh**(1) or **csh**(1)). The **init** program is a general process spawner that is invoked in the boot procedure and every time you change run levels (Chapter 3 discusses run levels). It spawns a **getty** process for each line that a user may log in on, guided by instructions it reads from **/etc/inittab**. The **getty** program requires the name of a special file from **/dev**, such as **tty01**, as a *line* argument.

A user attempting to make a connection generates a request-to-send signal that is routed by the hardware to the **getty** process for one of the tty line files in **/dev**. The **getty** process responds by sending an entry from file **/etc/gettydefs** down the line. The **gettydefs** entry used depends on the *speed* argument used with the **getty** command. (In the SYNTAX of the **getty**(1M) command, the argument name is *speed*, but it is really a pointer to the *label* field of a **gettydefs** entry.) If there is no *speed* argument, **getty** uses the first entry in **/etc/gettydefs**. The login prompt is among the fields in the **gettydefs** entry. See the **gettydefs**(4) manual page for more information.

On receiving the login prompt, the user enters a login name. Then **getty** starts a program named **login**, using the login name as an argument. The **login** program sends the prompt for a password, evaluates the user's response against the **passwd** file, and assuming the password is acceptable, calls in the user's shell as listed in the **/etc/passwd** entry for the login name. If no shell is named, **/bin/sh** is furnished by default. Upon login, a user's shell executes **/etc/profile** for **sh** users, and executes **/etc/login.csh** for **csh** users.

The **/bin/sh** program executes the user's **.profile** for **sh** and **/bin/csh** executes the user's **.login** for **csh**. These files often contain **stty** commands that allow a user to reset terminal options should they not want the defaults.

Modifying TTY Linesets in /etc/inittab

You have two ways to modify tty linesets:

- Use the **sysadm modtty** command. This command leads you through a series of prompts. Your responses edit a **getty** entry in **/etc/inittab**.
- Use a text editor to edit **/etc/inittab**.

The **/etc/inittab** file contains instructions for the **/etc/init**(1M) command. The general format of a line entry in the **/etc/inittab** file is as follows.

identification:level:action:process

The four colon-separated fields are as follows:

identification	A unique one- or two-character identifier for the line entry.
level	The run level in which the entry is to be performed. If this entry is blank, **init** performs the action at every run level. Valid values are **s** and the digits **0** through **6**. Chapter 3 discusses run levels.
action	How **/etc/init** treats the process field.
process	The command line to be executed.

The **/etc/inittab** directory contains several entries that spawn **getty** processes. The following example is a selection of such entries from an **/etc/inittab** file.

```
con::respawn:/etc/getty console console
01:23:respawn:/etc/getty tty01 vt100 9600
02:23:respawn:/etc/getty tty02 vt100 9600
03:23:respawn:/etc/getty tty03 vt100 9600
04:23:off:/etc/getty tty04 vt100 9600#line not in use
05:23:off:/etc/getty tty05 vt100 9600#line not in use
```

There are at least three things you might want to do to an **inittab** entry for a tty line:

- Change the action. Two actions that apply to tty lines are "respawn" and "off" (see the **inittab**(4) manual page for complete information on this field).
- Add or change arguments to **/etc/getty** in the process field. A frequently used argument is **−t***nn*. This tells **getty** to hang up if nothing is received within *nn* seconds. It's good practice to use the **−t** argument on dial-up lines.
- Add or change comments. Insert comments after a pound sign (**#**).

Setting Terminal Options

Once the user has successfully logged in, he or she may find that certain terminal options would be preferable to ones in the default set.

The command to control terminal options is **stty**(1). Many users add an **stty** command to their **.profile** (or **.login** for **csh**) so the options they want are automatically set as part of the **login** process. Here is an example of a simple **stty** command:

```
$ stty cr0 nl0 echoe –tabs erase ^H
```

The options in the example mean the following:

cr0 nl0 — No delay for Carriage Return or New Line. Delays are not used on a video display terminal, but are necessary on some hardcopy terminals to allow time for the mechanical parts of the equipment to move.

echoe — Erase characters as you backspace.

–tabs — Expand spaces to tabs when printing.

erase ^H — Change the character-delete character to a ^H. The default character-delete character is the pound sign (#). Most terminals transmit ^H when the **backspace** key is pressed.

Interpreting Linesets in /etc/gettydefs

The **/etc/gettydefs** file contains information used by the **getty**(1M) command to establish the speed and terminal settings for a line. The general format of the **gettydefs** file is

```
label# initial-flags # final-flags #login-prompt #next-label
```

The following are sample lines from a **gettydefs** file:

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600

9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800

4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400

2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200

1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300

300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
```

The entries in the **gettydefs** file form a single, circular hunt sequence; the last field on each line is the label of the next line. The `next-label` field for the last line shown points back to the first line in the sequence. The object of the hunt sequence is to link a range of line speeds so that when you try to log in to a dial-up line, say at 300 baud, the system will be able to match that rate. If you first get nonsense or garbage characters instead of a clear login prompt, press the BREAK key to cause **getty** to step to the next entry in the sequence. The hunt continues until the baud rate of the line matches that of your terminal. The flag fields shown have the following meanings:

B300-B19200	The baud rate of the line.
HUPCL	Hang up on close (terminate).
SANE	A composite flag that stands for a set of acceptable line characteristics.
IXANY	Allow any character to restart output. If this flag is not specified, only DC1 (Ctrl-Q) will restart output.
TAB3	Send tabs to the terminal as spaces.

For a description of all **getty** flags, see **termio**(7).

Changing TTY Linesets in /etc/gettydefs

You must be superuser to edit the **/etc/gettydefs** file. In the following example, we will change the login banner for a lineset that starts at M300 and ranges to M1200 (baud range 300-1200). First, before invoking a text editor to make the change, copy **/etc/gettydefs** into **/tmp/gettydefs**. This keeps the original safe until you have had a chance to verify your changes. Type:

```
# cp /etc/gettydefs /tmp/gettydefs ↵

# vi /tmp/gettydefs ↵
```

In the example, we use the **vi** text editor. Before editing the copy of **gettydefs**, the entry for the **M300 getty** definition looks like this:

```
M300# B300 ECHO ECHOE ECHOK KILL ^u ERASE ^? INTR ^c HUPCL CS8 \
 OPOST ONLCR ICRNL CREAD ISTRIP # B300 ECHO ECHOE ECHOK IEXTEN \
 KILL ^u ERASE ^? INTR ^c HUPCL ICAN ON CS8 OPOST ICRNL ONLCR \
 ISTRIP CREAD IXON IXOFF ISIG TAB3 #login: #M300
```

NOTE: A getty definition is all on a single line. If you edit **gettydefs**, make sure your editor does not insert New Line characters in the middle of any entries. In the **vi** editor, use the **J** command to join a line with the line that follows it. Remember that even after joining lines, an entry may still appear to break over multiple lines due to the width of your terminal screen.

After editing, the entry looks like this:

```
M300# B300 RCHO ECHOE ECHOK KILL ^u ERASE ^? INTR ^c HUPCL CS8 \
 OPOST ONLCR ICRNL CREAD ISTRIP # B300 ECHO ECHOE ECHOK IEXTEN \
 KILL ^u ERASE ^? INTR ^c HUPCL ICAN ON CS8 OPOST ICRNL ONLCR \
 ISTRIP CREAD IXON IXOFF ISIG TAB3 #System ZEBRA login: #M300
```

Besides changing `#login:` to `#System ZEBRA login:`, we accidentally changed `ECHO` to `RCHO`. Errors like this are not a problem, however, because **getty** provides a way of catching them. Any time you edit **/etc/gettydefs** (or a copy of it, as we

suggested), you should check the file to see if there are any unrecognized modes or improperly constructed entries. Type:

```
# getty -c /tmp/gettydefs > out ♪
```

The **getty -c** command checks the file then prints out the result of that check in a file named **out** (or whatever you choose). If you look at **out**, you'll see:

```
M300# B300 RCHO ECHOE ECHOK KILL ^u ERASE ^? INTR ^c HUPCL CS8 \
 OPOST ONLCR ICRNL CREAD ISTRIP # B300 ECHO ECHOE ECHOK IEXTEN \
 KILL ^u ERASE ^? INTR ^c HUPCL ICAN ON CS8 OPOST ICRNL ONLCR \
 ISTRIP CREAD IXON IXOFF ISIG TAB3 #System ZEBRA login: #M300

UNDEFINED: RCHO
```

After you correct the error, run the **getty -c** command again to verify that it does not contain errors; then use the **mv** command to write the temporary file over the original:

```
# mv /tmp/gettydefs /etc/gettydefs ♪
```

Your changes to a **gettydefs** entry take effect the next time a **getty** process begins execution and reads its definition from **gettydefs**.

End of Chapter

Chapter 11
LP System Management

This chapter shows you how to manage the DG/UX LP system. Originally, LP stood for line printer, but it is now understood to include laser printers as well. Another term often associated with LP systems is spool. The term spool is an acronym for "simultaneous peripheral output online." So when we say LP spooling, we mean that print jobs are queued to be printed on a specific printer, while you continue with other work.

You manage the LP system by selecting **sysadm lpmgmt** menu options or by executing the corresponding commands. Then you respond to the queries that are printed to your screen. To get online help for any question, type ?.

The major sections of this chapter are:

- LP System Terms
- LP System Procedures
- Printing Path
- LP Directories and Files
- Expert LP Information

LP System Terms

Become familiar with these LP terms before reading on:

scheduler A program that assigns requests to printer queues. The LP scheduler starts automatically when the DG/UX system comes up.

queue An ordered list of requests (jobs) to print.

accept A mode in which a printer will put requests in a queue.

reject A mode in which a printer will not put requests in a queue.

enable To allow printing on a given printer.

disable To stop all printing on a given printer.

device file A file that represents a physical input/output unit, such as **/dev/tty01**.

model A prototype printer interface program for sending output to a printer. The following prototype interface programs are in **/var/spool/lp/model**:

```
455x        async_300   async_9600  lpj         ps_9600
async_1200  async_4800  dg455x      parallel    remshlp
async_2400  async_600   dumb        parallel-2  termprinter
```

For an interface program to work with your printer, you may need to copy the program and change it. See the entry for **/var/spool/lp/model** in the section "LP Directories and Files" for more information on these interface programs.

The scheduler program must be running for the LP system to function. You will be asked to start and stop it during the execution of various **sysadm** commands. After a printer has been enabled and placed in accept mode, the scheduler can assign your printing requests to that printer.

LP System Procedures

When you select **lpmgmt** from the Main Menu, the following is displayed:

```
                    Line Printer Management

  1 addlp           Define a new printer
  2 dellp           Delete a printer
  3 modlp           Modify an existing printer
  4 lslp            List printers
  5 defaultlp       Define the default printer
  6 acceptlp        Set a printer to accept print requests
  7 rejectlp        Set a printer to reject print requests
  8 enablelp        Enable a printer
  9 disablelp       Disable a printer
 10 queuelp         Display the print queue of a printer
 11 cancellp        Cancel print requests
 12 movelp          Move print requests from one printer to another
 13 startlp         Start the lp scheduler
 14 stoplp          Stop the lp scheduler

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, or q to QUIT:
```

The LP system displays error messages when necessary. See Appendix C for a listing and explanation of LP error messages.

The following sections discuss the options available from the Line Printer Management Menu.

Adding Printers

Purpose	Add printers to the LP system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**addlp**
References	**lpadmin**(1M)

To add a printer to your system, select option 1, **addlp**. A printer name may be comprised of letters, numbers, and underscores. The name you give a printer is associated with a local or remote printer. For a local printer, one attached to the system you are administering, the printer name is associated with a device in **/dev** and to an interface script (called a *model*) that sends output to the device. Local printers use model **dumb** by default.

You can "add" a remote printer (one attached to another processor) by specifying the name of a remote system, then specifying the name of the remote printer. Remote printers use model **remshlp** by default. Once added, users on the local system can access the remote printer as though it were local. You should already have installed TCP/IP before you attempt to add a remote printer to your system. Without TCP/IP, you cannot make the remote connection. Because **lp** uses the remote shell (**remsh** or **rsh**) to submit a remote printing job, the system on which you submit the job must have an entry in the **/etc/hosts.equiv** file of the system where the printer is installed. See the **remsh**(1C) manual page.

To add a printer, select the **addlp** command from the Line Printer Management Menu. Messages having to do with the scheduler appear only if the scheduler is already running. The following example shows how we add a serial printer.

```
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? newlp
Is this a local printer? [yes]  ♪
Printer model? [dumb] ♪

Printer device file?  list ♪
The available devices are:
```

```
lp
tty00 through tty15

Printer device file?  tty01 ♪
```

If we want to add a remote printer, say one named **newlp** on system **system10** in our network, we would first have to make sure that our system name appears in the **/etc/hosts.equiv** file on **system10**. Then we may add the printer on our system, following the same dialog as before until we get to the local printer query. We respond to the local printer query with **n**, continuing like this:

```
Is this a local printer? [yes] n ♪
Remote host name? system10 ♪
Remote printer name? newlp ♪
```

Note that for model (printer interface program), we specified **dumb** for the local case. For the remote case, **remshlp** is automatically used. After this point, whether you're adding a remote or local printer, **sysadm** dialogues are the same.

```
newlp has been added.
```

Notice that we were asked for a device file (from **/dev**) only for the local printer. For the remote printer, **sysadm** requires the host name and the printer name; no device files are used in the remote case. The **addlp** dialog continues like this:

```
Accept and Enable newlp? [yes] ♪

newlp has been enabled.
newlp has been set to accept requests.

Restart the scheduler now? [yes] ♪
The scheduler has been restarted.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Select the **addlp** command again to add another printer.

Deleting Printers

Purpose	Remove printers from your system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**dellp**
References	**lpadmin**(1)

To delete a printer from your system, select option 2, **dellp**. Deleting a printer with **dellp** disconnects that printer's name from its association with a device file in **/dev**. For example, if you delete **newlp**, which is associated with **/dev/tty01**, then you are deleting the printer name only. No device files are deleted in **/dev**. After you delete a printer, any attempt to use it will generate an appropriate error message.

Make sure that no one is using a printer before you delete it. To delete a printer, select the **dellp** command. A sample dialog for **dellp** follows.

```
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 delete 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.

Which printer? newlp ♪
Printer newlp has been deleted.

Restart the scheduler now? [yes] ♪

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Modifying an Existing Printer

Purpose	To change the attributes of an existing printer.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**modlp**
References	**lpadmin**(1M)

To modify a printer on your system, select option 3, **modlp**. Use **modlp** to change a printer's name or any values associated with it. Defaults shown in the prompts are the previous values that were given to the printer. If you are modifying a remote printer, you must have TCP/IP running.

The following example shows the dialog that results when we use **modlp** to change a printer's name and terminal line.

```
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 mod 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? newlp ♪
New Printer Name? [newlp] oldlp ♪
Is this a local printer? [yes]  ♪
Printer model? [dumb] ♪

Printer device file?  list ♪
The available devices are:

        tty00 through tty15

Printer device file? [tty01] tty02 ♪
```

If we had wanted to modify a remote printer, say one named **newlp** on another system named **system10**, we would have done the following:

```
Is this a local printer? [yes] n ♪
Remote host name? system10 ♪
Remote printer name? newlp ♪
```

Note that for models (printer interface programs), we specified **dumb** for the local case. **sysadm** uses **remshlp** for the remote case. After this point, whether you're adding a remote or local printer, **sysadm** dialogues are the same.

```
newlp has been modified.
```

Notice that we were asked for a device file (from **/dev**) only for the local printer. For the remote printer, **sysadm** requires the host name and the printer name; no device files are used in the remote case. The **modlp** dialog continues below.

```
Accept and Enable oldlp? [yes] ♪

oldlp has been enabled.
oldlp has been set to accept requests.

Restart the scheduler now? [yes] ♪
The scheduler has been restarted.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Listing Printers

Purpose	To list printers on the LP system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**lslp**
References	**lpstat**(1M)

To list the characteristics for a printer on your system, select option 4, **lslp**. Characteristics consist of the name, device, enable/disable mode, accept/reject mode, and the length of the print queue.

To list the printers, select the **lslp** command. An example **lslp** dialog follows:

```
          Which printer? [all]  ↵

Printer Name       Device           Enabled?  Accept?    Requests
------------    ---------------     --------  -------    --------
bigboy          /bigboy@system10    enabled   accepting     2
laser           /dev/tty01          disabled  rejecting     0
newlp           /dev/lp01           enabled   accepting     1

          Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Notice the three different `Device` entries. Printer `bigboy` is on a remote processor named `system10` and is not associated with a device file. Printer `laser` is a serial laser printer, so it is associated with a tty device file. Printer `newlp` is a parallel line printer, so it is associated with an **lp** device file.

If you know the name of a particular printer for which you want information, specify that printer's name at the `Which printer?` prompt.

Setting the Default Printer

Purpose	Define the default printer on your system.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**defaultlp**
References	**lpadmin**(1M)

To set the default printer on your system, select option 5, **defaultlp**. The default printer is the printer that is used when someone executes an **lp** command without selecting a particular printer with the destination switch, **-d**.

To define a printer as the default, select the **defaultlp** command. An example **defaultlp** dialog follows:

```
There is no current default.

New default printer?  newlp ♪

The new default printer is: newlp.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Setting a Printer to Accept Requests

Purpose	To put a printer into accept mode.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**acceptlp**
References	**accept**(1M)

To set a printer to accept print requests, select option 6, **acceptlp**. When a printer is set to accept requests, it will queue all requests. If a printer will be unavailable for a short time, you may disable it but leave it in accept mode. In accept mode, the printer will continue to accept job requests, and it will print them as soon as you enable it. An example **acceptlp** dialog follows:

```
Which printer (for accept)? [newlp] ♪

Printer newlp has been set to accept requests.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Now, printer **newlp** will accept jobs submitted to it, that is, jobs will go into a queue associated with **newlp**. If it has not been put in accept mode, **sysadm** prints a message telling the user that **lp** cannot accept requests for that destination.

Setting a Printer to Reject Requests

Purpose	To put a printer into reject mode.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**rejectlp**
References	**accept**(1M)

To set a printer to reject requests, select option 7, **rejectlp**. Once placed in reject mode, the printer will continue printing whatever has been queued. When users attempt to make requests to a printer that has been put in reject mode, a message will be displayed explaining why the printer is not accepting new print requests. You might use the **rejectlp** command when a very large job is printing and you want to divert users elsewhere. Or, you might use this command when you know that a printer will be down long enough to inconvenience users.

An example **rejectlp** dialog follows:

```
Which printer (for reject)? [newlp] ♪

Please give a one-line reason for rejecting requests
on printer newlp.

Reason? Printer newlp is down for repairs.

Printer newlp has been set to reject requests.

Press the NEWLINE key to see the lpmgmt menu [?,^,q]:
```

Starting and Stopping Printers

Purpose	To start and stop a printer.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**enablelp** **disablelp**
References	**enable**(1)

Enabling a Printer

To enable a printer, select option 8, **enablelp**. To disable a printer, select option 9, **disablelp**. When you enable a printer, it is free to print whatever jobs are in its queue. (If the printer is in reject mode, however, it will not allow users to submit print requests–use the **acceptlp** command to put the printer in accept mode.) An example **enablelp** dialog follows:

```
Which printer (for enable)? [newlp] laser ♪

Printer laser has been enabled.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

If you want to see the list of available printers, type **?**.

Disabling a Printer

If a printer is disabled, it will not print any jobs in the queue. The printer *will* continue to accept jobs (queue them). An example **disablelp** dialog follows:

```
Which printer (for disable)? [newlp] ♪

Please give a one-line reason for disabling printer newlp.
Reason?  Printer is very busy right now.

Printer newlp has been disabled.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

The reason you provide appears when a user issues the **lpstat –t** command.

Displaying the Print Job Queue

Purpose	To display print requests (jobs) in the queue.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**queuelp**
References	**lpstat**(1)

To display a printer's print queue, select option 10, **queuelp**. A print queue is the list of print requests waiting to be sent to a printer. In the **queuelp** display, the first line shows the next job to be printed. Individual requests are numbered and attached to the printer name.

To list print requests (jobs) in the queue, select the **queuelp** command. An example **queuelp** dialog follows:

```
Which printer (to list)? [newlp] ♪

Printer newlp:

Request                User      Size      Time
---------              -----     ----    -----------
newlp-11                root     17      Jul 27 18:23
newlp-12               della     4224    Jul 27 18:30
newlp-13               tobor     2345    Jul 27 18:33

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Here, we chose to see the queue of the default printer, `newlp`. Typing **all** will list the queues of all printers, one printer at a time.

Canceling a Job

Purpose	To cancel a job in the LP system queue.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**cancellp**
References	**lp**(1), **cancel**(1)

To cancel a print request, select option 11, **cancellp**. The requested job may be printing or waiting to print. If you cancel a job while it is printing, it will be stopped, and the next job will begin printing.

To cancel a job in the queue, select the **cancellp** command. An example **cancellp** dialog follows:

```
Which printer (for cancel)? [newlp] ♪

The current requests on newlp are:

Request              User     Size       Time
----------           -----    ----    ------------
newlp-11              root    17      Jul 27 18:23
newlp-12             della    4224    Jul 27 18:30
newlp-13             falco    2345    Jul 27 18:33

Which request(s) should be canceled? [none] newlp-11 ♪

  The requests have been canceled.

  Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Printer internal buffer sizes may determine how much of the job prints before cancellation takes effect.

Moving Jobs to a New Printer

Purpose	To move jobs in one printer's queue to the queue of another printer.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**movelp**
References	**lpsched**(1M)

To move print requests from one printer queue to another, select option 12, **movelp**. The **movelp** command is useful when you need to take one printer out of service and have another printer available to handle its work load. You may specify which print requests (or all requests) you want **movelp** to move. If all are moved, it will place the printer with the emptied queue in reject mode.

To move requests from one printer to another, select the **movelp** command. A sample dialog follows:

```
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 move 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.

From printer? newlp ♪
To printer? bigboy ♪

The current requests on newlp are:

Request              User      Size      Time
---------            -------   ----    -----------
newlp-12               della    4224    Jul 27 18:30
newlp-13               tobor    4550    Jul 27 18:33

Which request? [none]  all ♪

The requests have been moved.

Because you have moved all requests from newlp to bigboy,
sysadm has left newlp in the reject mode.  All attempts to
use newlp will fail until you use acceptlp to change the mode.
```

If you used the **queuelp** command for printer bigboy, you'd see that the jobs have been moved.

```
Restart the scheduler? [yes] ♪
The scheduler has been restarted.

Press the NEWLINE key to see the lpmgmt menu [?, ^, q]:
```

Starting and Stopping the LP Scheduler

Purpose	To start and stop the LP scheduler, the program that sends print requests to the printers. If the scheduler is stopped, the entire LP system stops.
Starting Conditions	administrative or multiuser state
sysadm menu	lpmgmt
Commands	**startlp** **stoplp**
References	**lpsched**(1M), **lpshut**(1M)

To start the LP scheduler, select option 13, **startlp**. To stop the LP scheduler, select option 14, **stoplp**. The LP scheduler is the program that actually sends print requests to the printers. When the scheduler is not running, the entire printer subsystem is not running.

Use the **stoplp** command to stop the scheduler so that you can perform administrative tasks. When you're ready to start it again, use the **startlp** command. The **stoplp** command responds with:

```
The lp scheduler has been stopped.
```

The **startlp** command responds with:

```
The lp scheduler is now running.
```

Printing Path

When a user issues this command:

```
$ lp -dlaser1 memo ↵
```

here is what happens:

1. The user includes the **–d** option with the **lp** command to specify that **memo** be printed on a printer named **laser1**.
2. The **lp** command puts the request in the queue, **/var/spool/lp/outputq**.
3. The **lpsched** program reads the request from the queue file and passes it to the interface program.
4. The interface program sends the request to the specified printer to print **memo**.

Figure 11-1 shows the path that the file takes.

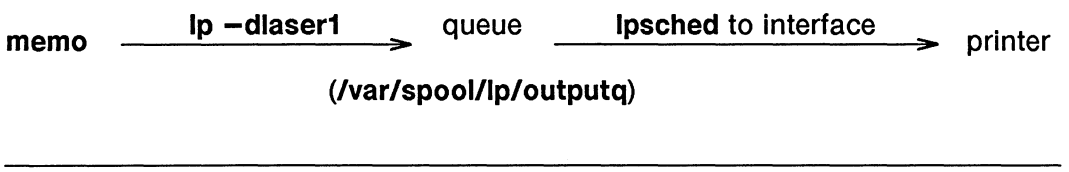


Figure 11-1 *LP System Print Path*

LP Directories and Files

This section describes the files and directories in the LP system. You should never need to manipulate these files yourself, but if your LP system begins to behave abnormally, you may find this information useful during diagnosis.

/var/spool/lp/class

The **class** directory may not exist on your system (it is optional). This directory contains files named for each LP class that has been identified. The class file identifies each member, in this case an LP printer, that is assigned to the class. Class files are created, modified, and deleted by the **lpadmin** command. A class is a group of printers that you have named such for reasons of performance or location. For instance, you might have a class named LPB1. In class LPB1 are printers LPB2 and LPB3. Requests to print go to the class, then the printer that is the least busy prints the job request.

/var/spool/lp/FIF0

FIFO is a special file that all the commands use to send messages to **lpsched.** Any of the LP commands may write to **FIFO,** but only **lpsched** may read it.

/var/spool/lp/default

The **default** file contains the name of the system default destination printer. If this file does not exist or if it is empty, the LP system has no default printer.

/var/spool/lp/interface

The **interface** directory contains one executable interface script for each printer that is in the LP system. The file name of the interface script is the same as the printer name. The interface program is invoked with its standard error and standard output directed to the printer. Interface programs may be shell procedures or compiled C programs.

/var/spool/lp/log

The **log** file contains records of all the printing activity that has taken place since the LP scheduler was last started. This file contains the logname of the user who made the request, the request ID, the name of the printer that the request was printed on, and the date and time that printing started. Any **lpsched** error messages that occur are also recorded in the **log** file. The first line of the log file shows the time that the LP scheduler was started.

/var/spool/lp/member

The **member** directory contains one file for each LP printer. The file name is the same as the printer name. Each file contains the path name of the device to which the member is connected.

/var/spool/lp/oldlog

The **oldlog** file contains a record of what was in the previous **log** file. When the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, all the information that had accumulated in the **log** file is copied to the **oldlog** file, and a new **log** file is started. Any information that had been in the **oldlog** file is overwritten. The first line of the file tells the time that the scheduler was turned on, and the last line tells the time the scheduler was turned off.

/var/spool/lp/outputq

The **output1** file is a binary file. The **lp** command writes to this file and the LP scheduler reads from it. When a user submits a print request, the **lp** command puts an entry in the **outputq** file. The LP scheduler takes the job request and passes it to the appropriate interface program to be printed. When the job is completed, a binary tag is attached to that job entry. Also, whenever you manipulate a particular job with the commands **lpmove**, **disable**, **lpsched**, or **cancel**, an appropriate binary tag is attached to the **outputq** entry for the job in question. This tag tells the scheduler not to run the job again.

/var/spool/lp/pstatus

The binary file **pstatus** contains status information for each printer. Entries are added to and removed from this file by the **lpadmin** command, and the entries are modified by the **cancel**, **enable**, **disable**, and **lpsched** commands. When the **lpstat** command is invoked with the **−p** option, printer status information is obtained from this file.

/var/spool/lp/qstatus

The **qstatus** file is a binary file that keeps track of whether a destination printer is accepting or rejecting requests. Entries are added or removed from this file by the **lpadmin** command and modified by the **accept** and **reject** commands. When the **lpstat** command is invoked with the **−o** option, the request status is obtained from this file.

/var/spool/lp/seqfile

The **seqfile** file contains the sequence number of the last request ID that was assigned by the **lp** command. The sequence number (*seqno*) is incremented by **lp** for each request. When the number 10101010 is reached, the sequence number is reset to 1.

/var/spool/lp/model

The **model** directory contains the prototype printer interface programs that are distributed with the DG/UX system. You do not invoke these programs yourself; instead, you provide the name of an interface program to **addlp** (when you add a printer queue) when it prompts for the printer model. You may need to change an interface program to make it work with your printer. These programs are

async_300, async 600, async_1200, async_2400, async_4800, async_9600
Line printer interface script for asynchronous line printer. Sets line to the stated baud rate, x-on/x-off protocol, tabs are expanded before printing.

dg455x, 455x
Line printer interface script for Data General Model 4557 or 4558 laser printer (**455x** comes with Documenter's Tool Kit, DTK). Supports two options:

−o66 66-lines per 11-inch page -- compressed vertical spacing

−o62 62-lines per 11-inch page -- normal 6 lines/inch (default)

parallel
Line printer interface script for generic parallel line printers (similar to **async_9600** but without the **stty** settings).

parallel-2
Same as above, but maps New Line to Carriage Return/New Line. Line Printer Interface Script for generic parallel line printers.

lpj Line printer interface script for DG asynchronous line printer. Sets line to 2400 baud, x-on/x-off protocol, tabs are expanded before printing.

remshlp
Line printer interface script for network printer access. Executes the **lp** command on a remote system through a remote shell (**remsh**(1C)). Options available on the remote printer are passed to it.

termprinter
Line printer interface script for printers connected to the TermServer product.

dumb
Interface for a line printer without special functions and protocol. Form-feeds are assumed. This is a good model to copy and modify for printers that do not have models.

Administrators may also add their own interface programs to this directory.

/var/spool/lp/request

The **request** directory contains a subdirectory for each destination in the LP system. The name of the subdirectory is the same as the name of the destination. When an **lp** request is made, a **request** file (which has the prefix **r-**) and, in most cases, a **data** file (which has the prefix **d-**) are created in the subdirectory of the destination to which the request is going. The data file stores the file to be printed until the scheduler is ready to print it. A data file is not created if the file to be printed cannot be linked to the request subdirectory.

The name of the request file is derived from the request identification number and is of the form **r-***seqno* (where *seqno* is the job sequence number). The name of the data file is of the form **d***n*-*seqno* (where *n* is a non-negative number and *seqno* is the job sequence number),

The request and data files are deleted by the **cancel** and **lpsched** commands. When you move a print request with the **lpmove** command, **lpmove** moves the data and request files to the appropriate directory.

Lock Files

To guarantee LP commands exclusive access to data files, several lock files are maintained in the LP system. They are binary files that contain the process ID of the locking process. Table 11-1 shows the lock files and their associated data files.

Table 11-1 LP Lock Files and Data Files

Lock File	Data File
OUTQLOCK	**outputq**
PSTATLOCK	**pstatus**
QSTATLOCK	**qstatus**
SEQLOCK	**seqfile**

Lock files "expire" after ten seconds and may be unlinked by any LP process. If a file is older than 10 seconds (not active), the next process will begin. Thus, commands that lock a data file for longer than this interval must update the modification time on the lock file. The creation, updating, and unlinking of lock files is handled automatically by the LP system. Another lock file, **SCHEDLOCK**, is present while the LP scheduler is running to ensure that only one invocation of **lpsched** is active. Unlike other lock files, **SCHEDLOCK** has no expiration time.

Cleaning Out Log Files

As described earlier, when the scheduler is stopped, the **log** file is closed. When the scheduler is restarted, the **log** file is copied to **/var/spool/lp/oldlog**, and a new **log** file is started.

If the scheduler is not stopped for long periods of time and if you have a large number of LP requests, the **log** file can grow to be a large file. You can manually remove the contents of this file, or you can let the system do it for you on a scheduled basis. To clean out the log file periodically, submit a cron job to do it for you:

1. Log in as **sysadm**.

2. Use the **crontab**(1) command to write a list of your **cron** jobs to a file:

   # crontab −l > cronjobs ↵

3. Edit the file, adding a line that performs the desired file deletions at the desired time(s). See the **crontab**(1) manual page for information on entry format.

4. Make sure that the **/var/spool/cron/cron.allow** file exists. If it does not, create it (it does not need to contain any entries). If you have already been running **cron** jobs, this step is not necessary. See the **crontab**(1) manual page for more information.

5. Submit the edited file (which includes not only the new job but also any old jobs that you wish to continue) with the **crontab** command:

   # crontab cronjobs ↵

The following example shows a typical listing of the superuser's **cron** jobs:

```
# crontab −l ↵
0 4 * * * /bin/su - adm -c "/usr/lib/acct/runacct 2> \
                /usr/adm/acct/nite/fd2log"
5 * * * * /bin/su - adm -c "/usr/lib/acct/ckpacct"
15 4 * * 5 /bin/su - adm -c /usr/lib/acct/monacct
0 2 * * * /usr/lib/acct/dodisk
0 3 * * 2-5 /bin/find /tmp /usr/tmp /var/tmp -atime +3 \
                ! -name 'X[0-9]*' -exec rm {} ;
0 23 * * 5 /bin/su lp -c "cp /var/spool/lp/log  \
                                /var/spool/lp/oldlog"
1 23 * * 5 /bin/su lp -c ">/var/spool/lp/log"
```

Look at the last two lines. Every Friday at 11:00 PM **cron**(1M) executes the last two commands. First, the contents of the **log** file are copied to the **oldlog** file, and then the **log** file is cleaned out. So, you have a small, manageable file for the current week, and you can look back and read the file for the previous week.

Expert LP Information

The DG/UX system comes with printer interface scripts for all Data General equipment. If you need to write your own interface script, you should read the following sections. Otherwise, the information in the following sections is for administrators who want to know more about the system. You do not need to read these sections to administer the LP system.

Administrative Commands

Table 11-2 shows a separate set of commands available for the LP administrator. These commands are in the **/usr/lib** directory. If you expect to use the commands frequently, you might find it convenient to include that directory in your PATH variable. To use the administrative commands, you must be logged in as **root** or **lp**.

Table 11-2 Administrative Commands for the LP System

Command	Description
/usr/lib/accept	Permit job requests to be queued for a specified destination.
/usr/lib/reject	Prevent jobs from being queued for a specified destination. Described on the same manual page as **accept**(1M).
/usr/lib/lpadmin	Set up or change the LP configuration.
/usr/lib/lpmove	Move output requests from one destination to another. Described on the same manual page as **lpsched**(1M).
/usr/lib/lpsched	Start the LP scheduler.
/usr/lib/lpshut	Stop the LP scheduler. Described on the same manual page as **lpsched**(1M).

/usr/lib/lpadmin

The **lpadmin**(1M) command is used to add a new printer to the system, assign classes of printers, name or remove a default destination, and specify interface programs to be used. The **lpadmin** command may not be used when the LP scheduler, **lpsched**(1M), is running, except when the **−d** option is specified.

You must include one (and only one) of the following three options on the **lpadmin** command line:

−d[*dest*] Defines an existing system destination as the new default destination. If *dest* is not specified, there is no default destination.

The LP scheduler may be running when you use this option. The default destination is used to determine where a file named in a user's **lp** command is sent. The destination (*dest*) must already exist.

−x*dest* Removes a destination (*dest*) from your system. You cannot invoke this option when the scheduler is running. If the scheduler is running, you must issue the **lpshut** command before **lpadmin.**

No destination (class or printer) may be removed if it has pending requests. The pending requests must either be cancelled using the **cancel**(1) command or moved to other destinations using the **lpmove**(1M) command before the destination can be removed.

Removing the last remaining member of a class causes the class to be deleted. If the destination removed is the system default destination, the system will no longer have a default destination. However, the removal of a class does not imply the removal of printers that were assigned to that class.

−p*printer* Names a printer to which arguments apply. If *printer* does not exist, it is created.

No other options are allowed with **−d** and **−x**. However, many arguments are allowed with the **−p** option, and at least one argument must always be present. The arguments that can be used with **−p** are as follows:

−c*class* Assigns the printer specified in the **−p** option to the specified *class*.

−e*printer* Allows you to use an existing interface program for a new printer that you are adding to the LP system. When you select this argument, the interface program for the printer specified in this argument is copied for the printer specified in the **−p** option.

−h The device associated with the printer is hardwired (not associated with a login terminal). This is the default unless you specify the **−l** option.

−i*interface* Specifies a new interface program for the printer specified in the **−p** option. *interface* is the path name of the new program.

−l Indicates that, when adding a new printer, the device associated with the printer is a login terminal.

−m*model* Select a model interface program for the printer. The interface program *model* exists in **/var/spool/lp/model.**

−r*class* Removes a printer from the specified class.

−v*device* This argument must be used when you add a new printer to the LP system. It associates the printer with the device file specified by *device*. The complete path name must be given for the file.

The following examples demonstrate how to use the **lpadmin** command. In examples 2 through 7, the LP scheduler has already been stopped with the **lpshut**(1M) command. Example 1 does not require the scheduler to be stopped since only the **−d** option to **lpadmin** is used.

Example 1

Make printer **newlp** the system default destination.

```
# lpadmin −dnewlp ♪
```

Example 2

Add a new printer called **fastlp** and associate it with device **/dev/tty11**. Use the **aslp** model interface program.

```
# lpadmin −pfastlp −v/dev/tty11 −maslp ♪
```

When you add a new printer, it is left in a disabled state and does not accept requests.

Example 3

Create a hardwired printer called **lp1** on device **/dev/tty13**. Add **lp1** to a new class called **cl1**, and use the same interface program that is used with printer **fastlp**.

```
# lpadmin −plp1 −v/dev/tty13 −efastlp −ccl1 ♪
```

Example 4

Change the interface program for printer **lp1** to model interface program **dclp**.

```
# lpadmin −plp1 −mdclp ♪
```

Example 5

Add printer **fastlp_2** to class **cl1**:

```
# lpadmin −pfastlp_2 −ccl1 ♪
```

Printers that you add to a class are ordered according to the sequence in which you add them. If all three printers are available, and a request is routed to class **cl1**, the request will be serviced by the one that you added first. If all three printers are busy, the request will be printed by the first available printer.

Example 6

Remove printers **newlp** and **fastlp** from class **cl1**:

```
# lpadmin -pnewlp -rcl1 ♪
# lpadmin -pfastlp -rcl1 ♪
```

/usr/lib/lpsched

The LP scheduler starts automatically each time the system is booted. It does so via an **rc** script named **rc.lpsched**.

The **lpsched**(1M) command starts the LP scheduler. The LP scheduler takes the top job request off the queue and "hands" it to the appropriate interface program to be printed on a printer. The LP scheduler keeps track of the job progress, and as soon as the job is completed, it takes the next job request off the queue and repeats the process. As long as the LP scheduler is running, jobs requested by **lp** will be printed. If the scheduler is not running, jobs will not be printed.

Every time the scheduler is started, **lpsched** creates a file called **SCHEDLOCK** in the **/var/spool/lp** directory. As long as the **SCHEDLOCK** file is present, the system will not allow another scheduler to run. When the scheduler is stopped under normal conditions, either with **lpshut**(1M) or as part of the normal shutdown procedure, the **SCHEDLOCK** file is removed. However, if the system comes down abnormally, there is a possibility that the **SCHEDLOCK** file may not get removed. To ensure that the **SCHEDLOCK** file does not exist, the **rc.lpsched** script (invoked when bringing the system up or down) contains a command line to remove **SCHEDLOCK** first before it attempts to start the scheduler.

Type the command without arguments as follows:

```
# lpsched ♪
```

Note that the command shows no response to let you know that the scheduler is running. To verify that the scheduler is running, use the **lpstat**(1M) command with the **-r** option.

```
# lpstat -r ♪
scheduler is running
```

/usr/lib/lpshut

Two of the three **lpadmin** command options (**-x** and **-p**) cannot be executed unless the LP scheduler is stopped. The **lpshut** command stops the LP scheduler and terminates all printing activity. All requests that were in the middle of printing will be reprinted in their entirety when the scheduler is restarted. Type this command without arguments as follows:

```
# lpshut ↵
scheduler stopped
```

/usr/lib/lpmove

Occasionally, you may find it necessary to move output requests from one destination to another. For example, if you have a printer that was removed for repairs, you will want to move all the pending job requests to a destination with a working printer. This is done using the **lpmove** command. Be aware that job requests routed to a destination without a printer are automatically rejected.

Another use of the **lpmove** command is to move specific requests from one destination to another. When this is done, **lp** will no longer accept requests for the original destination (this is the same effect as a **reject** command). The **lpmove** command does not, however, move requests while the LP scheduler is running. The general format of the **lpmove** command is as follows:

lpmove *requests dest*

where *requests* are the request identification numbers (request IDs) of jobs waiting to be printed, and *dest* is the destination to which the requests are to be moved. The destination can be a printer or a class of printers.

The following examples show how you can use **lpmove**:

Example 1

Move all the requests for printer **lp1** to printer **lp2**. Moving the requests renames the request IDs from **lp1-***nnn* to **lp2-***nnn*. After the requests are moved, **lp** will no longer accept requests for **lp1**. (This has the same effect as a **reject lp1** command issued after the **lpmove**.)

```
# lpmove lp1 lp2 ↵
```

Example 2

Move requests **lp1-11** and **lp2-25** to printer **newlp**:

```
# lpmove lp1-11 lp2-25 newlp ↵
  total of 2 requests moved to newlp
```

/usr/lib/accept

The **accept**(1M) command allows job requests to be placed in a queue at the named destination(s), with the destination being the name of a printer or class of printers. The general format of the **accept** command is as follows:

accept *destination...*

where *destination* is a printer or class of printers. You may specify multiple destinations on the command line.

For example, the following command line allows printer **newlp** to start receiving requests.

```
# /usr/lib/accept newlp ↵
   destination "newlp" now accepting requests
```

/usr/lib/reject

Sometimes it is necessary to stop **lp** from routing requests to a destination. For example, if a printer has been removed for repairs, or if too many requests are building at a destination, you may want to prevent new jobs from being queued at this destination. The **reject**(1M) command performs this function.

Requests in the queue when the **reject** command is invoked will be printed as long as the printer is enabled. After the condition that led to denying requests has been corrected, use the **accept** command to allow the printer to receive requests again. The general format of the **reject** command is as follows:

reject [**−r**[*reason*]] *destinations*

The **−r** option is for telling users why requests are being rejected. The *reason* may be anything you like, but it should be a brief explanation of the purpose for rejecting requests. If the reason consists of more than one word, enclose it in double quotes. The *destinations* are the printers that will no longer accept requests.

Below is an example **reject** command line for a printer, **fastlp**, that is being repaired. While it is out of service you want to prevent **lp** from routing requests to it.

```
# reject −r"printer fastlp under repair" lqp40_1 ↵
  destination "lqp40_1" is no longer accepting requests
```

Users who try to route a job to **fastlp** will receive the following message:

```
$ lp −dfastlp file1 ↵
  lp: can't accept requests for destination "fastlp" -
  printer fastlp under repair
```

Printer Interface Scripts

Printers must have an interface script to work with the DG/UX system. Every print request made with the **lp** command is routed through the appropriate printer interface script before the request is printed on a line printer. Use the **lpadmin −i** command to associate a printer with an interface program. See **lpadmin**(1M).

Model Interface Scripts

Each type of printer requires its own interface script. Several prototype interface scripts, called models, are furnished with the DG/UX system. The model interface scripts are written as shell procedures, but they can be written as C programs or any other executable program. The DG/UX system uses copies of the models. They are located in the **/var/spool/lp/model** directory.

Writing Interface Scripts

If you have a printer that is not supported by one of the model programs, you will have to furnish an interface script for it. The shell script for a "dumb" printer interface script (a model program) is shown in Figure 11-2 at the end of the chapter. This program may be used as a guide if you have to do one of your own.

When the LP scheduler routes an output request to a printer, the interface script for that printer is invoked in the directory **/var/spool/lp** as follows:

interface/*P id user title copies options files ...*

Arguments for the interface program are:

P	Printer name.
id	Request ID returned by lp.
user	User who made the request.
title	Optional title specified by the user.
copies	Number of copies requested by user.
options	Blank-separated list of options specified by user.
files	Full path name of files to be printed.

When the interface program is invoked, its standard input comes from **/dev/null** and both the standard output and standard error output are directed to the printing device. Interface programs format their output based on the command line arguments. Make sure that the interface program has the proper **stty** modes (terminal characteristics such as baud rate and output options). You can do this by adding **stty**(1) command lines of the form:

 # stty *mode options* **<&1**

This command line takes the standard input for the **stty** command from the device. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown below.

```
# stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1 ↵
```

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by **lpsched** as follows:

Code	Meaning to lpsched
0	The print job has completed successfully.
1 to 127	A problem was encountered in printing this particular request (for example, too many nonprintable characters). This problem will not affect future print jobs. The **lpsched** command notifies users by **mail**(1) that there was an error in printing the request.
greater than 127	These codes are reserved for internal use by **lpsched**. Interface programs must not exit with codes in this range.

Dumb Line Printer Interface Program

The interface program in Figure 11-2 is provided as a guide should you need to write one of your own.

```
# Copyright (C) Data General Corporation, 1984 - 1988
# 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.
#
# "@(#)dumb     9.2"
#
#     dumb - Line Printer Interface Script for dumb asynchronous line printer
#
#model# Dumb Asynchronous Line Printer (no baud rate)
#
# PARAMETERS:
#       This interface is called with the following parameters:
#
#       $0 - interface name     - "interface/'printer_name'"
#       $1 - Request id returned by lp
#       $2 - Logname of user making request
#       $3 - Optional title specified by user
#       $4 - Number of copies specified by user
#       $5 - Blank separated list of options - none apply to this interface
#       $6+- Filenames to be printed
#
# EXIT CODES:
#       This interface returns codes that are interpreted by "lpsched"
#       as follows:
#       0       - Print job was completed successfully
#       1 to 64  - A problem code that does not affect future jobs
```

```
#       1- Bad options list
#
#       65 - 127 - If the problem detected affects future jobs, the
#     printer will be disabled. Exit codes are then:
#       65- Can not execute filter - /usr/lib/lptab
#
#       over 127 - Reserved for internal use by lpsched.
#

# check options - none apply to this printer interface
if [ ! -z "$5" ]
then
        exit 1
fi
# check if filter is executable
if [ ! -x /usr/lib/lptab ]
then
        disable -r"Can not execute /usr/lib/lptab" `basename $0`
        exit 65
fi

stty tabs opost onlcr -onlret clocal ixon ff1 cr2 nl0 0<&1

x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
echo "\014\c"
echo "$x\n"
banner "$2"
echo "\n"
user=`grep "^$2:" /etc/passwd | line | cut -d: -f5`
if [ -n "$user" ]
then
        echo "User: $user\n"
else
        echo "\n"
fi
echo "Request id: $1    Printer: `basename $0`\n"
date
echo "\n"
if [ -n "$3" ]
then
        banner $3
fi
copies=$4
echo "\014\c"
shift 5
files="$*"
i=1
while [ $i -le $copies ]
do
        for file in $files
        do
                /usr/lib/lptab < $file 2>&1
                echo "\014\c"
        done
        i=`expr $i + 1`
done
echo "$x"
exit 0
```

Figure 11-2 *Sample LP Interface Program*

End of Chapter

Chapter 12
UUCP Management

This chapter shows you how to setup and manage UNIX-to-UNIX Copy (UUCP). The major sections of this chapter are

- What is UUCP?
- UUCP Setup Overview.
- UUCP Programs, Daemons, and Data Files.
- UUCP Directories.
- UUCP Management Procedures.
- Remedies for Common UUCP Problems. The DG/UX System uses the HoneyDanBer version of UUCP. You can find additional expert information in the section "Expert UUCP Information" at the end of the chapter. For a listing of UUCP error messages, see Appendix C.

What Is UUCP?

We refer to uppercase UUCP and lowercase **uucp**; both originate from UNIX-to-UNIX Copy. Lowercase **uucp** refers to a DG/UX command, **uucp**(1). UUCP refers to a set of programs and data files that allow you to transfer files and to execute remote commands between UNIX systems. Throughout this chapter, we'll mainly be referring to the set of programs; when we refer to the command, we'll append the **(1)** notation to the command name. The functions of UUCP are also used by the **mail**(1) program for remote exchanges.

When we talk about UUCP connections, we mean via a *direct* or a *dial-up* connection. A direct connection is simply a physical wire between machines. A dial-up connection uses telephone lines and a modem at both ends to connect machines.

Understanding the UUCP Components

The UUCP system consists of multiple files and programs. These are briefly discussed here and are more thoroughly discussed in the section "Expert UUCP Information" at the end of the chapter. The five main configuration files, located in **/etc/uucp**, are **Systems**, **Poll**, **Devices**, **Dialers**, and **Permissions**. You can connect only to the remote systems listed in **Systems**. You can do this from the command line for an immediate connection, or you can connect and transfer files automatically at the times set in the **Poll** file. The **uudemon.poll** shell script reads the **Poll** file and initiates connections. Files queued for transfer are exchanged via the modems and devices listed in the **Devices** file. The entries in **Devices** need data from **Dialers**. Finally, the **Permissions** file restricts a remote computer's ability to request and receive files. The default **Permissions** file is set up to provide the maximum amount of security. You can use the **uucheck -v** command to see exactly what your default permissions are. If you wish to change them, see "Permissions File" in the section "Expert UUCP Information" at the end of the chapter.

Before you can put this system to work, you must have a location with which you wish to set up file transfer connections. This means you will have to contact the system administrator of a remote site and exchange certain information: passwords, system NODE names, baud rates, and phone numbers. When you have exchanged this information, you are ready to set up the UUCP files.

UUCP Setup Overview

We recommend that you setup your UUCP facility in the following order:

1. Read the "HoneyDanBer: New UUCP vs Old UUCP" section of this chapter.
2. Start the **uudemon.poll, uudemon.hour, uudemon.admin,** and **uudemon.cleanup** shell scripts.
3. Add devices with the **adddevice** command.
4. Add systems with the **addsystem** command.
5. Edit the **/etc/uucp/Systems** and **/etc/inittab** files for systems running the **uugetty** program.
6. Add poll entries with the **addpoll** command.
7. Test your connections with the **trysystem** command.

HoneyDanBer: New UUCP vs. Old UUCP

There are two versions of UUCP: a new version, referred to as HoneyDanBer, and an old version. The new version includes the following changes to the old version's data files:

- **L-devices** is now **Devices**.
- **L-sys** is now **Systems**.
- **L-dialcodes** is now **Dialcodes**.
- **L.cmds** and **USERFILE** are now merged into a single file called **Permissions**.
- **L.cmds** is now **Devices**.
- The **Poll** and **Sysfiles** files are new.

Functional differences between the two versions of UUCP are reflected in the **/etc/inittab** file. The HoneyDanBer version allows for bidirectional login by respawning **uugetty** instead of **getty**. Bidirectional means that a computer can call *or* receive. So if you have new UUCP on your system, you can connect with other systems running old UUCP. To do this, the **inittab** and **Systems** files must be set up correctly. After using **sysadm addsystem**, you will need to edit some of the data files to reflect what versions of UUCP you'll communicate with. See "Connecting Like and Unlike Versions of UUCP" later in this chapter for more information.

Starting the UUCP Shell Scripts

Your first step in setting up UUCP is to start several important shell scripts. The scripts are discussed below.

uudemon.poll

- Reads the **Poll** file (**/etc/uucp/Poll**) as scheduled.
- If any of the systems in the **Poll** file are scheduled to be polled, places a work file (**C.***file_name*) in the **/var/spool/uucp/***nodename* directory (where *nodename* is the name of the system to be polled).

uudemon.hour

- Calls the **uusched** program to search the spool directories for work files (of the form **C.***file_name*) that have not been processed and schedules these files for transfer to a remote machine.
- Calls the **uuxqt** daemon to search the spool directories for execute files (of the form **X.***file_name*) that have been transferred to your computer and were not processed at the time they were transferred.

uudemon.admin

- Runs the **uustat** command with **−p** and **−q** options. The **−q** reports on the status of work files (**C.***file_name*), data files (**D.***file_name*), and execute files (**X.***file_name*) that are queued. The **−p** prints process information for processes listed in **/var/spool/locks**.

- Sends resulting status information to the **nuucp** administrative login via **mail**(1).

uudemon.cleanup

- Takes log files for individual machines from the **/var/spool/uucp/.Log** directory, merges them, and places them in the **/var/spool/uucp/.Old** directory with other old log information.

- Removes work files seven days old or older, data files seven days old or older, and execute files two days old or older from the spool files.

- Returns mail that cannot be delivered to the sender.

- Mails a summary of the status information gathered during the current day to the **nuucp** administrative login.

To run the scripts on a regular basis, submit them as **cron** jobs. Follow these steps:

1. Log in as **nuucp**. This profile exists so that you can administer and maintain your UUCP environment.

2. Go to a directory where you may create and edit (and later delete) a short file. Use the **crontab** command to produce a listing of any existing **cron** jobs for **nuucp**. In the example below, we call our work file **cronlist**.

   # crontab −l > cronlist ↵

3. Edit the work file, adding lines like the following:

```
1,30 * * * * "/etc/uucp/uudemon.poll > /dev/null"
41,11 * * * * /etc/uucp/uudemon.hour > /dev/null
48 22 * * * /bin/su uucp -c "/etc/uucp/uudemon.admin" > /dev/null
45 23 * *  ulimit 5000; /bin/su uucp -c \
       "/etc/uucp/uudemon.cleanup" > /dev/null 2>&1
```

   These lines represent jobs that you want to run regularly on your system. See the **crontab**(1) manual page to see what these lines mean and how to change them to do what you want.

4. When you have finished editing your work file. check to see if the file **/var/spool/cron/cron.allow** exists. If it does, make sure it contains an entry for **nuucp**. An entry is simply the word **nuucp** on a line by itself. If **cron.allow** does not exist, create it and add the **nuucp** entry.

5. Next, use the **crontab**(1) command to submit the work file you prepared a moment ago:

   # **crontab cronlist** ↵

6. Finally, remove your work file.

Your UUCP shell scripts are started and will run periodically as specified above. To change the schedules you have set, repeat the steps above, changing the schedules when you edit the work file.

Managing UUCP

This section contains methods for administering UUCP with the **sysadm** program. Alternatively, you can manually edit the UUCP data files. For expert information, see the instructions later in this chapter.

When you select **uucpmgmt** from the **sysadm** Main Menu, the following is displayed on your screen:

```
                              UUCP Management

  1 addsystem       Add an entry to the UUCP Systems file
  2 delsystem       Delete an entry from the UUCP Systems file
  3 modsystem       Modify an entry in the UUCP Systems file
  4 lssystem        List entries in the UUCP Systems file
  5 adddevice       Add an entry to the UUCP Devices file
  6 deldevice       Delete an entry from the UUCP Devices file
  7 moddevice       Modify an entry in the UUCP Devices file
  8 lsdevice        List entries in the UUCP Device file
  9 addpoll         Add an entry to the UUCP Poll file
 10 delpoll         Delete an entry from the UUCP Poll file
 11 modpoll         Modify an entry in the UUCP Poll file
 12 lspoll          List entries in the UUCP Poll file
 13 trysystem       Test a UUCP connection

Enter a number, a name, the initial part of a name,
? or <number>? for HELP,  q to QUIT:
```

The following sections discuss the options listed in the UUCP Management menu above.

Adding Entries to the Systems File

This subsection discusses how to add entries to the **Systems** file.

Purpose	To add the name of a remote computer system to **/etc/uucp/Systems**. To add **uugetty** entries to **/etc/inittab**. To edit **/etc/uucp/Systems** to add a "send carriage return" line.
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**addsystem**
Note	If you add a system running **uugetty**, be sure to follow the steps for editing **/etc/inittab** and **/etc/uucp/Systems** at the conclusion of this procedure.
References	**uucp**(1), **uugetty**(1M), "Expert UUCP Information" section

Select option 1, **addsystem**, to add remote system names to your **Systems** file. All systems that you want to connect with must be listed in this file. The **addsystem** command also queries for the name of the remote modem, the tty line, the baud rate, a login name for the remote system, a password, and a telephone number.

If the remote system is connected via a direct line, then you should answer the `Remote Modem Type?` query by entering **direct**. In addition, for direct line connections, **addsystem** makes an entry in the **Devices** file in which the word "direct" is treated like a device name. If you want bidirectional communication on direct line connections running HoneyDanber UUCP, **uugetty** must be running at both ends.

Direct and Dial-up Connections

Let's use **addsystem** for two cases: a direct line connection and a modem connection.

Direct Connects

Below, let's assume that we've spoken with the administrator of a remote system named **dogfish8** and we've obtained all the information we need to add this system to the **Systems** file. We will be running **uugetty**.

An example **addsystem** dialog follows:

```
System Name?  dogfish8 ♪
Remote Modem Type? [hayes] direct♪

TTY Number? 21 ♪

Login Name? uucp ♪
Password? jethro ♪

System entry for dogfish8 has been added.

Do you want to create another system entry? [yes] n ♪
Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Dial-up Connections

Let's use the same system, **dogfish8**, only this time, we'll assume that the connection will be via modem.

```
System Name?  dogfish8 ♪

Remote Modem Type? [hayes]  ♪

Modem Speed? [1200]  ♪
Phone Number? 555-4444 ♪

Login Name? uucp ♪
Password? jethro ♪

System entry for dogfish8 has been added.

Do you want to create another system entry? [yes] n ♪
Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Editing the /etc/uucp/Systems File

You need to add a line to any entry in your **Systems** file that will be running **uugetty**. This sends carriage returns to systems running **uugetty**. Below, we'll edit the modem example entry we just created in **Systems**:

Before:

```
dogfish8  Any ACU 1200 in:--in uucp ssword: jethro
```

After:

```
dogfish8  Any ACU 1200 "" \r\d\r\d\r\d\r in:--in: uucp ssword:jethro
```

The previous script is needed because **uugetty** expects to read a character before outputting the login message.

Adding uugetty to inittab

If you want bidirectional communications with new UUCP (direct or dial-up), you must add **/usr/lib/uugetty** and some options to the *process* field of any tty line you want to use for UUCP communications. You need to make the following changes with an editor: change `/usr/sbin/getty` to `/usr/lib/uugetty -r -t 60`. The following command line completes this change.

```
tt15:23:respawn:/usr/lib/uugetty -r -t 60 tty15 1200
```

The **-r** option causes **uugetty** to wait before sending out the login message. The **-t 60** option sets the timeout delay to sixty seconds.

Connecting Like and Unlike Versions of UUCP

This section shows some example connection setups for various versions of UUCP.

New UUCP to Old UUCP: Direct or Dial-up Connection

In **/etc/inittab**, if New uses **off** and **getty**, Old uses **respawn** and **getty**.

There is no bidirectional communication here. The computer with the tty line turned **off** will have to poll the other computer. In this example, new UUCP would have to call old UUCP. Old UUCP cannot call New UUCP.

New UUCP to New UUCP: Direct Connection

In **/etc/inittab**, both computers *must* have **respawn** and **uugetty** in order to have bidirectional communication. With anything else in **inittab**, communications will not work.

New UUCP to New UUCP: Dial-up Connection

In **/etc/inittab** —

- Both could have **respawn** and **uugetty**. Both would then have bidirectional capability.
- One could have **respawn** and **uugetty**, the other could have **off** and **uugetty**. The one set to **off** might choose this purposely not to have bidirectional capability. That is, the administrator doesn't want to allow other machines to call in.

Deleting Entries in the Systems File

This subsection discusses how to delete entries from the **Systems** file.

Purpose	To remove remote system names from the **Systems** file.
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**delsystem**
References	**inittab**(4), **uucp**(1)

Select option 2, **delsystem**, to delete entries from your **Systems** file. When you stop UUCP communications with a remote system, use the **delsystem** command to remove that system from **/etc/uucp/Systems**. When you select **delsystem**, your interaction with the system proceeds as follows:

```
System Name? mongo5 ♪

System mongo5 had been deleted.

Delete another system? [yes] n ♪

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Modifying Entries in the Systems File

This subsection discusses how to modify entries in the **Systems** file.

Purpose	To change all or parts of the entries of a given **Systems** file entry.
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**modsystem**
References	**inittab**(4), **uucp**(1)

Select option 3, **modsystem**, to change entries in your **Systems** file. Below, let's assume that the modem type and phone number have changed at the remote system connected via modem named **dogfish8**.

An example **modsystem** dialog follows:

```
System Name?  dogfish8 ♪

New System Name? [dogfish8]  ♪

Remote Modem Type? [hayes] att400 ♪

Modem Speed? [1200]  ♪

Phone Number? [555-4444] 555-2222 ♪

Login Name? [uucp] ♪

Password? [jethro] ♪

System entry for dogfish8 has been modified.

Do you want to modify another system entry? [yes] n ♪

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Listing Entries in the Systems File

This subsection discusses how to list entries in the **Systems** file.

Purpose	To display remote system names in the **Systems** file.
Starting Conditions	administrative or multiuser state
sysadm menu	**uucpmgmt**
Commands	**lssystem**
References	**uucp**(1)

To list all of the computer systems that you can communicate with via UUCP, select option 4, the **lssystem** command. An example **lssystem** dialog follows:

```
        System Names? [all] ♪

 Name      Modem   Speed  Phone/Token  Login String
-------    -----   -----  -----------  ------------
dogfish8   hayes   1200     555-4444    in:--in:<uucp> rd:<jethro>
opus       micom   1200     555-0004    in:--in:<uucp> rd:<delray>


  Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

See the section on the **Devices** file in the section "Expert UUCP Information" at the end of the chapter for more information on the **Device** entries.

Adding Entries to the Devices File

This subsection discusses how to add entries to the **Devices** file.

Purpose	To set up the devices that UUCP will use for file transfer connections.
Starting Conditions	administrative state – run level 1
sysadm menu	**uucpmgmt**
Commands	**adddevice**
References	**uucp**(1), **cu**(1)

Select option 5, **adddevice**, to add entries to the **Devices** file to specify which tty lines are to be used and how these lines are to be accessed. You must supply a tty number, a modem name, and a baud rate.

An example **adddevice** dialog follows:

```
TTY number?  15 ♪

Local Modem Type? [hayes]  ♪

Device entry for tty 15 has been added.

Do you want to create another device entry? [yes] n ♪

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

See the section on the **Devices** file in the section "Expert UUCP Information" at the end of the chapter for more information on the **Device** entries.

Deleting Entries From the Devices File

This subsection discusses how to delete entries from the **Devices** file.

Purpose	To remove devices no longer used by UUCP for file transfer connections.
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**deldevice**
References	**inittab**(4), **uucp**(1)

Select option 6, **deldevice**, to remove entries from **/etc/uucp/Devices**. An example **deldevice** dialog follows:

```
TTY Number? 15↵

Device tty 15 has been deleted.

Delete another device? [yes] n↵

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Modifying Entries in the Devices File

This subsection discusses how to modify entries in the **Devices** file.

Purpose	To modify information on the devices in **/etc/uucp/Devices.**
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**moddevice**
References	**uucp**(1)

Select option 7, **moddevice**, to change entries for devices in **/etc/uucp/Devices.** Below, let's change the modem type and speed. An example **moddevice** dialog follows:

```
TTY number?  15 ♪

New TTY number? [15] ♪

Local Modem Type? [hayes] micom ♪

Device entry for tty 15 has been modified.

Do you want to modify another device entry? [yes] n

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Listing Entries in the Devices File

This subsection discusses how to list entries in the **Devices** file.

Purpose	To display all or part of the device entries in the **Devices** file.
Starting Conditions	administrative or multiuser state
sysadm menu	**uucpmgmt**
Commands	**lsdevice**
References	"Expert UUCP Information" section

Select option 8, **lsdevice**, to list entries in the **Device** file. An example **lsdevice** dialog follows:

```
    TTY Numbers [all] ♪

Modem     Type     TTY     Dial TTY     Speed     Parameters
-----     ----     ---     --------     -----     ----------
hayes      ACU      15       --         1200        \D
direct     direct   11       --         9600        \D
penril     ACU      04       17         1200        \D

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Adding Entries to the Poll File

This subsection discusses how to add entries to the **Poll** file.

Purpose	Add times for polling remote systems to the **/etc/uucp/Poll**.
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**addpoll**
References	**uucp**(1), "Expert UUCP Information" section

Select option 9, **addpoll**, to add entries to the **Poll** file. Systems listed in **Poll** can be polled at specified times. This means that at a given hour, a connection is made and any waiting files are transferred. A poll entry consists of the name of the system to be called and the hours when calls are to be attempted.

Below, let's set the polling times for remote system **dogfish8**. Set hours using the numbers 0 to 24. Enter **all** to poll on every hour. An example **addpoll** dialog follows:

```
Polled System Name? dogfish8 ↵

Polling Hours?  2 6 10 14 ↵

Do you want to create another poll entry? [yes] n ↵

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

The polling times set by the above example indicate that **dogfish8** will be polled at 2:00 a.m., 6:00 a.m., 10:00 a.m., and 2:00 p.m daily.

Deleting Entries in the Poll File

This subsection discusses how to delete entries in the **Poll** file.

Purpose	To remove remote system names from **/etc/uucp/Poll.**
Starting Conditions	administrative state
sysadm menu	**uucpmgmt**
Commands	**delpoll**
References	**uucp**(1), "Expert UUCP Information" section

Select option 10, **delpoll,** to delete entries from the **Poll** file. An example **delpoll** dialog follows:

```
Polled System Name?  opus ♪

Poll entry for opus has been deleted.

Delete another poll entry? [yes] n ♪

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Modifying Entries in the Poll File

This subsection discusses how to modify entries in the **Poll** file.

Purpose	To change polling times to remote systems.
Starting Conditions	administrative or multiuser state
sysadm menu	**uucpmgmt**
Commands	**modpoll**
References	**uucp**(1), "Expert UUCP Information" section

Select option 11, **modpoll**, to change the times that remotes systems are called. An example **modpoll** dialog follows:

```
Polled System Name? dogfish8 ♪

New Polled System Name? [dogfish8] ♪

Polling Hours? [2  6  10  14] 1 4 8 ♪

Do you want to modify another poll entry? [yes] n ♪

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

The above example indicates the the polling times for **dogfish8** will be changed to 1:00 a.m., 4:00 a.m., and 8:00 a.m.

Listing Entries in the Poll File

This subsection discusses how to list entries in the **oll** file.

Purpose	To display remote systems and their polling times.
Starting Conditions	administrative or multiuser state
sysadm menu	**uucpmgmt**
Commands	**lspoll**

Select option 12, **lspoll**, to list entries in the **Poll** file. An example **lspoll** dialog follows:

```
Polled System Names? [all] ↵

  Name          Polling Hours
--------       -------------
dogfish8         1  4  8
guss             2  5

Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

The above example indicates that the polling times for the system **dogfish8** are 1:00 a.m., 4:00 a.m., and 8:00 a.m. The polling times for system **guss** are 2:00 a.m. and 5:00 a.m.

Testing a UUCP Connection

This subsection discusses how to test whether a UUCP connection has succeeded or failed.

Purpose	To determine if a UUCP connection has succeeded or failed.
Starting Conditions	administrative or multiuser state
sysadm menu	**uucpmgmt**
Commands	**trysystem**
References	**uutry**(1M), **cu**(1), **uucp**(1)

After you have set up UUCP to communicate with a remote system, select option 13, **trysystem**, to test the connection. The **trysystem** command invokes the **uucico** daemon to start a connection with the system you specify. The **uucico** daemon prints information on the attempted connection. That output will look similar to the following, an example of a failed connection. When you select the **trysystem** command, the system prompts you for the system name. The following example shows what happens when the connection fails.

```
System Name? dogfish8 ↵

./uucico -r1 -sdogfish8 >/tmp/dogfish8  2>&1&
name (dogfish8) not found; return FAIL
_Request (FALSE), _Switch (TRUE), _CallBack (FALSE),
chdir(/var/spool/uucp/dogfish8)

Device Type dogfish8 wanted

Requested Device Type Not Found
Call Failed: NO DEVICES AVAILABLE
Conversation Complete: Status FAILED

Try another system? n ↵
Press the NEWLINE key to see the uucpmgmt menu [?, ^, q]:
```

Output from **trysystem** will vary depending on the situation. Above, we see that device type **dogfish8** was wanted, but not found, so the call failed. Finally, the status of the connection attempt is given as FAILED. To fix this problem, first check your

Devices file and make sure that the line is correct for **dogfish8**. If you have added device entries with **adddevice**, then your entries should be correct. But if you have edited the file manually, you may have introduced an error. If all information is correctly entered, then you may want to verify information with the administrator of **dogfish8**.

UUCP Programs, Daemons, and Data Files

This section discusses the major components of UUCP. See the expert section later in this chapter for more information.

Administrative Programs

The following administrative programs are available for use for your convenience.

If you administer UUCP without **sysadm**, use the **nuucp** login ID because it owns the UUCP files and the spooled data files. The other UUCP login ID is **uucp**, which UUCP systems on remote computers use when they need to transact UUCP business with another machine. Instead of starting a shell like a normal user's login, the **uucp** profile starts the **uucico** program. See Chapter 4 for more information on administrative logins.

uuname — Lists those machines you can contact. This command is in **/usr/bin**. You can do this from the command line or you can use **sysadm lssystem**.

uulog — Displays the contents of the log directories for specified hosts. Log files are created for each remote computer with which your computer communicates. The log files contain records for each use of **uucp**, **uuto**, and **uux**. This command is in **/usr/bin** and is not available through **sysadm**.

uucleanup — Cleans up the spool directory. This command is normally executed from a shell script called **uudemon.cleanup**, which is started by the command **cron**.

uutry — Tests connections between computers. This command displays messages on failed or successful sessions and does a moderate amount of debugging. It invokes the **uucico** daemon to establish a communication link between your computer and the remote computer you specify. You can also do this with the **sysadm trysystem** command.

uucheck — Checks for the presence of UUCP directories, programs, and support files. With a **−v** option, it displays the current permissions for your system. This command is in **/usr/lib/uucp** and is not available through **sysadm**.

User Programs

The user programs for UUCP are in **/usr/bin**. No special permission is needed to use these programs. These commands are described in the *User's Reference for the DG/UX™ System*.

ct — This program instructs your computer to call a modem attached to a terminal over the telephone network. When the modem answers, your computer issues a **getty** (login) process to the modem and allows the terminal user to log in.

The user of the remote terminal may call in to the computer and request that the computer call the remote terminal back. The computer will hang up the initial link to the terminal so that it will be available to answer the call back. This is similar to making a collect call.

cu — This program connects your computer to a remote computer and allows you to be logged in on both computers at the same time. You can execute commands on either computer without dropping the communication link.

uucp — This program lets a user copy files from one computer to another. It creates work files and data files, queues the job for transfer, and calls the **uucico** daemon, which in turn attempts to contact the remote computer.

uuto — This program copies files from one computer to a public spool directory on another computer (**/var/spool/uucppublic/receive**). Unlike **uucp**, which lets you copy a file to any accessible directory on the remote computer, **uuto** places the file in an appropriate spool directory and tells the remote user to pick it up with the **uupick** program.

uupick — This program retrieves the files placed under **/var/spool/uucppublic/receive** when files are transferred to a computer using **uuto**.

uux — This program creates the work, data, and execute files needed to execute commands on a remote computer. The work file contains the same information as work files created by **uucp** and **uuto**. The execute files contain the command string to be executed on the remote computer and a list of the data files. The data files are those files required for the command execution.

uustat — This program displays the status of requested file transfers (**uucp**, **uuto**, or **uux**). It also provides you with a means of controlling queued transfers.

Daemons

Daemons are routines that run as background processes and perform system-wide public functions. These daemons handle file transfers and command executions. They can also be run manually from the shell.

uucico Selects the device used for the link, establishes the link to the remote computer, performs the required login sequence and permission checks, transfers data and execute files, logs results, and notifies the user by mail of transfer completions. When the local **uucico** daemon calls a remote computer, it "talks" to the **uucico** daemon on the remote computer during the session.

The **uucico** daemon is executed by **uucp**, **uuto**, and **uux** programs, after all the required files have been created, to contact the remote computer. It is also executed by the **uusched** and **uutry** programs.

uuxqt Executes remote execution requests. The **uuxqt** daemon searches the spool directory for execute files (always named **X.***file*) that have been sent from a remote computer. When an **X.***file* file is found, **uuxqt** opens it to get the list of data files that are required for the execution. It then checks if the required data files are available and accessible. If the files are present and can be accessed, **uuxqt** checks the **Permissions** file to verify that it has permission to execute the requested command. The **uuxqt** daemon is executed by the **uudemon.hour** shell script, which is started by **cron**.

uusched Schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote computers will be called. The **uusched** daemon is executed by a shell script called **uudemon.hour**, which is started by the **cron** program.

UUCP Data Files

The support files for UUCP are in the **/etc/uucp** directory. You can make all changes to these files with **sysadm uucpmgmt**. Below, we provide details on the structure of these files in case you want to edit them manually. In releases of DG/UX before release 4.0, the names of the various UUCP database files differ from their current names. The description for each file below includes the old name.

Devices Contains information concerning the location and line speed of the automatic call unit, direct links, and network devices. This file was previously called **L-devices**.

Dialers Contains character strings required to negotiate with network devices (automatic calling devices) in the establishment of connections to remote computers (non 801-type dialers). This

new file contains some sample chat scripts. A chat script is a series of modem commands and strings that **uucico** sends to a modem to initiate a connection (or attempt to connect) with a remote system. If your dialer is not already in this file, add it.

Systems Contains information needed by the **uucico** daemon and the **cu** program to establish a link to a remote computer. It contains information such as the name of the remote computer, the name of the connecting device associated with the remote computer, when the computer can be reached, telephone number, login ID, and password. This file was previously called **L-sys**.

Dialcodes Contains dial-code abbreviations that may be used in the phone number field of **Systems** file entries. This file was previously called **L-dialcodes**.

Permissions Defines the level of access that is granted to computers when they attempt to transfer files or remotely execute commands on your computer. Previously, this file was split into **USERFILE** and **L.cmds**.

Poll Defines computers that are to be polled by your system and when they are polled. This is a new file.

Sysfiles Assigns different or multiple files to be used by **uucico** and **cu** as **Systems**, **Devices**, and **Dialers** files. This is a new file.

UUCP Directories

This section lists and describes the directories necessary to run UUCP.

/usr/bin Contains UUCP user programs.

/usr/lib/uucp Contains the executable files for the UUCP system.

/usr/spool/uucp The HOME directory for the **uucp** login.

/etc/uucp Contains the files that make up the UUCP database.

/var/spool/locks Contains lock files for UUCP devices.

/var/spool/uucp Contains directories for administrative purposes and for storing log and status information. The spool directory for queued work that is to be processed by UUCP daemons.

/var/spool/uucppublic
Stores work that has been sent to your computer. The public directory for UUCP.

Remedies for Common UUCP Problems

This section contains remedies for some of the common problems that may prevent UUCP from operating correctly.

Remote system down

Call the remote system's number yourself and listen for the high-pitched tone of the answering modem. If there is none, you know the system or modem is not operating. Call the system administrator of the remote system.

Incorrect login information for remote system

- Dialing sequence: look in the **Systems** and **Dialcodes** files. Consider inserting pauses (commas) in the dialing sequence.
- Login name/password: make sure the login name/password in the local **Systems** file match the login name/password in the remote system's **passwd** file.
- Login sequence: examine the expect/send sequence in the **Systems** file and make sure it uses the correct conventions and login/password strings.
- Remote system name mismatch: make sure the system name in the **Systems** entry matches the nodename of the remote system. On the remote system, use the **uname −n** command to get the nodename.

Modem cannot make connection

- Verify that your modem switches are properly set (refer to modem documentation). Read the **Dialers** file for guidance.
- Make sure your **Dialers** file is set correctly for touch tone or pulse.

Cannot dial in

- Verify that your modem switches are correct.
- Put a phone on the line and call the remote system's modem.
- Make sure **getty** or **uugetty** is up.
- If running **uugetty**, try typing anything or press carriage return.

uucico or cu dies immediately

- Use appropriate debugging switch first. If you get no response or a hangup, make sure **Systems**, **Devices**, **Permissions**, and **Dialers** files are present and readable by **uucp**.
- Make sure **passwd** and **group** files are correct.

Hung modem
Send the reset command (such as ATZ for Hayes modems) to the modem. You can do this with the **cu**(1) command or by turning the modem on and off.

Hung syac
Reload the syac with **/usr/sbin/tcload.**

Modem in wrong state (such as echo mode)
Examine the modem switches. In the case of Hayes-compatible modems, sending ATZ to the modem may fix the problem. See the documentation for your modem. You can find some modem settings in the **Dialers** file.

Getty on line
On DG/UX systems, there should be a **getty** on a dial-in line but not on a dial-out line. For a bi-directional port, **uugetty** must be used.

Wrong line speed
Make sure the modem line speed (baud rate) is compatible with the entry in the **/etc/inittab** file.

Wrong permissions/access
The device file (such as **ttyxx**) should permit reading and writing for any user. Permissions should be set at 666. All files in **/etc/uucp**, **/usr/lib/uucp**, and **/var/spool/uucp** should be owned by **uucp**.

Too many unsuccessful attempts
Remove the status files (files with names matching **/var/spool/uucp/.Status/***system_name*, where *system_name* is the name of the remote system in the UUCP request). Also remove any related lock files (**/var/spool/locks/LCK.***) and start **uucico** yourself to try and complete the job. You may also have to delete any temporary files (/var/spool/uucp/*system_name*/TM*, where *system_name* is the name of the remote system in the UUCP request) that **uucico** might have created during a file transfer.

No daemon
Make sure the **uucp** or **uux** commands start the **uucico** or **uuxqt** daemons without problems.

Permissions file
Make sure this file does not prohibit the desired transfer. Make sure it contains an entry for the **uucp** login name on the remote system.

Bad modem/phone line
Test the modem and cable and make sure they are functioning properly. Test the phone line for noise or interruptions.

Out of file system space
There is not enough space on the remote system to transfer the file. Contact the administrator of the remote system.

Bad pathname
Incorrect or illegal path name. Enter the correct path name.

Expert UUCP Information

This appendix gives further information on some of the topics covered earlier in the chapter. Those topics are:

- UUCP Connections
- UUCP Data Files
- UUCP Spool Files
- Log Files
- UUCP Cleanup

UUCP Connections

Before your computer can communicate with a remote computer, you must set up a two-way communication connection between the machines. This section describes the two kinds of UUCP connections: the direct connection and the dial-up connection.

Direct Connection

The direct connection communication method requires a direct connection from a port on a local computer to a port on the remote computer. A direct line is advantageous when communication is required with the remote computer on a regular basis. The link is always available and access time is short. The disadvantage of the direct link is that the port cannot be used for anything else. The direct connection is made over an RS-232C serial port at transmission rates of up to 19200 bits/second. The recommended length of direct links is 50 feet (around 15.5 meters) or less. Longer lengths can be obtained by using a lower transmission rate and/or limited distance modems at both ends of the link.

Direct connections are beneficial only when:

- It is not possible to link the computers together through a local area network (LAN).
- Two computers transfer large amounts of data on a regular basis.
- Two computers are connected by no more than several hundred feet of cable.

The distance between two directly linked computers is dependent on the environment in which the cable is run. The standard for RS-232 connections is 50 feet (around 15.5 meters) or less with transmission rates as high as 19200 bits per second. As the cable length is increased, noise on the lines may become a problem, which means that the transmission rate must be decreased or limited distance modems be placed on each end of the line.

Do not use more than 1000 feet of cable to connect the two computers or communications will be unreliable. This link should operate comfortably at 9600 bits per second in a clean (noise free) environment. (around 31 meters) of cable,

Dial-up Connection

In this case, the computer that is going to make the connection would call the remote computer using an Automatic Calling Unit (ACU). The remote computer answers via its own ACU and makes the connection. With this arrangement, the ports are not dedicated to only one computer. A dial-up link also requires more hardware (such as the ACU) than the direct connection. Transmission rates are limited to the capacity of the ACUs.

Refer to your modem documentation for information on configuring your modem for dial-out or dial-in use. In **/etc/uucp/Dialers.proto**, you'll find a description for setting up a Hayes modem.

If your modem name is not listed in the **Dialers** file, you will need to edit this file and create a chat script with your modem name as a label. The chat script is the sequence of commands a modem uses for dialing out. Refer to your modem documentation for information on your modem's language and command syntax.

Modem and Direct Link Support Files

Be sure to update the following support files to reflect the presence of a direct link or a modem connection:

- **/etc/uucp/Devices**
- **/etc/inittab**
- **/etc/uucp/Systems.**

Additionally, the **/etc/uucp/Dialers** file must contain information on any modem you use.

When you have determined which communication links best suit your needs, you will need to dedicate one tty line to each communication link you wish to use, unless you run **uugetty** on a line. In this case, the line may be used for both dialing in and dialing out.

UUCP Data Files

UUCP data files must be owned by **nuucp** and must have Read and Write access permissions. The following sections describe the files in **/etc/uucp** that support UUCP file transfers. The files discussed are:

Devices	**Dialers**
Systems	**Dialcodes**
Permissions	**Poll**
Sysfiles	**Maxuuxqts**
Maxuuscheds	**remote.unknown**

Devices File

The **Devices** file (**/etc/uucp/Devices**) contains information for all the devices that may be used to establish a link to a remote computer; these are devices such as automatic call units, direct links, and network connections.

NOTE: This file works closely with the **Dialers, Systems,** and **Dialcodes** files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the **Devices** file has the following format:

Type Line Line2 Class Dialer-Token-Pairs. . .

Each of these fields is defined in the following section.

Type This field may contain one of two keywords (**Direct** or **ACU**), the name of a local area network switch, or a system name.

Direct This keyword indicates a Direct Link to another computer or a switch (for **cu** connections only).

ACU This keyword indicates that the link to a remote computer is made through an automatic call unit (also known as an automatic dial modem). This modem may be connected either directly to your computer or indirectly through a LAN switch.

LAN_Switch This value can be replaced by the name of a LAN switch. Micom and Develcon are the only LAN switches for which there are caller scripts in the **Dialers** file. You can add your own LAN switch entries to the **Dialers** file.

Sys-Name This value indicates a direct link to a particular computer. (*Sys-Name* is replaced by the name of the computer.) This naming scheme is used to convey the fact that the line associated with this **Devices** entry is for a particular computer in the **Systems** file.

The keyword used in the *Type* field is matched against the third field of **Systems** file entries as shown below:

Devices: **ACU** tty11 - 1200 penril

Systems: eagle Any **ACU** 1200 3251 ogin: uucp sword: Oakgrass

Line This field contains the device name of the line (port) associated with the **Devices** entry. For instance, if the ACU for a particular entry was attached to the **/dev/tty11** line, the name entered in this field would be **tty11.**

Line2 If the keyword **ACU** appears in the *Type* field and the ACU is an 801 type dialer, *Line2* would contain the device name of the 801 dialer. (801 type ACUs do not contain a modem. Therefore, a separate modem is required and would be connected to a different line, defined in the *Line* field.) This means that one line would be allocated to the modem and another to the dialer. Since non-801 dialers will not normally use this configuration, the *Line2* field will be ignored by them, but it must still contain a hyphen (-) as a placeholder.

Class If the keyword **ACU** or **Direct** is used in the *Type* field, *Class* may be just the speed of the device. However, it may contain a letter and a speed (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX). The letter is necessary because many larger offices may have more than one type of telephone network: one network may be dedicated to serving only internal office communications while another handles the external communications. In such a case, it becomes necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The keyword used in the *Class* field of the **Devices** file is matched against the fourth field of **Systems** file entries as shown below:

Devices: ACU tty11 - **D1200** penril

Systems: eagle Any ACU **D1200** 3251 ogin: nuucp sword: Oakgrass

Some devices can be used at any speed, so the keyword **Any** may be used in the *Class* field. If **Any** is used, the line will match any speed requested in a **Systems** file entry. If this field is **Any** and the **Systems** file *Class* field is **Any**, the speed defaults to 1200 bps.

Dialer-Token-Pairs

This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a LAN switch, or it may be **direct** for a Direct Link device. You can have any number of *Dialer-Token-Pairs*. The *token* portion may be supplied immediately following the *dialer* portion or if not present, it will be taken from a related entry in the **Systems** file.

This field has the format:

dialer token dialer token

The last pair of tokens may or may not be present, depending on the associated device (dialer). In most cases, the last pair contains only a *dialer* portion. The *token* portion is retrieved from the *Phone* field of the **Systems** file entry.

A valid entry in the *dialer* portion may be defined in the **Dialers** file or may be a special dialer type. The 801 - Bell 801 auto dialer is compiled into the software and is therefore available without having an entry in the **Dialers** file.

```
801 - Bell 801 auto dialer
```

The *Dialer-Token-Pairs* (*DTP*) field may be structured four different ways, depending on the device associated with the entry. Note that any \T or \D escape sequence describes the token *but is not the token*. See below.

1. If an automatic dialing modem is connected directly to a port on your computer, the *DTP* field of the associated **Devices** file entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular **Devices** file entry with an entry in the **Dialers** file. Therefore, the *dialer* field must match the first field of a **Dialers** file entry as shown below:

   Devices: `ACU tty11 - 1200` **ventel**

   Dialers: **ventel** `=&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!`

   Notice that only the *dialer* portion (**ventel**) is present in the *DTP* field of the **Devices** file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the *Phone* field of a **Systems** file entry.

2. If a direct link is established to a particular computer, the *DTP* field of the associated entry would contain the keyword **direct**. This is true for both types of direct link entries, **Direct** and *System-Name* (refer to discussion on the *Type* field).

3. If a computer with which you wish to communicate is on the same local network switch as your computer, your computer must first access the switch and the switch can make the connection to the other computer. In this type of entry, there is only one pair. The *dialer* portion is used to match a **Dialers** file entry as shown below:

   Devices: `develcon tty13 - 1200` **develcon** `\D`

   Dialers: **develcon** `"" "" \pr\ps\c est:\007 \E\D\e \007`

   As shown, the *token* portion is left blank, which indicates that it is retrieved from the **Systems** file. The **Systems** file entry for this particular computer will contain the token in the *Phone* field, which is normally reserved for the phone number of the computer (refer to **Systems** file, *Phone* field). This type of *DTP* contains an escape character (**\D**), which ensures that the contents of the *Phone* field will

not be interpreted as a valid entry in the **Dialcodes** file.

4. If an automatic dialing modem is connected to a switch, your computer must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The *dialer* portion of each pair (fifth and seventh fields of entry) will be used to match entries in the **Dialers** file as shown below:

   Devices: ACU tty14 - 1200 **develcon** vent **ventel**

   Dialers: **develcon** "" "" \pr\ps\c est:\007 \E\D\e \007
   Dialers: **ventel** =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!

   In the first pair, **develcon** is the dialer and **vent** is the token that is passed to the Develcon switch to tell it which device (**ventel** modem) to connect to your computer. This token would be unique for each LAN switch since each switch may be set up differently. Once the **ventel** modem has been connected, the second pair is accessed, where **ventel** is the dialer and the token is retrieved from the **Systems** file.

There are two escape characters that may appear in a *DTP* field:

\T Indicates that the *Phone* (*token*) field should be translated using the **Dialcodes** file. This escape character is normally placed in the **Dialers** file for each caller script associated with an automatic dial modem (**penril**, **ventel**, and so on). Therefore, the translation will not take place until the caller script is accessed.

\D Indicates that the *Phone* (*token*) field should not be translated using the **Dialcodes** file. If no escape character is specified at the end of a **Devices** entry, the **\D** is assumed (default). A **\D** is also used in the **Dialers** file with entries associated with network switches (develcon and micom).

Dialers File

The **Dialers** file (**/etc/uucp/Dialers**) specifies the initial conversation that must take place on a line before it can be made available for transferring data. This conversation is usually a sequence of ASCII strings that are transmitted or expected (called a chat script). A chat script is often used to dial a phone number using an ASCII dialer (such as an automatic dial modem).

As shown earlier, the fifth field in a **Devices** file entry is an index into the **Dialers** file or a special dialer type (801). Here an attempt is made to match the fifth field in the **Devices** file with the first field of each **Dialers** file entry. In addition, each odd numbered **Devices** field (the token field) starting with the seventh position is used as an index into the **Dialers** file. If the match succeeds, the **Dialers** entry is interpreted to perform the dialer negotiations.

Each entry in the **Dialers** file has the following format:

dialer substitutions expect-send ...

The *dialer* field matches the fifth and additional odd numbered fields in the **Devices** file. The *substitutions* field is a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate the equal (=) and hyphen (-) characters into whatever codes the dialer requires for "wait for dialtone" and "pause."

The remaining *expect-send* fields are character strings. Below are some character strings distributed with the **Dialers** file.

```
penril =W-P "" \d > K\c : \EP\T  OK
penril_old =W-P "" \d > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c
OK
ventel =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
hayes  =,-, "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
hayes_att =,-, "" \dAT\r\c OK\r ATDT\T\r\c CONNECT
rixon  =&-% "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadic =K-K "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c LINE
develcon "" "" \pr\ps\c est:\007 \E\D\e \007
micom ""    "" \s\c NAME? \D\r\c GO
direct
att2212c        =W-,    "" \r\c :—: atol2=y,T\T\r\c red
att4000 =,-,    "" \033\r\r\c DEM: \033s0401\c \006 \033s0901\c \
        \006 \033s1001\c \006 \033s1102\c \006 \033dT\T\r\c \006
att2224 =W-,    "" \r\c :—: T\T\r\c red
nls     ""      "" NLPS:000:001:1\M\c
```

Three AT&T modems have entries in the **Dialers** file. The Penril, Micom modem, and Hayes modem scripts have all been confirmed at Data General as have the Micom and Develcon data switches. The other entries have not been tested. If you need to modify the supplied script, refer to your modem documentation.

Table 12-1 lists the meanings of some of the escape characters (those beginning with "\") used in the **Dialers** file.

Table 12-1 Escape Characters Used in the Dialers File

Escape Character	Description
\D	Phone number or token without **Dialcodes** translation.
\E	Enable echo checking (for slow devices).
\T	Phone number or token with **Dialcodes** translation.
\K	Insert a BREAK.
\c	No New Line or carriage return.
\d	Delay (approximately 2 seconds).
\e	Disable echo checking.
\n	Send New Line.
\p	Pause (approximately ¼ to ½ second).
\r	Carriage return.
*****nnn*	Send octal number *nnn*.

Additional escape characters that may be used are listed in the section discussing the **Systems** file.

The Penril entry in the **Dialers** file is executed as follows. First, the phone number argument is translated, replacing any **=** with a **W** (wait for dialtone) and replacing any **-** with a **P** (pause). The handshake given by the remainder of the line works as follows:

"" Wait for nothing. (In other words, proceed to the next thing.)

\d Delay for 2 seconds.

> Wait for a **>**.

K\c Send a **K**. Send no terminating New Line.

: Wait for a **:**.

\EP\T Enable echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, send a **P** and the phone number. The **\T** means take the phone number passed as an argument and apply the **Dialcodes** translation and the modem function translation specified by field 2 of this entry.

OK Waiting for the string **OK**.

Systems File

The **Systems** file (**/etc/uucp/Systems**) contains the information needed by the **uucico** daemon to establish a communication link to a remote computer. Each entry in the file represents a computer that can be called by your computer. In addition, UUCP software can be configured to prevent any computer that does not appear in this file from logging in on your computer. More than one entry may be present for a particular computer. The additional entries represent alternative communication paths that will be tried in sequence.

Using the **Sysfiles** file, you can define several files to be used as "Systems" files. See the description of the **Sysfiles** file later in this appendix for details. Each entry in the **Systems** file has the following format:

System-name Time Type Class Phone Login

Each of these fields is defined in the following section.

System-name
This field contains the host name of the remote computer.

Time This field is a string that indicates the day-of-week and time-of-day when the remote computer can be called. The format of the *Time* field is:

daytime[*;retry*]

The daytime portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa
for individual days

Wk for any weekday (Mo Tu We Th Fr)

Any for any day

Never for a passive arrangement with the remote computer. This means that your computer will never initiate a call to the remote computer. Any call must be initiated by the remote computer. In other words, your computer is in a passive mode with respect to the remote computer (see discussion of **Permissions** file).

Here is an example:

```
Wk1700-0800,Sa,Su
```

This example allows calls from 5:00 p.m. to 8:00 a.m., Monday through Friday, and calls any time Saturday and Sunday. The example would be an effective way to call only when phone rates are low, if immediate transfer is not critical.

The *time* portion should be a range of times such as 0800-1230. If no *time* portion is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, **0800-0600** means all

times are allowed other than times between 6 a.m. and 8 a.m. An optional subfield, *retry*, is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. The subfield separator is a semicolon (;). For example, **Any;9** is interpreted as call at any time, but wait at least 9 minutes before retrying after a failure occurs.

Type This field contains the device type that should be used to establish the communication link to the remote computer. The keyword used in this field is matched against the first field of **Devices** file entries as shown below:

Systems: `eagle Any` **ACU**`,g D1200 3251 ogin: nuucp sword: Oakgrass`

Devices: ACU tty11 - D1200 penril

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol **g** to the device type **ACU**. See the information under the "Protocols" section in the description of the **Devices** file for details.

Class This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion on the **Devices** file, *Class* field). Some devices can be used at any speed, so the keyword **Any** may be used. This field must match the *Class* field in the associated **Devices** file entry as shown below:

Systems: `eagle Any ACU` **D1200** `NY3251 ogin: nuucp sword: Oakgrass`

Devices: ACU tty11 - D1200 penril

If information is not required for this field, use a hyphen (**-**) as a place holder for the field.

Phone This field is used to provide the phone number (token) of the remote computer for automatic dialers or LAN switches. The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the **Dialcodes** file. For example:

Systems: `eagle Any ACU D1200` **NY3251** `ogin: nuucp sword: Oakgrass`

Dialcodes: `NY 9=1212555`

In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause 4 seconds before dialing the next digit.

If your computer is connected to a LAN switch, you may access other computers that are connected to that switch. The **Systems** file entries for these computers will not have a phone number in the *Phone* field. Instead, this field will contain the token that must be passed on to the switch so it will know which computer your computer wishes to communicate with. (This is usually just the system name.) The associated **Devices** file entry should have a

\D at the end of the entry to ensure that this field is not translated using the **Dialcodes** file.

Login This field contains login information given as a series of fields and subfields of the format:

expect send

where *expect* is the string that is received and *send* is the string that is sent when the *expect* string is received.

The *expect* field may be made up of subfields of the form:

expect[*-send-expect*]...

where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string.

For example, with **login--login**, UUCP will expect **login**. If UUCP gets **login,** it will go on to the next field. If it does not get **login**, it will send a null string followed by a New Line, then look for **login** again. If no characters are initially expected from the remote computer, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a New Line unless the *send* string is terminated with a **\c**.

When assembling a send/expect sequence, it is good practice not to specify the first letter of the **login:** or **password:** strings that you want UUCP to expect. The reason for this is that systems in general are inconsistent as regards the case of these first letters--some systems prompt with **login:** while others with **Login:**. Even on a given system, **login:** may appear all lowercase while **Password:** appears with a capital **P**. To avoid problems with capitalization, it is best to specify shortened forms like **ogin:** and **sword:**.

Here is an example of a **Systems** file entry that uses an *expect-send* string:

```
owl Any ACU 1200 5556013 "" \r ogin:-BREAK-ogin: uucpx word: xyz
```

This example says expect nothing, but send a carriage return and wait for **ogin:** (for **Login:**). If you don't get **ogin:**, send a **BREAK**. If you next receive **ogin:**, send the login name **uucpx**. When you receive **word:** (for **Password:**), send the password **xyz**.

There are several escape characters that cause specific actions when they are a part of a string sent during the login sequence. Table 12-2 lists the escape characters that are useful in UUCP communications.

Table 12-2 Escape Characters for UUCP Communications

Escape Character	Description
\b	Send or expect a backspace character.
\c	If at the end of a string, suppress the New Line that is normally sent. Ignored otherwise.
\d	Delay two seconds before sending or reading more characters.
\e	Echo check off.
\n	Send a New Line character.
\p	Pause for approximately ¼ to ½ second.
\r	Send or expect a carriage return.
\s	Send or expect a space character.
\t	Send or expect a tab character.
\E	Start echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.)
\K	Send or expect a break character.
\N	Send or expect a null character (ASCII NUL).
****	Send or expect a \ character.
BREAK	Send or expect a break character.
EOT	Send or expect EOT New Line twice.
\ddd	Collapse the octal digits (*ddd*) into a single character.

Dialcodes File

The **Dialcodes** file (**/etc/uucp/Dialcodes**) contains the dialcode abbreviations that can be used in the *Phone* field of the **Systems** file. Each entry has the format:

abb dial-seq

where *abb* is the abbreviation used in the **Systems** file *Phone* field and *dial-seq* is the dial sequence that is passed to the dialer when that particular **Systems** file entry is accessed.

The entry

```
jt 9=847-
```

would be set up to work with a *Phone* field in the **Systems** file such as `jt7867`. When the entry containing `jt7867` is encountered, the sequence 9=847-7867 would be sent to the dialer if the token in the dialer-token-pair is `\T`.

Permissions File

The **Permissions** file (**/etc/uucp/Permissions**) specifies the permissions that remote computers have with respect to login, file access, and command execution. There are options that restrict the remote computer's ability to request files and its ability to receive files queued by the local site. Another option is available that specifies the commands that a remote site can execute on the local computer. Note that the **Permissions** prototype file sent with this software release is most restrictive.

Permissions File Entries

Each entry is a logical line with physical lines terminated by a \ to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair in the following format:

name=*value*

Note that no white space is allowed within an option assignment.

Comment lines begin with a "#" and they occupy the entire line up to a New Line character. Blank lines are ignored (even within multi-line entries).

There are two types of **Permissions** file entries:

LOGNAME Specifies the permissions that take effect when a remote computer logs in on (calls) your computer.

MACHINE Specifies permissions that take effect when your computer logs in on (calls) a remote computer.

LOGNAME entries will contain a LOGNAME option and MACHINE entries will contain a MACHINE option. See your **/etc/uucp/Permissions.proto** file for more information on entry format.

Considerations

The following items should be considered when using the **Permissions** file to restrict the level of access granted to remote computers:

- Each login IDs used by remote computers to login for UUCP communications must appear in one and only one LOGNAME entry.

- Any site that is called whose name does not appear in a MACHINE entry will have the following default permissions/restrictions: local send and receive requests will be executed, the remote computer can send files to your computer's **/var/spool/uucppublic** directory, and the commands sent by the remote computer for execution on your computer must be one of the default commands; usually **rmail**.

Options

This section describes each option, specifies how each is used, and lists the default values.

REQUEST

When a remote computer calls your computer and requests to receive a file, this request can be granted or denied. The REQUEST option specifies whether the remote computer can request to set up file transfers from your computer. The string

```
REQUEST=yes
```

specifies that the remote computer can request to transfer files from your computer. The string

```
REQUEST=no
```

specifies that the remote computer cannot request to receive files from your computer. This is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry. A note on security: When a remote machine calls you, unless you have a unique login and password for that machine you don't know if the machine is who it says it is.

SENDFILES

When a remote computer calls your computer and completes its work, it may attempt to take work your computer has queued for it. The SENDFILES option specifies whether your computer can send the work queued for the remote computer.

The string

```
SENDFILES=yes
```

specifies that your computer may send the work that is queued for the remote computer as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if your computer is in a "passive mode" with respect to the remote computer.

The string

```
SENDFILES=call
```

specifies that files queued in your computer will be sent only when your computer calls the remote computer. The call value is the default for the SENDFILES option. This option is only significant in LOGNAME entries since MACHINE entries apply when calls are made out to remote computers. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that the **uucico** program can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

The default for both the READ and WRITE options is the **uucppublic** directory as shown in the following strings:

```
READ=/var/spool/uucppublic
WRITE=/var/spool/uucppublic
```

The strings

```
READ=/ WRITE=/
```

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a colon-separated list of path names. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full path name of a file coming in or going out. To grant permission to deposit files in **/usr/news** as well as the public directory, the following

values would be used with the WRITE option:

```
WRITE=/var/spool/uucppublic:/usr/news
```

If the READ and WRITE options are used, all path names must be specified because the path names are not added to the default list. For instance, if the **/usr/news** path name was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably wouldn't want remote computers to be able to write over your **/etc/passwd** file so **/etc** shouldn't be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings

```
READ=/ NOREAD=/etc WRITE=/var/spool/uucppublic
```

would permit reading any file except those in the **/etc** directory (and its subdirectories—remember, these are prefixes) and writing only to the default **/var/spool/uucppublic** directory. NOWRITE works in the same manner as the NOREAD option. The NOREAD and NOWRITE can be used in both LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. There are two examples of when you would use CALLBACK. From a security standpoint, if you call back a machine you can be sure it is the machine it says it is. If you are doing long data transmissions, you can choose the machine that will be billed for the longer call.

The string

```
CALLBACK=yes
```

specifies that your computer must call the remote computer back before any file transfers will take place.

The default for the CALLBACK option is

```
CALLBACK=no
```

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a transmission conversation will never get started.

COMMANDS

The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The **uux** program will generate remote execution requests and queue them to be transferred to the remote computer. Files and a command are sent to the target computer for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote computer can execute on your computer. Note that COMMANDS is not used in a LOGNAME entry; COMMANDS in MACHINE entries define command permissions whether we call the remote system or it calls us.

The string

```
COMMANDS=rmail
```

indicates the default commands that a remote computer can execute on your computer. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:mail:lp
```

overrides the COMMAND default so that the computers owl, raven, hawk, and dove can now execute **rmail**, **mail**, and **lp** on your computer.

In addition to the names as specified above, there can be full path names of commands. For example,

```
COMMANDS=rmail:/usr/local/mail:/usr/bin/lp
```

specifies that command **rmail** uses the default path. The default paths for your computer are **/bin**, **/usr/bin**, and **/usr/local**. When the remote computer specifies **mail** or **/usr/bin/mail** for the command to be executed, **/usr/local/mail** will be executed regardless of the default path. Likewise, **/usr/bin/lp** is the **lp** command that will be executed.

Including the ALL value in the list means that any command from the remote computer(s) specified in the entry will be executed.

CAUTION: *If you use this value, you give the remote computer full access to your computer. BE CAREFUL. This allows far more access than even normal users have.*

The string

```
COMMANDS=/usr/local/mail:ALL:/usr/bin/lp
```

illustrates two points: The ALL value can appear anywhere in the string, and the path names specified for **mail** and **lp** will be used (instead of the default) if the requested command does not contain the full path names for **mail** or **lp**.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like **cat** and **uucp** are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

VALIDATE

The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged computers have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure. (VALIDATE is merely an added level of security on top of the COMMANDS option, though it is a more secure way to open command access than ALL.)

Careful consideration should be given to providing a remote computer with a privileged login and password for UUCP transactions. Giving a remote computer a special login and password with file access and remote execution capability is like giving anyone on that computer a normal login and password on your computer.

The LOGNAME entry

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

specifies that if one of the remote computers that claims to be eagle, owl, or hawk logs in on your computer, it must have used the login **uucpfriend**. As can be seen, if an outsider gets the **uucpfriend** login/password, masquerading is trivial.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? The entry links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote computer is logged in. In fact, it is an asynchronous process with no knowledge of what computer sent the execution request. Therefore, the real question is how does your computer know where the execution files came from?

Each remote computer has its own "spool" directory on your computer. These spool directories have write permission given only to the UUCP programs. The execution files from the remote computer are put in its spool directory after being transferred to your computer. When the **uuxqt** daemon runs, it can use the spool directory name to find the MACHINE entry in the **Permissions** file and get the COMMANDS list, or if the computer name does not appear in the **Permissions** file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/local/mail \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

The value in the COMMANDS option means that remote mail and **/usr/local/mail** can be executed by remote users.

In the first entry, you must make the assumption that when you want to call one of the computers listed, you are really calling either **eagle**, **owl**, or **hawk**. Therefore, any files put into one of the **eagle**, **owl**, or **hawk** spool directories is put there by one of those computers. If a remote computer logs in and says that it is one of these

three computers, its execution files will also be put in the privileged spool directory. You therefore have to validate that the computer has the privileged login **uucpz**.

You may want to specify different option values for the computers your computer calls that are not mentioned in specific MACHINE entries. This may occur when there are many computers calling in, and the command set changes from time to time. The name "OTHER" for the computer name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:mail:/usr/local/Photo:/usr/local/xp
```

All other options available for the MACHINE entry may also be set for the computers that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME Entries

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

share the same REQUEST, READ, and WRITE options. These two entries can be merged as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

Poll File

The **Poll** file (**/etc/uucp/Poll**) contains information for polling remote computers. Each entry in the **Poll** file contains the name of a remote computer to call, followed by a tab character (a space won't work), and the hours the computer should be called. The format of entries in the **Poll** file are:

sys-name hour ...

For example the entry:

```
eagle    0 4 8 12 16 20
```

will provide polling of computer **eagle** every four hours.

The **uudemon.poll** script does not actually perform the poll. It merely sets up a polling work file (always named C.*file*), in the spool directory that will be seen by the scheduler, which is started by **uudemon.hour**.

Sysfiles File

The **/etc/uucp/Sysfiles** file lets you assign different files to be used by **uucp** and **cu** commands as **Systems**, **Devices**, and **Dialers** files. Here are some cases where this optional file may be useful.

- You may want different **Systems** files so requests for login services can be made to different addresses than UUCP services.
- You may want different **Dialers** files to use different handshaking for **cu** and **uucp**.
- You may want to have multiple **Systems**, **Dialers**, and **Devices** files. The **Systems** file in particular may become large, making it more convenient to split it into several smaller files.

The format of the **Sysfiles** file is:

service=*w* **systems=***x:x* **dialers=***y:y* **devices=***z:z*

where:

w is replaced by **uucico**, **cu**, or both separated by a colon

x is one or more files to be used as the **Systems** file, with each file name separated by a colon and read in the order presented

y is one or more files to be used as the **Dialers** file

z is one or more files to be used as the **Devices** file.

Each file is assumed to be relative to the **/usr/lib/uucp** directory, unless a full path is given (note that the configuration files that you can change, like **Devices**, **Systems**, and so on, are actually located in **/etc/uucp**; links in **/usr/lib/uucp** point to them). A backslash-carriage return (\<Newline>) can be used to continue an entry on to the next line.

Here's an example of using a local **Systems** file in addition to the usual **Systems** file:

```
service=uucico:cu   systems=Systems:Local_Systems
```

If this is in **/etc/uucp/Sysfiles**, then both **uucico** and **cu** will first look in **/etc/uucp/Systems**.

When different **Systems** files are defined for **uucico** and **cu** services, your machine will store two different lists of systems. You can print the **uucico** list using the **uuname** command or the **cu** list using the **uuname −c** command.

Maxuuxqts

The **Maxuuxqts** file defines the maximum number of **uuxqt** programs that can run at once. You are limited only by the number of processes you want running on your CPU. The default is 2.

Maxuuscheds

The **Maxuuscheds** file defines the maximum number of **uusched** programs that can run at once. You are limited only by the number of processes you want running on your CPU. The default is 2.

remote.unknown

The **remote.unknown** file is a shell script that executes when a machine that is not in the **Systems** file attempts to start a conversation. It will log the conversation attempt into the file **/var/spool/uucp/.Admin/foreign** and fail to make a connection. If you change the permissions of this file so it cannot execute (**chmod 000 remote.unknown**), your system will accept any conversation requests.

UUCP Spool Files

The files described in this section are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

TM Temporary date file. These data files are created by UUCP processes under the spool directory (i.e., **/var/spool/uucp/***X*) when a file is received from another computer. The directory *X* has the same name as the remote computer that is sending the file. The names of the temporary data files have the format:

TM.*pid.ddd*

where *pid* is a process-ID and *ddd* is a three-digit sequence number starting at 000.

When the entire file is received, the **TM.***pid.ddd* file is moved to the path name specified in the **C.***sysnxxx* file (discussed later in this section) that caused the transmission. If processing is abnormally terminated, the **TM.***pid.ddd* file may remain in the *X* directory. These files will be automatically removed by **uucleanup**.

LCK Lock file. Lock files are created in the **/var/spool/locks** directory for each device in use. Lock files prevent duplicate conversations and

multiple attempts to use the same calling device. The names of lock files have the format:

LCK..*str*

where *str* is either a device or computer name.

These files may remain in the spool directory if the communications link is unexpectedly dropped (usually on computer crashes). The lock files will be ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock.

C.*name* Work file. Work files are created in a spool directory when work (file transfers or remote command executions) has been queued for a remote computer.

The names of work files have the format:

C.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work; the **uucico** code sets this priority and you may change it with **uucp**(1) and **uux**(1). *xxxx* is the four digit job sequence number assigned by **uucp**.

Work files contain the following information:

- Full path name of the file to be sent or requested
- Full path name of the destination or user file name
- User login name
- List of options
- Name of associated data file in the spool directory. If the **uucp −c** or **uuto −p** option was specified, a dummy name (**D.0**) is used
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer

D.*name* Data file. Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

D.*systmxxxxyyy*

where *systm* is the first five characters in the name of the remote computer, and *xxxx* is a four-digit job sequence number assigned by **uucp**. The four digit job sequence number may be followed by a sub-

sequence number, *yyy* that is used when there are several **D.** files created for a work (**C.**) file.

X.*name* Execute file. Execute files are created in the spool directory prior to remote command executions. The names of execute files have the following format:

X.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is a four digit sequence number assigned by **uucp**.

Execute files contain the following information:

- Requester's login and computer name.
- Name of file(s) required for execution.
- Input to be used as the standard input to the command string.
- Computer and file name to receive standard output from the command execution.
- Command string.
- Option lines for return status requests.

Log Files

Log files are created for each remote machine with which your computer communicates. There are directories for each of the **uucico, uucp, uux** and **uuxqt** commands with subdirectories under these for each machine making requests. The logfiles are kept in the directory **/var/spool/uucp/.Log**.

The information from the individual log files for each machine and each program (e.g., machine **dumbo** has a logfile for **uucico** requests and a logfile for **uuxqt** execution requests) can be accessed with the **uulog** program. These files are combined and stored in directory **/var/spool/uucp/.Old** whenever **uudemon.cleanup** is executed. This shell script saves files that are three days old. The three days can be easily modified in the **uudemon.cleanup** shell. If space is a problem, you might consider reducing the number of days the files are kept.

UUCP File Cleanup

The **uustat** program should be invoked regularly to provide information about the status of connections to various machines and the size and age of the queued requests. The **uudemon.admin** shell should be started by **cron** at least once per day to send the administrator the current status. Of particular interest are the the age (in days) of the oldest request in each queue, the number of time a failure has occurred

when attempting to reach that machine, and the reason for the failure. In addition, the age of the oldest execution request (**X.file**) is also given.

Execution files older that a few days can probably be deleted since the only reason they have not been executed is because data files required for execution were not sent. These files are usually sent at the same time as the **X.file**, so the problem is likely at the other end.

The **uucleanup** program, which is run from **uudemon.cleanup** removes these files. Options to **uucleanup** specify the age for sending a warning message to the requester and age for deleting various files. Before deleting, the program tries to figure out what the job was and, if possible, tries to send it to the receiver. If this is not possible, it is returned to the sender.

Public Area Cleanup

To keep the local file system from overflowing when files are sent to the public area, the **uudemon.cleanup** procedure is set up with a **find** command to remove any files that are older than seven days and directories that are empty. The interval may need to be shortened if there is not sufficient space to devote to the public area.

End of Chapter

Chapter 13
Network Management

Information in this chapter is useful only if your system is running TCP/IP, NFS, or the Yellow Pages.

This chapter shows you how to use **sysadm** to add, delete, and modify hosts, networks, and Ethernet addresses after you have already installed your network software. We do not show you how to install, maintain, or use TCP/IP, NFS, or the Yellow Pages in this chapter. For such information, see

- *Setting Up and Managing TCP/IP on the DG/UX™ System*
- *Using TCP/IP on the DG/UX™ System*
- *Managing NFS® and Its Facilities on the DG/UX™ System*

Network Management Terms

We use the following terms in this chapter:

host name

The host name is normally the name of a remote system; it can, however, be a local system. The host name is assigned by that system's administrator. The first character must be a letter. The remaining characters can be any combination of the following characters.

a-z A-Z 0-9 . (period) - (dash)

Names cannot end with a period or dash. Maximum host name length is 255 characters.

host address

The host address is actually the Internet address of a host with which you wish to communicate. This address is composed of four fields separated by periods. Each field contains a decimal number which represents a byte value.

```
000.000.000.000
```

The first field of the Internet address is in the range 1-223, inclusive. The remaining fields are in the range 0-255. The host address is divided into a *network* part and a *host* part depending on the address class. The address class (A, B, or C) is determined by the number in the first field. Remaining fields are arbitrary specifications by the host, i.e., the host part.

Class A address A class A address has a first number in the range 0-127, inclusive. Only the first number (90, below) describes the network address.

90.1.2.3

Class B address A class B address's first field number is in the range 128-191, so that the network address is composed of the first and second fields (128.1, below).

128.1.2.3

Class C address A class C address's first field number is in the range 192-223, so that the network address part is composed of the first, second, and third fields (193.1.2, below).

193.1.2.3

/etc/hosts Contains Internet addresses and host names in the following format: `internet_address hostname`. For example

```
128.223.1.86    sys86
115.1.2.10      sys10
```

/etc/networks This file contains network names and network addresses in the following format: `network_name network_address`. For example,

```
fishnet   128.223
```

The entry `fishnet` is arbitrarily chosen to specify the network address 128.223 (class B). This alias is useful to network programs such as **route**(1M). The alias may also be used in system startup scripts (such as an **rc** script).

/etc/tcpip.params This file contains parameters for various commands (**hostname**(1), **hostid**(1), **security**(1M), **route**(1M), and **ifconfig**(1M)) invoked by the **rc** scripts to initialize the network. You supply these values when you set up TCP/IP with **sysadm setuppackage**. If you add another board or want to update existing entries, you will have to edit the settings in **/etc/tcpip.params**. For more information, see the **tcpip.params**(4) manual page.

/etc/nfs.params This file contains parameters used by **rc** scripts in **/etc/init.d** to initialize the Yellow Pages data base and the NFS processes.

YP master The Yellow Pages (YP) master is the single machine in the YP domain that holds the master **networks** and **hosts** files that are exported to other machines. Global changes can be made only on the YP master machine.

Network Management Procedures

When you select **networkmgmt** from the **sysadm** Main Menu, the following menu is displayed:

```
                Network Management

 1 addhost        Add an entry to the hosts file
 2 delhost        Delete an entry from the hosts file
 3 modhost        Modify an entry in the hosts file
 4 lshost         List entries in the hosts file
 5 addnetwork     Add an entry to the networks file
 6 delnetwork     Delete an entry from the networks file
 7 modnetwork     Modify an entry in the networks file
 8 lsnetwork      List entries in the networks file
 9 addether       Add an entry to the ethers file
10 delether       Delete an entry from the ethers file
11 modether       Modify an entry in the ethers file
12 lsether        List entries in the ethers file
13 nfsparams      Set boot time parameters for NFS and YP
14 tcpipparams    Set boot time parameters for TCP/IP

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, or q to QUIT:
```

You will do most of your network operations with host selections 1, 2, 3, and 4 from the Network Management menu. These selections allow you to make network changes at the same time you are making host changes. You will probably use the remaining sections less frequently, but they are provided for your convenience.

NOTE: The **sysadm** program checks to see if you have NFS loaded. If you do not, your system will not display any of the YP queries that are in the **networkmgmt** procedures.

Adding Hosts

This subsection discusses how to add host names and addresses to the file **/etc/hosts**.

Purpose	To add host names and addresses to **/etc/hosts**.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**addhost**
References	**hosts**(4), **gethostent**(3N)

Select option 1, **addhost**, to add a host name and address to **/etc/hosts**. If the host is the YP master, **addhost** queries you to select either the global or local file for modification. If the network portion of the address is not in **/etc/networks**, then **addhost** requests a unique network name and adds that network name to **/etc/networks**.

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

When you select **addhost**, the system responds as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host/network list.
```

Next, you will be queried for the name of the new host you wish to add and for a host Internet address. Let's assume that name is **sys86** with Internet address 128.223.1.46:

```
Host name?  sys86 ♪
Host address? 128.223.1.46 ♪
YP Server? [yes] ♪
```

We took the default for the YP server question. Enter **n** if the host is a YP client. The dialog ends like this:

```
The entry for sys86 has been added.
Do you want to add another host? [no]
```

You can loop back through this procedure and add another host by typing **y** to the last query. If you accept the default and you are the YP master, the system displays

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

Deleting Hosts

This subsection discusses how to delete host names from the **/etc/hosts** file.

Purpose	To delete a host name from **/etc/hosts.**
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**delhost**
References	**hosts**(4), **gethostent**(3N)

Select option 2, **delhost**, to delete an entry from **/etc/hosts.**

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

When you select **delhost**, the system responds as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

You may use **lshost** to see the names of existing hosts.

```
Host name?  sys00 ♪

The entry for sys00 has been deleted.

Do you want to delete another entry? [no]
```

If you are the YP master, when you are finished, the system displays:

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

Modifying a Host Entry

This subsection discusses how to change the name and address of a host.

Purpose	To change the name and/or address of a host.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**modhost**
References	**hosts**(4), **gethostent**(3N)

Select option 3, **modhost**, to change an entry in the **hosts** file. If your host is the YP master, the **modhost** command asks you to select the global or local file. If you are not the YP master, you can access only the local **hosts** file. Then **modhost** prompts for the host name whose entry you wish to change. After you enter the name of an existing host, **modhost** allows you to change the host name or the corresponding Internet address. In either case, the new value must be either the same as the old or unique. The file **/etc/networks** is not changed. To add a **hosts** entry, use **sysadm addhost**.

When you select **modhost** on a YP master, the system responds as follows:

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

Access the Global Host/Network List? [yes]
```

The system responds as follows if you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

The dialog continues like this:

```
Host name?  sys86 ♪
New host name?  sys00 ♪
Host address? [128.223.1.46] ♪
YP Server? [yes] ♪
The entry for sys86 has been modified.
Do you wish to modify another host? [no] ♪
```

Above, we changed **sys86** to **sys00**. We pressed New Line and kept the old (default) host Internet address. When you finish modifying entries on a YP master, you see:

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

Listing Hosts

This subsection discusses how to display the contents of **/etc/hosts.**

Purpose	To display the contents of **/etc/hosts.**
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**lshost**
References	**hosts**(4), **gethostent**(3N)

Select option 4, **lshost**, to list your **/etc/hosts** entries. **Sysadm** will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

When you select **lshost**, the system respond as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host list.
```

Assuming you are not the YP master, your system displays

```
Host Names(s)? [all] ↵

Host            Address
------          --------
localhost       127.1
sys86           128.223.1.46
sys10           128.223.1.10
```

Adding Networks

This section tells how to add an entry to the **/etc/networks** file.

Purpose	To add a network name and address to **/etc/networks.**
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**addnetwork**
References	**networks**(4)

Select option 5, **addnetwork**, to add an entry to your **/etc/networks** file. If you are the YP master, **addnetwork** asks if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

When you select **addnetwork** on the YP master host, the system responds as follows:

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

Access the Global Host/Network List? [yes]
```

If you are not on the YP master host, **addnetwork** responds like this:

```
This host is not the YP master.  You can only access
the local host list.
```

Next, **addnetwork** queries you for the name of the new network to be added. This name must not already exist in **/etc/networks**. The interaction proceeds:

```
Network name?  newnet ♪

Network Address? 192.114.25 ♪

The entry for newnet has been added.
Do you want to add another network? [no] ♪
```

If you are the YP master, when you are finished, the system displays:

```
Updating the YP Yellow Pages password and group maps.
```

Deleting Network Names

This section tells how to delete an entry from the **/etc/networks** file.

Purpose	To delete a network name from **/etc/networks**.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**delnetwork**
References	**networks**(4)

Select option 6 to execute the **delnetwork** command. If the host is the YP master, **delnetwork** asks you to select either the global or local file for modification. The **delnetwork** command requests the network name; if the network name exists, it is deleted from **/etc/networks**.

Sysadm will determine if you are the YP master. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files.

If you are the YP master, the **delnetwork** dialog proceeds like this:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host/network list.
```

You may use **lsnetwork** to see the names of existing network names.

```
Network name?  newnet-24 ♪

The entry for newnet-24 has been deleted.

Do you wish to delete another entry? [no]
```

If you are the YP master, when you are finished, the system displays:

```
Updating YP Yellow Pages password and group maps.
```

Modifying Networks

This subsection discusses how to change the name and address of a network.

Purpose	To change the name and/or address of a network.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**modnetwork**
References	**networks**(4)

Select option 7 to execute the **modnetwork** command. If you are the YP master, you will be asked if you want to access the global hosts/network lists. If you are not the YP master, you can only access local hosts and networks files. When you select **modnetwork**, the system responds as follows:

If you are the YP master:

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

Access the Global Host/Network List? [yes]
```

If you are not the YP master:

```
This host is not the YP master.  You can only access
the local host/network list.
```

Next, specify the name of the network to be modified. This name must already exist in **/etc/networks**. List the contents of **/etc/networks** with **lsnetwork**.

```
Network name?  newnet ♪
New Network name? [newnet]  newnet-24 ♪

Network Address? [192.114.25] ♪

The entry for newnet has been modified.
Do you want to modify another network? [no]
```

The network name you supply determines the defaults for the new name and address queries. We changed **newnet** to **newnet-24**. We left the network address unchanged. If you are the YP master, when you are finished modifying a network name, the system displays:

```
Updating the YP Yellow Pages password and group maps.
```

Listing Networks

This subsection discusses how to display the contents of the file **/etc/networks**.

Purpose	To display the contents of **/etc/networks**.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**lsnetwork**
References	**networks**(4)

Select option 8 to execute the **lsnetwork** command. If you are the YP master, **lsnetwork** asks you to select either the global or local file for listing. The entries from the hosts file are listed.

```
Access the Global Host/Network List? [yes] ♪

Network Name(s)? [all] ♪

Network          Address
-------          --------
fishnet          128.223
butterflynet     89.3
satnet           128.5
newnet-24        192.114.25
```

Adding an Entry to /etc/ethers

This subsection discusses how to add an entry to the file **/etc/ethers**.

Purpose	To add an entry to the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**addether**
References	**ethers**(4), *Setting Up and Managing TCP/IP on the DG/UX™ System*

Select option 9 to execute the **addether** command. A server machine that will be providing operating system service to a diskless client needs the client's Ethernet address so that the client can boot over the network. The netboot sequence is explained in Chapter 1. An Ethernet address is a unique sequence of hexadecimal numbers (like 08:00:1b:23:91:a6) that identify one Ethernet local area network (LAN) controller device. The Ethernet address is set at the factory for each controller. A workstation's Ethernet address appears at the console display when you turn on power to the system. Hosts on an Ethernet network need to know each other's Ethernet address in order to send requests and information within the network.

The **addether** function maps Ethernet addresses to client host names. Let's assume that we are adding a diskless client, **dg2**, whose Ethernet address is 08:00:1b:30:f3:21. We begin by typing:

```
# sysadm addether ↵
```

The system responds as follows:

```
Host Name? dg2 ↵
Ethernet Address? 08:00:1b:30:f3:21 ↵
The entry for dg2 has been added.
Do you want to add another entry? [yes] no↵
```

Deleting an Entry from /etc/ethers

This subsection discusses how to delete an entry from the file **/etc/ethers**.

Purpose	To delete an entry from the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**delether**
References	**ethers**(4), *Setting Up and Managing TCP/IP on the DG/UX™ System*

Select option 10 to execute the **delether** command. The **delether** function removes entries from the file **/etc/ethers**. Let's remove the entry for the diskless client **dg2**. We begin by typing:

```
# sysadm delether ↵
```

An example **delether** dialog follows.

```
Host Name? dg2 ↵
The entry for dg2 has been deleted.
Do you want to delete another entry? [yes] no ↵
```

Modifying an Entry in /etc/ethers

This subsection discusses how to modify an entry in the file **/etc/ethers**.

Purpose	To modify an entry in the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**modether**
References	**ethers**(4), *Setting Up and Managing TCP/IP on the DG/UX™ System*

Select option 11 to execute the **modether** command. If you need to change entries in the **/etc/ethers** file, use the **modether** function. If we were to change the system board in our **dg2** diskless client, for example, we would have to change the Ethernet address in the **dg2** client's **ethers** entry. (This is because the client's network controller chip, **inen**, which holds the Ethernet address, is a part of the system board. Thus, changing a workstation's system board also changes its Ethernet address.) To change **dg2**'s **ethers** entry, we do this:

```
# sysadm modether ↵
```

The **modether** dialog proceeds as follows:

```
Host Name? dg2 ↵

Ethernet Address? [08:00:1b:30:f3:21] 08:00:1b:30:22:a2 ↵

Do you want to modify another ethers entry? [yes]
```

Listing Entries in /etc/ethers

This subsection discusses how to display the entries in the file **/etc/ethers**.

Purpose	To display the entries in the **/etc/ethers** file.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**lsether**
References	**ethers**(4), *Setting Up and Managing TCP/IP on the DG/UX™ System*

Select option 12 to execute the **lsether** command. If you need to to know the Ethernet address of the hosts on your servnet, use the **lsether** function. This function produces the following:

```
Host            Ethernet Address
----            ----------------
dg1             08:00:1b:41:b5:11
dg2             08:00:1b:30:22:a2
```

Setting NFS and YP Parameters

This subsection discusses how to edit the parameters in the file **/etc/nfs.params**.

Purpose	To edit the parameters in **/etc/nfs.params**.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**nfsparams**
References	**nfs.params**(4), *Managing NFS® and its Facilities on the DG/UX™ System*

Select option 13 to execute the **nfsparams** command. The **nfsparams** function edits the **/etc/nfs.params** file. This file defines the configuration parameters for the Network File System (NFS) and the Yellow Pages (YP). The **nfsparams** command only works on the domain name and the host type. You must make other changes manually.

The **nfsparams** function displays the current values as defaults. We use `accounts`, below, as an example domain name.

When you select the **nfsparams**, you continue as in this example:

```
Domain Name? [accounts] ↵

Yellow Pages Host Type? [server] ↵

Do you want to use these settings? [yes] ↵
```

We selected the defaults above. Simply type in any changes you want at each query.

Setting TCP/IP Parameters

This subsection discusses how to edit the file **/etc/tcpip.params**.

Purpose	To edit the parameters in **/etc/tcpip.params**.
Starting Conditions	administrative or multiuser mode
sysadm menu	**networkmgmt**
Commands	**tcpipparams**
References	**tcpip.params**(4), *Using TCP/IP on the DG/UX™ System*

Select option 14 to execute the **tcpipparams** command. The **tcpipparams** function allows you to change the **hostname** and **hostid** parameters in the file **/etc/tcpip.params**. This file defines the TCP/IP parameters needed to configure the network at boot time. In addition, this function queries you to include or exclude each interface in your list of network interfaces (**hken0**, **inen0**, and so on). To include an interface means that it will be part of the configuration. When you include an interface, you will be asked to define a host name, a broadcast address, and a netmask for the interface. Below, let's assume that there are two entries in your **system.tcpip** file: a network board (**hken**) and a pseudo-device (**loopback**). Both of these must already have entries in **/etc/hosts**; those entries in this case could look like:

```
128.223.1.86    sys86
127.0.0.0       localhost
```

The **tcpiparams** dialog proceeds like this:

```
Local Host Name? sys86 ↵
```

Next, **tcpipparams** will go through all the TCP/IP interfaces listed in **/dev**. You will be asked if you want each interface to be configured at boot time. The example below shows the dialog for configuring interfaces for **hken0** and **loop0**. Note that **loop0** does not use the netmask or the broadcast address, so we just take the defaults for the **loop0** netmask query and the **loop0** broadcast address query.

```
Configure hken0? [yes] ↵
Host Name for hken0? [sys86] ↵

Netmask for hken0? [0xffff0000] ↵
Use 1-bits in Broadcast Address for hken0 [yes] no ↵

Configure loop0? [yes] ↵
Hostname for loop0? [localhost] ↵
```

```
Netmask for loop0? [0xffff0000] ♪
Use 1-bits in Broadcast Address for loop0 [yes] ♪

Do you want to use these settings? [yes] ♪
These settings have been placed in /etc/tcpip.params.
They will take effect the next time you boot DG/UX.
```

If there are other device entries, the function will start over; otherwise, the functions exits.

End of Chapter

Chapter 14
User Account Management

This chapter is for systems supporting one or more user accounts. As a system manager, you will set up and manage the everyday working environment in which users function. This task includes adding and removing user accounts, creating news aliases and groups, maintaining system security, answering users' questions, and helping users with system problems. If your system is part of a network that uses the Yellow Pages facility, some of these tasks will be the responsibility of the master server administrator. For more information on the Yellow Pages facility, see *Managing NFS® and its Facilities on the DG/UX™ System*.

The major sections of this chapter are:

- User Account Terms
- Request for User Login
- About User Accounts
- Security Suggestions
- User Account Procedures
- The User's Environment
- Expert User Management Information

User Account Terms

Read the following definitions before beginning the procedures in this chapter.

login name — A valid name for logging on to the system. Also known as *username*. A login name can be up to eight alphabetic or numeric characters; the first character must be alphabetic.

password — A unique string of alphabetic or numeric characters that allows a user access to the system. A password must be at least six characters, and a maximum of eight characters. At least one character *must* be a numeral or a special character. You may set explicit passwords or leave a password field open when creating a user account with **adduser**. If the field is open, then users are prompted to set a password when they first login.

password aging — A system which forces users to set new passwords within a specified number of weeks. An "aged" password is one that must be changed within the specified number of weeks. When this time period lapses, the password will no longer gain a user entry to the system. The user must choose a new password. System administrators decide if they want to use password aging or not. See "Expert User Management Information" for details.

group — A set of users with access privileges to the same set of files based on group ID numbers. Also known as a *group name*. People that need access to the same files can be listed in a group. For example, anyone in group "prog" could access those files associated with that group name. Other people in, for instance, group "pool" would not have access to the files of group "prog".

alias — A mailing list. An alias contains one or more login usernames. If you address mail to an alias, the mail is delivered to all users listed in that alias. An alias entry consists of an alias name and a list of alias members.

user ID — A unique number that identifies a user to the system. The user ID number (**uid**) is between 100 and 60,000. The number must not include a comma. Superuser (root) uses 0. System ID numbers are 1 to 99. The **sysadm** program supplies new ID numbers by default.

group ID — A unique number that identifies a group to the system. The same conditions apply to the group ID (**gid**) number as to the **uid**. The **sysadm** program supplies new ID numbers by default. System ID numbers are 1 to 99.

home directory — The origination point of the user's directory tree. The home directory is where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory such as **/mach_2/poulet**.

initial program The program invoked at the time the user logs in. Choices include **sh**, **csh**, or some other local executable program. Users typically select their own initial program.

YP master server The YP master server is the single machine in the network that holds the master networks and hosts files that are exported to other machines. Global changes can be made only on the master machine. If your system has the Network File System (NFS), and the current machine is the YP master server, you will be asked to choose local or global options in queries for **usermgmt** commands in this chapter.

Request for User Login

You may find it helpful to use a standard form for people who wish to become users on your system. The following is a suggested form for collecting information on new users.

```
**********************************************************

                    REQUEST FOR USER LOGIN

1. Full name _____________________________________________

2. Login name ____________________________________________

3. Login group ___________________________________________

4. Other groups __________________________________________

5. Login directory _______________________________________

6. Initial program (shell): sh  csh  other _____________

7. Mail aliases __________________________________________

**********************************************************
```

Figure 14-1 *User Login Request Form*

About User Accounts

Because of some preset defaults, you can begin adding users to the system immediately. Using **sysadm userdefaults**, you can change the defaults for passwords, groups, and initial programs. Use **modalias** to change alias defaults. The following list shows how defaults are originally set on the DG/UX system:

- Password aging -- Set to **off**.
- Default group -- Set to **general**.
- Default initial program -- Set to **/bin/sh**.
- Default mail aliases -- Set to **everybody**.
- Default parent directory -- Not set. You *must* set this value.

When you are ready, read the section "The User's Environment" later in this chapter. In it, we present information on global and local user profiles, environment variables, file creation permissions, default and restricted shells, an electronic bulletin board, and system mail. In addition, we offer suggestions for tracking user problems and provide a sample Trouble Log for users to fill out.

Using Groups and Aliases

Let's say that users on your system are divided into two categories: programmers and data entry people. Initially, you can assign everybody to the default group that comes with the DG/UX system. Members of group **general** are allowed access to all directories and files owned by **general**. This is a shared ownership. Later, you can assign users to additional groups. For instance, you might put programmers in group **prog** and data entry people in group **pool**. Note that group **general** is simply provided as a default to speed up the process of adding users. You can rename it or delete it. Later, you can create aliases and groups based on tasks, projects, or whatever you choose.

Aliases are simply mailing lists used by the **mailx**(1) command. The default is **everybody**. See "Adding Mail Aliases," later in this chapter, for details.

Specifying a Parent Directory and Initial Program

We do not provide a default parent directory because we can't be sure of how you've laid out your logical disks and file systems when you installed the DG/UX system. You will have to indicate the name of the file system that you wish to make the default parent directory for users.

If you have or plan to have NFS in the future, you should make sure that parent directories have unique names across machines. One way to do this is by using the hostname as the parent directory name. This practice makes it easy to identify the origin and ownership of file systems in your network.

The default initial program is traditionally specified as **/bin/sh**, but it can be any executable local shell program.

Security Suggestions

The following are suggestions for maintaining a secure system.

- Set the access permissions to directories and files to allow only the necessary permissions for owner, group, and others.
- All logins should have passwords. Advise users to change passwords regularly. Advise users not to pick obvious passwords. Password aging is an option.
- All dial-up ports should have passwords. Any system with dial-up ports is not really secure.
- Users who make frequent use of the **su** command can compromise the security of your system by accessing files belonging to other users without the other users' knowledge. The more people who know a given login and password, the less secure access is to the system. For this reason, a log is kept on the use of the command. Check the file **/usr/adm/sulog** to monitor use of the **su** command.
- Login directories, **.profile** files, and files in **/bin**, **/usr/bin**, and **/etc** should not be writable by others.
- Encrypt sensitive data files. The **crypt**(1) command together with the encryption capabilities of the editors (**ed** and **vi**) provide protection for sensitive information. The encryption/decryption capabilities mentioned above are available with U.S. releases of the DG/UX system only.
- Do not leave a logged-in terminal unattended, especially if you are logged in as **sysadm** or **root**.
- Use the **sysadm filescan** command to look for files that result from and can lead to breaches in security (specifically, files owned by the superuser that have the **setuid** bit set).

User Account Procedures

The following procedures are covered in this section:

- Setting user defaults
- Adding, deleting, modifying, or listing user accounts
- Adding, deleting, modifying, or listing groups
- Adding, deleting, modifying, or listing aliases

When you select **usermgmt** from the **sysadm** Main Menu, the following is displayed on your screen:

```
                User Management

 1 userdefaults   Set user account defaults
 2 adduser        Create a user account
 3 deluser        Delete a user account
 4 moduser        Modify a user account
 5 lsuser         List user account information
 6 addgroup       Add group entries
 7 delgroup       Delete group entries
 8 modgroup       Modify group entries
 9 lsgroup        List group entries
10 addalias       Add mail alias entries
11 delalias       Delete mail alias entries
12 modalias       Modify mail alias entries
13 lsalias        List mail alias entries

Enter a number, a name, the initial part of a name,
? or <number>? for HELP, ^ to GO BACK, or q to QUIT:
```

Setting User Defaults

This subsection discusses how to set defaults for group name, parent directory, initial program, and password aging.

Purpose	To set defaults for group name, parent directory, initial program, and password aging.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**userdefaults**
Note	The DG/UX system is shipped with many defaults already set. Examine the entries in **userdefaults** to make sure they are defaults you want.
References	**group**(4), **sh**(1), **csh**(1)

The **userdefaults** command, option 1, sets the defaults that the **adduser** command uses. If you set defaults first, **adduser** is much easier to use. With the **userdefaults** command, you can set defaults for group names, parent directories for home directories, initial programs (shell), and password aging.

If you're running NFS, **sysadm** will determine if your machine is the YP master server. If it is, you will be asked if you want to access the global or local user lists. If your machine is not the YP master server, you can only access local user lists.

When you select **userdefaults**, the dialog proceeds as follows:

```
Enable Password Aging? [no]  y ↵
```

Enter **yes** if you want to place time limits on user passwords. Enter **no** if you do not want to use password aging. If you use password aging, you will be asked to define a minimum and maximum age in weeks for passwords. Users will be forced to change passwords when the maximum time expires. Users will not be allowed to change a password until the minimum time expires. If you use password aging, the limits you set will apply to all users added later. Above, we chose to use password aging by answering **yes**. Next, the system asks for a maximum and minimum age. These numbers, which represent weeks, must be between 0 and 63, inclusive.

```
Maximum Password Age? 8 ↵

Minimum Password Age? 1 ↵
```

After password aging, you can set group defaults:

```
Group Name? [general]  ↵

Parent directory of login directory? /mach_2 ↵

Initial Program? [/bin/sh]   ↵

Press the NEWLINE key to see the usermgmt menu [?, ^, q]:
```

When you are satisfied with the defaults, you are ready to add users to the system.

Adding User Accounts

This subsection discusses how to add user accounts.

Purpose	To add one or more user accounts.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**adduser**
References	**passwd**(4), **passwd**(1), **group**(4), **newgrp**(1)

Select option 2, **adduser**, to add a user profile. For each user account to be added, **sysadm** requests the user's full name, the login name, a user ID, a login group name, the parent directory of the user's home directory, and the initial program. As an example, let's add user L.Q. Poulet to our system. If you select the **adduser** command, you enter into a dialog like this:

```
User Login Name? poulet ↵

Full User Name? L.Q. Poulet ↵
```

Next, you are queried for the user ID. This is a number the DG/UX system uses to associate files with a given login name. Each user has only one user ID. User ID 0 is reserved for profiles with superuser status. Numbers between 1 and 99 are reserved for other system logins, like **lp**. Regular user IDs are between 100 and 60,000. You may select a default ID number by responding with a New Line; this gives you the next unused ID number.

```
User ID? [101]  ↵
```

The next query asks for the group to which you want Poulet to belong. You can type the name of an existing group, or you can answer with the default, putting Poulet into group **general**. If you are just setting up users for the first time, you may wish to add them to the default group for now. Later, you can add users to specific groups as needed. For now, let's put user Poulet into the default group **general**:

```
Group Name? [general] ↵

Parent directory of login directory? [/mach_2] ↵
```

If you do not want Poulet to go into the default directory, enter a full path name of the parent directory where Poulet's login directory will be created. The login directory has the same name as the user. For example, if you choose **/mach_2** as the parent directory, then Poulet's login directory will be **/mach_2/poulet**. You can override the default here, but if you type the name of a directory that does not exist,

such as **/messages**, the system will respond:

```
Warning: The directory /messages does not exist.
Do you wish to create /messages? [yes]
```

You can create the new parent directory by answering **y** to the last query. But let's assume you selected the default parent directory. When user Poulet logs in, his home directory will be **/mach_2/poulet**.

Now that Poulet has a home directory, he needs to be assigned an initial shell program. Users will generally request a certain shell, usually **csh** or **sh**. We'll assume that user Poulet has asked for the **csh**. See **sh**(1) and **csh**(1) for details on both of these programs.

The system asks for the initial program that you want to assign to user Poulet. Below, we'll give Poulet the C shell. Poulet's local profile is also determined when you set his initial program. See "The User's Environment" later in this chapter for more information.

```
Initial Program? [/bin/sh] /bin/csh ↵
```

The next query determines whether or not Poulet will be able to log in with a password.

```
The password is currently clear.
Password Operation? [set] ↵
Password?
```

Responses to the Password Operation query are **set** or **disable**. With **set** you can type in a password for the new user, or you can simply press New Line and no password is assigned. If you don't assign the user a password here, then the user will be instructed to enter a password at the first log on. Selecting **disable** means that this user cannot log into the system.

The **adduser** command now gives you a chance to do one of the following with the entries for username Poulet:

```
Do you want to edit, skip, or install this user entry?
[install]  ↵

User poulet has been added.
```

None of the entries in **adduser** are recorded until you **install** them. We **installed** this user entry by pressing New Line. The choices are:

edit Use current values as default and start over.

skip Discard last values entered and quit.

install Add this user to the system.

If you choose, you can add another user by answering **y** to the next query. If you're finished, type **n** or **no** and you will exit from **adduser** and return to the User Management Menu.

```
Do you want to add another user? [yes] n ↵

Press the NEWLINE key to see the usermgmt menu [?, ^, q]:
```

Poulet now has a user account on your system. The first time Poulet logs on to the system, he will be prompted to set his password. **Sysadm** invokes the **passwd**(1) program. All Poulet has to do is type in a password of his choice, say **rover8**. Now, every time Poulet logs on, he will not be allowed access to the system unless he types **rover8**.

Deleting User Accounts

This subsection discusses how to delete entries from the user account list.

Purpose	To delete one or more entries from the user account list.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**deluser**
References	**passwd**(1), **group**(4)

The **deluser** command, option 3, removes one or more user accounts from the system. When a user account is removed, the home directory, the mailbox, group entries, and mail alias entries are deleted for that user. To delete L.Q. Poulet's account, select the **deluser** command.

```
User Login Name?  poulet
Do you really want to delete poulet? [no] y

User poulet 101 has been deleted.
Delete the Home Directory (/mach_2/poulet)? [no]  ♪

Do you want to delete another user? [yes] n ♪
```

We retained Poulet's home directory by pressing New Line for the default. We did this in case other users need access to this directory. So, in the above case, the user is deleted, but his directory remains. If you run the **ls −al** command on this directory, you will no longer see Poulet printed as owner; Poulet's ID number 101 will be listed instead. Delete this directory by hand when you are finished with it.

Remember, all group entries for login name poulet have also been deleted.

Modifying User Accounts

This subsection discusses how to modify user account entries.

Purpose	To modify user account entries.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**moduser**
References	**passwd**(4)

The **moduser** command, option 4, uses the same queries as the **adduser** selection. Default values are whatever entries were made with **adduser** or subsequently modified with **moduser**. Let's say we wanted to modify some of the entries for user Poulet. We select the **moduser** command, and the interaction proceeds as follows:

```
User Login Name? poulet ♪
New Login Name? [poulet] ♪
Full User Name? [L. Q. Poulet]  ♪


User ID? [101] ♪

Group Name? [general] ♪
Parent directory of login directory? [/mach_2] /mach_2/test ♪
```

We changed Poulet's login directory from **/mach_2** to **/mach_2/test.**

```
Initial program? [/bin/csh] /bin/sh ♪
```

We changed Poulet's initial login program from **csh** to **sh.** Next, we can modify a user's password operation:

```
Password Operation? [keep]  ♪
```

We took the default, which retains the user's password operation exactly as it was. We could have selected **disable** to prevent this user from logging on if we wanted, or we could have used **set** to change or delete the password.

```
Do you want to edit, skip, or install this entry? [install] ♪
Do you want to modify another user entry? [yes] n ♪
```

So, for Poulet, we changed his home directory and his initial program. Everything else remains the same.

Displaying User Information

This subsection discusses how to display user account entries.

Purpose	To list user account entries.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**lsuser**
References	**who**(1)

The **lsuser** command, option 5, prints information about all user accounts on the system. You can request information on all or specific users. The **lsuser** command will print the login names, user IDs, group names, login directories, and shells. Let's try both methods. First, we'll list all users, then we'll list information on one user, Dupree. Select the **lsuser** command and display information on all users:

```
User Name(s)? [all]  ↵

Login Name     UID    Group      Home Directory    Program
----------     ---    -----      --------------    -------
dupree         102    general   /mach_2/dupree      /bin/csh
kermit         103    general   /mach_2/kermit      /bin/sh
bond           104    largo     /mach_2/bond        /bin/csh


   Press the NEWLINE key to see the usermgmt menu [?, ^, q]:
```

Next, let's get information on just one user, Dupree:

```
User Name(s)? [all] dupree

Login name:           dupree
Real Name:            Morton El Dupree
User ID:              102
Group:                general
Login Directory:      /mach_2/dupree
Initial Program:      /bin/csh
Password:             set

Press the NEWLINE key to see the usermgmt menu [?, ^, q]:
```

Adding Groups

This subsection discusses how to add group names and members.

Purpose	To add a new group name. To add group members.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**addgroup**
References	**group**(4), **newgrp**(1)

The **addgroup** command, option 6, adds entries to the list of groups in **/etc/group**. A group entry consists of a group name, group number, and a list of group members. A group member is an established login name. In this procedure, let's create an example group named **writers**. Select **addgroup** from the **usermgmt** menu to enter into the following interaction:

```
Group name? writers ♪
```

A group name can be up to eight characters long. Use upper or lowercase letters, numbers, or a combination of letters and numbers. Always use a letter for the first character. If you type a name that is already in use, the system responds:

```
Group name writers already exists.  If you wish to modify
this group, quit now and use modgroup.  Otherwise, choose
another name.
```

Next, you are queried for the group ID number. The **sysadm** program will issue ID numbers by default. However, you may override **sysadm** and enter ID numbers. If you do, choose a number between 101 and 60,000 that is not being used by any other group. If you choose a number outside the specified range, or a number already in use, **sysadm** will ask you to change your entry. Below, we'll take the default ID number by pressing New Line:

```
Group ID? [106]  ♪
The member list for writers is now empty.
```

Since **writers** has just been created, it has no members. The next query is:

```
Group Member List Operation? [list] add ♪
```

Your choices for the "List Operation" query are:

add Enter member names.

delete Delete member names.

list Display list of member names.

end Write entries to **/etc/group**.

To use any of the above operations, the member must have already been added with the **adduser** command. Invalid names will be ignored. The default is **list**, but we choose to add members. After specifying the **add** operation, we specify the user name, then **end**:

```
Add User Name(s)? [none] poulet ♪
Group Member List Operation? [list] end ♪
```

Typing **end** installs the new group with its one member. Any member you specify must already exist. Next, the system displays:

```
Group writers has been added.
Do you want to create another group? [yes] n ♪

Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

So, we have created an example group **writers**.

Deleting Groups

This subsection discusses how to delete group entries from a group list.

Purpose	To delete a group.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**delgroup**
References	**group**(4), **newgrp**(1), **grpck**(1)

The **delgroup** command, option 7, deletes group entries from a group list. After **delgroup** prompts you, enter the name of the group you wish to remove:

```
Group name?  ranchers ♪

Do you really want to delete ranchers 103? [no]  y ♪
Group ranchers 103 has been deleted.

Do you want to delete another group? [yes] n ♪
Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

All files associated with group **ranchers** will now be listed by group ID number instead of by group name. If a user tries to change groups (with the **newgrp**(1) command), that user will receive an error message stating that the group, in this case **ranchers**, does not exist.

Modifying Groups

This subsection discusses how to modify entries in the group list.

Purpose	To modify entries in the group list.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**modgroup**
References	**group**(4), **newgrp**(1), **grpck**(1)

The **modgroup** command, option 8, modifies entries in the list of groups. You can change a group name, a group ID, or a group member. To modify a member list, you will use the add, delete, and list operations. After modifying one group, you have the option of modifying another or exiting **modgroup**. Below, let's modify the group **cowboys**. If you select the **modgroup** command, the queries follow:

```
Group Name?  cowboys ♪
New Group Name? [cowboys] cowdogs ♪
Group ID? [104] ♪

Group Member List Operation? [list] delete ♪
User Name(s) to be deleted? roy ♪

User Name(s) roy has been deleted.
```

After deleting user **roy**, let's check to see if the deletion has occurred.

```
Group Member List Operation? [list] list ♪

cisco    gene    paladin    poncho
Group Member List Operation?  end ♪

Group cowboys has been modified.
Do you want to modify another group? [yes] n ♪
Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

As the display shows, member **roy** has been deleted.

Listing Groups

This subsection discusses how to print group entries.

Purpose	To print group entries.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**lsgroup**
References	**group**(4), **grpck**(1)

The **lsgroup** command prints one or more group entries. A group entry consists of a group name, a group ID, and group members. Group members are listed by their login names. You can list all group entries by pressing New Line for the default value in the first query. If you select **lsgroup**, the system displays:

```
Group name(s)? [all]  ↵

Group Name     GID        Members
----------     ---   ---------------------
root             0   root
other            1
bin              2   root, bin, daemon
sys              3   root, bin, sys, adm
adm              4   root, adm, daemon
mail             5   mail, bin
lp               6   lp
uucp             8   uucp
daemon          12   root, daemon
operator        18   adm
nfs             38   nfs
ftp             39   ftp
general        101
farmers        102   grandpa, luke, pepino
cowdogs        104   paladin, cisco, poncho, gene
stooges        105   moe, larry, curly, shemp
writers        106   woody, bugs, daffy, elmer

Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

Adding Mail Aliases

This subsection discusses how to add an alias to the list of mail aliases.

Purpose	To add an alias to the list of mail aliases.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**addalias**
References	**mailx**(1), **sendmail**(1C), **newaliases**(1C)

Select option 10, **addalias**, to add a mail alias to the **aliases** file. A mail alias is a mailing list. An alias contains one or more login user names. If you address mail to an *alias*, you can send mail to all users listed *in* that alias. The **addalias** command adds names to alias entries. An alias entry usually consists of an alias name and a list of alias members. But, for now, let's make an empty mail alias called **everybody**; this will be the default alias that the **userdefaults** command will read when you add users to the system. When you select the **addalias** command, the interaction proceeds as follows:

```
Alias Name?  everybody ↵
```

An alias name can be from 1 to 32 characters long. Use upper or lowercase letters, numbers, or a combination of letters and numbers. Always use a letter for the first character.

The next query gives us the choice of using **add**, **delete**, **list**, or **end** as operations to perform on an alias. Below, let's just install the alias **everybody**. Later, if you choose, you can use the **add** operation to put members in this alias. The system displays the following queries for you to respond to.

```
Alias Member List Operation?  end ↵

Do you want to create another alias? [yes] n ↵
Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

By typing **end**, we added the empty alias **everybody** to the file **/etc/aliases**. Remember to type **?** if you need additional information on aliases.

Deleting Mail Aliases

This subsection discusses how to delete an alias from the list of mail aliases.

Purpose	To delete an alias from the mail alias list.
Starting Conditions	multiuser state
sysadm menu	usermgmt
Command	**delalias**
References	**mailx**(1), **sendmail**(1C), **newaliases**(1C)

The **delalias** command, option 11, deletes one or more alias entries from the mail alias list. When deleted, the alias is no longer available to the mail system. Let's say that earlier you used the **addalias** command to create alias **pawns**. Now you want to delete **pawns**. If you select **delalias**, the interaction proceeds as follows:

```
Alias Name? pawns ↵
Do you really want to delete pawns? [no] y ↵

Alias Name pawns has been deleted.

Do you want to delete another alias? [no] ↵

Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

Modifying Mail Aliases

This subsection discusses how to modify entries in the mail aliases list.

Purpose	To modify entries in the list of mail aliases.
Starting Conditions	multiuser state
sysadm menu	**usermgmt**
Command	**modalias**
References	**mailx**(1), **sendmail**(1C), **newaliases**(1C)

The **modalias** command, option 12, requests an existing alias name. If the name is valid, you can change the name and member list. You will use the add, delete, list, and end operations. Below, let's delete and add a member in alias **pawns**. If you select the **modalias** command, the interaction proceeds as follows:

```
Alias Name?  pals ♪
New Alias Name? [pals] ♪

Alias Member List Operation? [list] delete ♪
Delete Mail Address(es)?  kermit ♪

Member Name(s) kermit has been deleted.

Alias Member List Operation? add ♪
Add Mail Address(es)? lucy@sys5 ♪

Member Name(s) lucy@sys5 has been added.

Alias Member List Operation? end ♪

Do you want to modify another alias? [no]  ♪

Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

Above, we deleted **kermit** and added **lucy@sys5**. We typed **end** when we were ready to record these changes to the list. Finally, we answered **n** to the last query to exit from the **modalias** command.

Listing Mail Aliases

This subsection discusses how to print mail alias entries.

Purpose	To print the list of mail alias entries.
Starting Conditions	administrative or multiuser
sysadm menu	**usermgmt**
Command	**lsalias**
References	**mailx**(1), **sendmail**(1C), **newaliases**(1C)

The **lsalias** command, option 13, prints one or more alias entries. An alias entry consists of an alias name and alias members. The list of alias members includes mail login names for local aliases and addresses for remote aliases.

If you select the **lsalias** command, here is what happens:

```
Alias Name(s)? [all]   ↵

Alias Name              Members
----------        --------------------
everybody         luke, grandpa, pepino, dupree,
                  kermit, bond, paladin, cisco@sys3,
                  poncho, gene, moe, larry, curly,
                  shemp@sys5, lucy@sys5

pawns             kermit, pepino, curly

Press the NEWLINE key to see the usermgmt menu[?, ^, q]:
```

The User's Environment

The profile is the key element in establishing an environment in which users can successfully communicate with the computer.

Among the things that a profile can contain are:

- A PATH (searchlist) specifying directories to be searched for commands.
- Definitions of TERM (terminal) and TZ (time zone) variables.
- A command that prints the **/etc/motd** file upon login.
- Commands to print notification of new mail messages.

Types of Profile

Profiles are of two types: global and local.

The Global Profile

The global profile is an ASCII text file, **/etc/profile** for **sh** users or **/etc/login.csh** for **csh** users, that can contain commands, shell procedures, and environment variables. Whenever a user logs in, the **login** process executes the global profile. The global profile allows people to immediately begin using the system after an account is established. Later, users can modify their environments with the individual (local) profile. A sample global profile appears in Figure 14-2 below. The sample profile set the user's TERM and TZ variables as well as terminal characteristics. Then the profile displays the message of the day (**/etc/motd**) and prints a mail notification message if the user has mail.

```
#!/bin/sh
#
# Copyright (C) Data General Corporation, 1984 - 1988
# 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.
#
# Rcsid = "$What: <@(#) chap14,v          1.4> $"
#
# <PassStamp:_@(#)DG/UX_4.30.04.02>
#
# The profile that all logins get before using their own .profile.

trap "" 2 3

#       Set LOGNAME
export LOGNAME


#       Set TZ.
if [ -f /etc/TIMEZONE ]
then
elif [ -f /etc/TIMEZONE.proto ]
then
```

```
fi

#       Set TERM.
if      [ -z "$TERM" ]
then
        TERM=vt100      # for standard async terminal/console
        export TERM
fi

#       Login and -su shells get /etc/profile services.
#       -rsh is given its environment in its .profile.
case "$0" in
-su )
        export PATH
        ;;
-sh )
        export PATH

        # Allow the user to break the Message-Of-The-Day only.
        trap "trap '' 2"  2
        if [ -f /usr/bin/cat ] ; then
                cat -s /etc/motd
        fi
        trap "" 2

        if [ -f /usr/bin/mail ] ; then
                if mail -e ; then
                        echo "you have mail"
                fi
        fi
        ;;
esac

#       set the umask for more secure operation
umask 022
trap  2 3
```

Figure 14-2 *The Global Profile*

The Local Profile

The local or individual profile is **.profile** for **sh** users, and **.login** for **csh** users. These files reside in a user's home directory. These local profiles are copies of two prototype files: **/etc/stdprofile** and **/etc/stdlogin**. At the local profile level, users can add commands and variables to customize their environments. A local profile does not have to exist, but if it does exist, it is executed at login time, after the execution of the global profiles **/etc/profile** or **/etc/login.csh**.

Environment Variables

An array of strings called the environment is made available by **exec**(2) when a process begins. Since **login** is a process, the array of environment strings is made available to it. The local profiles in Figure 14-3 show how environment variables can be defined for users running the Bourne shell (**sh**(1)) or the C shell (**csh**(1)).

```
#LOCAL .PROFILE (sh)
#
#
LOGNAME=poulet
PWD=/usr/poulet
HOME=/usr/poulet
PATH=:/bin:/usr/bin
SHELL=/bin/sh
MAIL=/usr/mail/poulet
```

```
#LOCAL .LOGIN (csh)
#
#
set prompt = 'sys5>'
set noclobber
setenv PATH .:$HOME/bin:/usr/bin
```

Figure 14-3 *Local Profiles*

Other programs make use of the information in the environment array list. New strings can be defined at any time. By convention new strings are defined with the variable in uppercase letters, followed by an equal sign, followed by the value. The individual **.profile** and **.login** files can contain whatever the user wants.

Default Permissions Mode: umask

A system default controls the permissions mode of any files or directories created by a user. The DG/UX system gives default values of 666 for files and 777 for directories (see **chmod**(1M)). That means that for files everyone automatically gets read and write permission. For directories, everyone gets read, write, and execute permission. (Execute permission on a directory means the ability to **cd** to the directory and to copy files from it.)

Users frequently set up a user mask in their local profiles by means of the **umask**(1) command. The **umask** command alters the default permission levels (666) by a specified amount. You issue the **umask** command with a three-digit argument where the first digit tells how much to reduce the digit representing the owner's permissions; the second digit tells how much to reduce the digit representing group permissions; and the third digit tells how much to reduce the digit representing permissions for others. For example:

```
$ umask 027 ↵
```

leaves the permission level for owner unchanged, lowers the permission level for group by 2, and reduces the permissions for others by 7. Since system default is 666; this user mask changes it to 640, which translates into read and write permission for the owner, read permission for the group, and no permission for others. See the **chmod**(1) manual page for more information on permissions.

A **umask** command in a user's global profile does not change a user's ability to put one in the local profile. The local profile always overrides the global.

Default Shell and Restricted Shell

Generally, when a user logs in the default program that is started is **/bin/sh**. There may be cases, however, where a user needs to be given a restricted shell, **/bin/rsh**. A restricted shell is one where the user is not allowed to:

- Change directories
- Change the value of PATH
- Specify path names or command names containing a slash (**/**). That is, the user of a restricted shell may not access files or directories other than the present working directory or those included in PATH
- Redirect output

You can use a restricted shell strategy to limit certain users to the execution of a small number of commands or programs. By setting up a special directory for executables and controlling PATH so it only references that directory, you can restrict a user's activity in whatever way is appropriate for your system.

User Communication Services

Several ways of communicating with and among users are available. Some of the most frequently used are described in this section.

Message of the Day

You can put items of broad interest that you want to make available to all users in the **/etc/motd** file. The contents of **/etc/motd** are displayed on the user's terminal as part of the login process. The login process executes global files called **/etc/profile** for **sh** users and **/etc/login.csh** for **csh** users. In these files is commonly contained the command:

```
$ cat /etc/motd ↵
```

Any text contained in **/etc/motd** is displayed for each user each time the user logs in. For this information to have any impact on users, you must take pains to use it sparingly and to clean out outdated announcements. A typical use for the Message of the Day facility might be

```
5/30:  The system will be unavailable from 6-11pm Thursday, 5/30
- preventive maintenance.
```

Part of the preventive maintenance should be to remove the notice from **/etc/motd**.

News

Another electronic bulletin board facility is the **/usr/news** directory and the **news**(1) command. The directory stores announcements in text files, the names of which are usually used to provide an indication of the content of the news item. The **news** command displays the items on your terminal.

The **/etc/profile** file can also inform users about news items. A typical **/etc/profile** contains the line:

```
news -n
```

The **−n** argument causes the names of files in the **/usr/news** directory to be printed on a user's terminal as the user logs in. Item names are displayed only for current items, that is, items added to the **/usr/news** directory since the user last looked at the news. The idea of currency is implemented like this: when you read a news item an empty file named **.news_time** is written in your login directory. As with any other file, **.news_time** carries a time stamp indicating the date and time the file was created. When you log in, the system compares the time stamp of your **.news_time** file and time stamp of items in **/usr/news**.

Unlike the Message of the Day where users have no ability to turn the message off, with **news** users have a choice of several possible actions:

read everything — If the user enters the command, **news** with no arguments, all news items posted since the last time the user typed in the command are printed on the user's terminal.

select some items — If the **news** command is entered with the names of one or more items as arguments, only those items selected are printed.

read and delete — After the **news** command has been entered, the user can stop any item from printing by pressing the DELETE key. Pressing the DELETE key twice in a row stops the program.

ignore everything — If the user is too busy to read announcements at the moment, the messages can safely be ignored. Items remain in **/usr/news** until removed. The item names will continue to be displayed each time the user logs in.

flush all items — If the user simply wants to eliminate the display of item names without looking at the items, a couple of techniques will work:

$ touch .news_time ↵

updates the time-accessed and time-modified fields of the

$ news > /dev/null ↵

prints the news items on the null device.

Broadcast to All Users

With the **wall**(1M) command, you can send "broadcast" messages to the screens of all users currently logged on the system. While **wall** is a useful device for getting urgent information out quickly, users tend to find it annoying to have messages print out on their screens right in the middle of whatever else is going on. The effect is not destructive, but is somewhat irritating. It is best to reserve this for those times when you need to ask users to get off the system so that you may perform an administrative task.

DG/UX System Mail

The DG/UX system has two electronic mail utilities through which users can communicate among themselves. If your system is connected to others by networking facilities, **mail**(1) and **mailx**(1) can be used to communicate with persons on other systems.

The **mail** program is the basic utility for sending messages. The **mailx** program uses **mail** to send and receive messages, but adds some useful features for storing messages, adding headers, and many other functions. Because of these added features, **mailx** is often the preferred mail facility among users.

Users can, by default, send and receive mail when you add them to the system with the **adduser** command. A simple example of using **mailx** follows. In a hypothetical world, Poulet wants to send a mail message to Moe. He types:

```
$ mailx moe ↵

Subject: rubber chicken ↵

Please return my rubber chicken when the puppet show is over. ↵

^D

$
```

Poulet types in a `Subject:` line, presses New Line, and then types the message. When finished, he presses **<Ctrl-D>** on the next line after finishing the message, and the message is mailed.. The next time Moe presses the New Line key, or when he logs in, his screen will display:

```
You have new mail.
```

Then, Moe types the **mailx** command to see what mail has arrived. For detailed information on using mail facilities, see *Using the DG/UX™ System*.

A setup file called **.mailrc** contains the mail characteristics for each user. You can find a description of how to use a **.mailrc** set-up file in the **mailx**(1) pages of the *User's Reference for the DG/UX™ System*.

User Trouble Log

As the system administrator, you can expect users to look to you to help solve any number of problems. Others in your situation have found it helpful to keep a user trouble log. You will discover that user problems will fall into patterns. And if you keep a record of how these problems are resolved, you will not have to start from scratch every time a problem occurs. Figure 14-4 is an example trouble report that you can use to track and record system problems.

```
**************************************************************

                        TROUBLE REPORT

1. Machine _____________________________________________

2. DG/UX Revision ______________________________________

3. Program/command running _____________________________

4. Error Messages ______________________________________

   _____________________________________________________

   _____________________________________________________

5. Symptoms ____________________________________________

   _____________________________________________________


   Date __________

   Person Reporting _____________  Login _____________

   Location _____________________  Phone _____________

**************************************************************
```

Figure 14-4 *Sample Trouble Report*

Expert User Management Information

The remainder of this chapter provides additional information on user services. The **sysadm** program should take care of any questions you have while you are performing the major procedures for user services. Still, you should feel free to read and use this additional information to enhance your understanding and performance.

User Passwords

Before users are permitted to log in to your system, they must be listed in the **/etc/passwd** file. An entry in the **passwd** file consists of a single line with the following seven colon-separated fields:

login_name:*password*:*uid*:*gid*:*comment*:*home_directory*:*program*

For a user named L.Q. Poulet, the line might look like the following:

```
poulet:Rm27oQak1:103:104:L.Q. Poulet:/usr/poulet:/bin/csh
```

The fields are:

login name — `poulet`: A valid name for logging onto the system. A login name can be up to eight alphabetic or numeric characters; the first character must be alphabetic. It is usually chosen by the user.

password — `Rm27oQak1`: A user chooses a password and registers it with the system with the **passwd**(1) command. The encrypted form of the password appears in this field. No one but the user ever knows the real password. The actual password can be a maximum of eight characters. At least one character *must* be a numeric character or special character. This policy discourages users from choosing ordinary words as passwords. When you add a user to the file, you may use a default password, such as **passwd9**, and instruct the user to change it at the first login. Following the encrypted password, separated by a comma, there may be a field that controls password aging. For information on password aging see **passwd**(4) and **passwd**(1).

user ID — `103`: The user ID number (**uid**) is between 101 and 60,000. The number may not include any punctuation. Numbers 100 and below are reserved. User ID 0 is reserved for the superuser.

group ID — `104`: The same conditions apply to the group ID (**gid**) number as to the **uid**.

comment — `L.Q.Poulet`: Optional. May contain user's name, office, office phone number, home phone number, and so on. There is no required format for this field.

home directory — `/usr/poulet`: The directory where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory such as **/usr**. The home directory is the origination point of the user's directory tree.

program — `/bin/csh`: The name of a program invoked at the time the user logs in. If the field is empty, the default program is **/bin/sh**. This field is most commonly used to invoke a special shell, such as **/bin/rsh** (restricted shell).

As noted above, the password field may contain a subfield that controls the aging of passwords. A description of how the process works can be found in **passwd**(4). The effect is to force users periodically to select a new password. If password aging is not used, a person can keep the same password indefinitely.

Password Aging

The password aging mechanism forces users to change their passwords periodically. It also prevents them from changing a password before a specified time interval. Password aging is selectively applied to logins by editing the **/etc/passwd** file.

The password aging information is appended to the encrypted password field in the **/etc/passwd** file. The password aging information consists of a comma and up to four characters in the format:

,Mmww

The meaning of these fields is as follows:

, — The delimiter between the password itself and the aging information.

M — A single character from the 64-character alphabet (described below) representing the maximum duration of the password in weeks.

m — A single character from the 64-character alphabet (described below) representing the minimum time interval before the existing password can be changed by the user, in weeks.

ww — Two characters from the 64-character alphabet (described below) representing the week (counted from the beginning of 1970) when the password was last changed. You add this information through the codes that you edit into the **/etc/passwd** file. Then, the system automatically adds these characters to the password aging field. All times are specified in weeks (0 through 63) by a 64-character alphabet (. (dot), / (slash), 0-9, A-Z, a-z) where **.** is zero, **/** is one, **0** is 2, and so on. The expression **Ea**, for example, represents the base-10 value 1061.

Table 14-1 shows the relationship between the numerical values and character codes. Any of the character codes may be used in the four fields of the password aging information.

Table 14-1 Password Aging Character Codes

Character	Number of Weeks
. (period)	0 (zero)
/ (slash)	1
0 through 9	2 through 11
A through Z	12 through 37
a through z	38 through 63

Two special cases apply for the character codes:

- If *M* and *m* are equal to zero (the code, ..), the user is forced to change the password at the next login. No further password aging is then applied to that login.
- If *m* is greater than *M* (for example, the code, **./**), only **root** is able to change the password for that login.

Group IDs

Group IDs are a means of establishing another level of ownership of and access to files and directories. Users with some community of interest can be identified as members of the same group. Any file created by a member of the group carries the group ID as a secondary identification. By manipulating the permissions field of the file, the owner (or someone with the effective user ID of the owner) can grant read, write, or execute privileges to other group members.

Information about groups is kept in the **/etc/group** file. Entries consist of the following colon-separated fields:

group_name:*password*:*gid*:*login_names*

A sample entry from this file is shown and explained below:

```
prog::104:reynard, poulet
```

Each entry is one line; each line has the following fields:

group_name `prog`: The group name can be up to eight characters, the first of which must be alphabetic.

password The password field should not be used. Leave this field blank.

group_ID `104`: The group ID is a number from 101 to 60,000. The number may not include any punctuation. Numbers below 100 are reserved.

login_names `reynard, poulet`: The login names of group members are in a comma-separated list. A user may be a member of more than one group. Nothing prevents a user from having more than one login name, however, as long as each name is unique within the system.

Sample /etc/passwd Entries

Password administration can be set up in a variety of ways to meet the needs of different organizations. Some examples are discussed in the following sections. The following shows the password aging information required to establish a new password every 2 weeks (0) and to deny changing the new password for 1 week (/).

Here is a typical login/password entry in the **/etc/passwd** file for the typical user **jqu**:

```
jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:
```

To cause **jqu** to change the password at least every 2 weeks, but keep it at least for 1 week, you should use the code **0/**. After you edit the **/etc/passwd** file, adding **,0/** to the password field, the entry looks like this:

```
jqu:RTKESmMOE2m.E,0/:100:1:J. Q. Username:/usr/jqu:
```

After the password entry is changed, **jqu** will have to change the password at the next login and every 2 weeks thereafter.

After **jqu**'s first login following the change, the system automatically adds the two-character, "last-time-changed" information to the password field.

```
jqu:RTKESmMOE2m.E,0/W9:100:1:J. Q. Username:/usr/jqu:
```

In this example, **jqu** changed the password in week W9.

Changing or Deleting Aliases

As with changes to the **/etc/passwd** file, all changes that you make when adding or deleting an alias by hand in the **/etc/aliases** file should adhere to a format that the **sysadm** program can use. This format is:

```
alias_name:          name1, name2, name3, name4, name5, name6,
                     name7, name8, name9, name10

alias_name2:         nameA, nameB, nameC, nameD
```

There must be a colon after each `alias_name`. All alias member names, except the last one, must be separated by commas; spaces are ignored. The last entry in the member list should not be followed by a comma. If `name10` had a comma after it, the system would search for another member name and would erroneously read `alias_name2` as a member name.

After you edit **/etc/aliases**, run the following command:

```
# /usr/bin/newaliases -bi ↵
```

This command initializes the alias database, displaying the total number of aliases, the length in bytes of the longest alias, and the length in bytes of the entire aliases file. When the command has finished, your changes are available to the system.

End of Chapter

Chapter 15
Accounting Management

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. System-use data is logged and then organized and directed into summary files and reports which you can print or display on your terminal. Among other things, these reports are useful for helping to maintain security because you can trace who logged on when and what commands were run.

The accounting functions are not handled by the **sysadm** program.

The major sections of this chapter are:

- What the Accounting System Does
- The Default Accounting System
- How the Accounting System Works
- Daily Accounting with Runacct
- Runacct Accounting Reports
- Recovering From Failure
- Updating Holidays
- Accounting Directories and Files

What the Accounting System Does

The accounting system does the following:

- Records connect sessions, logins, date changes, reboots, and shutdowns.
- Records disk use and system use for each login.
- Formats accounting data into summary files and reports (daily).
- Saves summary files, generates a report, and cleans up files (monthly).

The Default Accounting System

To make accounting easier to use, the DG/UX system comes with a ready-to-go method of doing your accounting. This section describes this accounting system. The remainder of this chapter shows you in detail how the components of the accounting system function.

When you have set up the system and brought it up to run level 3, the accounting system does the following:

- Executes **runacct** daily at 4 a.m. to collate statistics on connects, user activity, CPU usage, fees, disk usage, and so on.
- Direct error messages to **fd2log**.
- Execute **dodisk** daily at 2 a.m. to perform disk accounting.
- Execute **ckpacct** at 5 minutes after every hour to monitor the size of the **pacct** data repository file, renaming it when it reaches its size limit.
- Execute **monacct** at 5:15 a.m. on the first day of every month to collate and summarize monthly statistics.
- Clean out files at 3 a.m. in **/tmp** that are older than three days.

The above actions are executed by the following lines in **/var/spool/cron/crontabs/root.proto**:

```
0 4 * * * /bin/su - adm -c "/usr/lib/acct/runacct \
        2> /usr/adm/acct/nite/fd2log"
5 * * * * /bin/su - adm -c "/usr/lib/acct/ckpacct"
15 5 1 * * /bin/su - adm -c /usr/lib/acct/monacct
0 2 * * * /usr/lib/acct/dodisk
0 3 * * 2-5 /bin/find /tmp /var/tmp /usr/tmp -mount \
        -atime +3 ! -name 'X[0-9]*' -exec rm {} ;
5 4 * * 6 /usr/lib/newsyslog >/dev/null 2>&1
```

The **monacct** procedure cleans up all daily reports and daily total accounting files and deposits one monthly total report and one monthly total accounting file in the **fiscal** directory. The default action of **monacct** adds the current month's date to the file names.

To make these lines available to **cron**, type:

```
# cd /var/spool/cron/crontabs ♪
# cat root.proto >> root ♪
# crontab root ♪
```

Changing the Default Accounting System

If you wish to change the default accounting system, edit **/var/spool/cron/crontabs/root**. Any time you make changes to this file, you must run the **crontab**(1) command on it.

Turning on Accounting

Before your accounting system will work, you must do two things:

1. Edit **/etc/dgux.params**. Change **account_START="false"** to **account_START="true"**. With these parameters set, accounting will be in effect the next time to bring your system up to run level 2.

2. If you are already at run level 2 or higher and want to bring accounting up immediately, type this command:

   ```
   # /usr/lib/acct/turnacct on ♪
   ```

   To turn accounting off, specify **off** as the argument to **turnacct**.

Printing Default System Daily Reports

The default accounting system produces daily reports. If you type the following command line, you will display the reports assembled for April 11. Without a date designation, **prdaily** prints the previous day's reports. You can also direct the output to a file using a command line like this:

```
# prdaily > yesterday_acct ♪
```

To direct the output to a lineprinter, use a command line like this:

```
# prdaily | lp ♪
```

```
# /usr/lib/acct/prdaily 0411 ♪
```

Printing Default System Monthly Reports

The default accounting system produces monthly reports. Monthly reports are in **/usr/adm/fiscal/fiscrpt***xx*, where *xx* is the number of last month, such as 01 for January. You can direct the contents of monthly report to your screen with a command line like this:

```
# more /usr/adm/fiscal/fiscrpt02 ↵
```

To print this file, use a command line like this:

```
# lp /usr/adm/fiscal/fiscrpt02 ↵
```

How the Accounting System Works

The accounting system may be thought of as having three major parts: a logging mechanism, a scheduling mechanism, and a processing mechanism for the data that is logged.

- The general *logging mechanism* is activated by a script called **/usr/lib/acct/startup**; it does the major logging work for all user processes running. The **startup** script is executed according to a setting in **/etc/inittab**. Normally, the **startup** script is set to begin running in multi-user run levels 2 or 3. In this case, whenever your system enters run levels 2 or 3, the **startup** script is executed, thus the logging mechanism for the accounting system automatically begins working in run levels 2 or 3 (or whatever you set). Chapter 4 explains how the **init**(4) program reads entries from **/etc/inittab** to establish run levels.

- The *scheduling mechanism* is the **cron**(1M) daemon. You use **cron** as a timer to start the accounting programs at regular intervals. We demonstrated how this scheduling mechanism works earlier in "The Default Accounting System."

- The *processing mechanism* is a set of accounting programs in **/usr/lib/acct** that you specify for **cron** to execute.

The following programs in **/usr/lib/acct** make up the *processing mechanism* we just described. These programs are divided into two categories: user level and internal.

User Level Accounting Programs

You can invoke the following programs from the shell.

turnacct — An interface to the **accton** program that turns process accounting on or off. When switched "on", **turnacct** starts the **accton** program; "off" stops the **accton** program. When "off", the accounting system is disabled.

acctcom — Searches and prints process accounting files. Note that this command is in **/usr/bin**.

chargefee Records billing information for file restores and other services and writes the data to **/usr/adm/fee**. This data is picked up by the **runacct** program and merged into the total accounting records.

prdaily Displays the previous day's accounting report.

wtmpfix Checks **/var/adm/acct/nite/wtmp** as step 2 in the execution of the **runacct** program. See "Daily Accounting with Runacct" later in this chapter for details on **runacct**.

fwtmp Fixes corrupted accounting files by manipulating content from binary to ASCII and back. See "Fixing Corrupted Files" later in this chapter.

acctcms Produces a summary of all processes that execute commands. See **acctcms**(1M).

acctprc1 Reads input in the form described by **acct**(4) and adds login names corresponding to user IDs. Then, it writes an ASCII line for each process giving user ID, login name, prime and non-prime CPU time (measured in *ticks*), and mean memory size in memory segment units. **Acctprc1** gets login names from **/etc/passwd**. It also reads **/var/adm/acct/nite/ctmp** to distinguish among different login names that share the same user ID.

acctprc2 Reads records in the form written by **acctprc1**, summarizes them by user ID and name, then writes the sorted summaries to the standard output as total accounting records.

prctmp Prints the login session record file created by **acctcon1** (normally **/var/adm/acct/nite/ctmp**).

prtacct Formats and prints total accounting files (**tacct**).

Internal Accounting Programs

The following programs are invoked through and interact with other accounting programs. They are internal to general accounting functions and therefore you should not invoke them as regular commands.

acctwtmp Records boot times in **/etc/wtmp**.

accton A kernel program that monitors processes, collects data, and records that data in **/var/adm/pacct**. The programs described in **acctprc**(1M) summarize this data. Use **acctcom**(1) to examine current process data.

remove A shell procedure that cleans up the saved **pacct** and **wtmp** files left in **/var/adm/acct/sum**.

ckpacct	Checks and controls the size of **/var/adm/pacct**. When **pacct** grows larger than 1100 blocks, **turnacct** "off" turns **accton** "off", and renames the current **pacct** file to **pacct1** and creates a new **pacct**. Finally, **ckpacct** turns **accton** back on.
runacct	The main shell program for daily accounting. See "Daily Accounting with Runacct" later in this chapter for details on **runacct**.
monacct	Uses the daily data organized by **runacct** and writes it into a monthly summary. Invoke **monacct** via **cron** once each month or each accounting period. **Monacct** creates summary files in **/var/adm/acct/fiscal**. Table 5-4 shows how **monacct** organizes this data. See **acctsh**(1M).
dodisk	Does disk accounting on the special files in **/etc/fstab**. Creates **/var/adm/dtmp**.
diskusg	Generates disk accounting information. See **diskusg**(1M).
acctcon1	Reads **/etc/wtmp** and converts login/logoff data to a sequence of records, one per login session. The output is ASCII, giving device, user ID, login name, prime connect time (seconds), non-prime connect time (seconds), session starting time, and starting date and time. See **acctcon**(1M) for complete details.
acctcon2	This program expects a sequence of login session records as input and converts them into total accounting records (**tacct**). See **acct**(4) for the format of **tacct**.
acctdisk	Reads lines from **/var/adm/dtmp** (created by **dodisk**) containing user ID, login name, and number of disk blocks, and converts them to total accounting records that are merged with other accounting records. See **acct**(1M).
acctdusg	Computes disk resource consumption (including indirect blocks) for each login. See **acct**(1M).
acctwtmp	Called by the **shutdown**(1M) command. Writes a **utmp**(4) record containing the current time and a string of characters describing "reason" which is either **accton** or **acctoff**.
acctmerg	Merges total accounting files (**tacct**). See **acctmerg**(1M).
lastlogin	Invoked by **runacct** to update **/var/adm/acct/sum/loginlog**, which shows the last date each person logged in.
nulladm	Called by most accounting scripts. Creates files with mode 664 and ensures that owner and group are **adm**.

shutacct Invoked automatically during system shut down (via **/etc/shutdown**) to turn process accounting off and append a "reason" record to **/etc/wtmp**.

The data organized by the accounting programs is manipulated, stored, and cleaned up through interactions with files in the following directories:

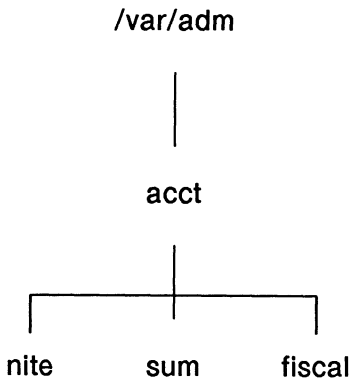


Figure 15-1 *The Accounting Directories*

The **/var/adm** directory contains the data collection files. The **/var/adm/acct/nite** directory contains files that are reused daily by the **runacct** procedure. The **/var/adm/acct/sum** directory contains the cumulative summary files updated by **runacct**. The **/var/adm/acct/fiscal** directory contains periodic summary files created by **monacct**.

Daily Accounting with Runacct

Runacct(1M) is the main daily accounting shell procedure. It is normally started by the **cron**(1M) program. **Runacct** works on files that contain data on connects, fees, disk usage, and accounting processes. It also prepares daily and cumulative summary files for use by the **prdaily** program or for billing purposes. **Runacct** programs produce the following files (the **nite** and **sum** directories reside in **/usr/adm/acct**):

nite/lineuse Produced by **acctcon**(1M), which reads the **wtmp** file, and produces usage statistics for each terminal line on the system. This report is useful for detecting bad lines. If the ratio between the number of logoffs to logins exceeds about 3:1, the line is probably failing.

nite/daytacct Yesterday's total accounting file in binary format.

sum/tacct The accumulation of each day's **nite/daytacct** procedure; it can be used for billing. The **monacct** shell procedure restarts the file each month or fiscal period.

sum/daycms Produced by the **acctcms** program; contains the daily command summary. The ASCII version of this file is **nite/daycms**.

sum/cms The accumulation of each day's command summaries. Restarted by the execution of **monacct**. The ASCII version is **nite/cms**.

sum/loginlog Produced by the **lastlogin** shell procedure. Maintains a record of the last time each login was used.

sum/rprt.*MMDD*
Each execution of **runacct**(1M) saves a copy of the output of the **prdaily** shell procedure in this file.

Runacct(1M) does not damage files if errors occur. It tries to recognize errors, provide diagnostics, and terminate processing so that it can be restarted easily. **Runacct** writes records of its progress into a file named **active** and uses files in the **/var/adm/acct/nite** directory unless otherwise noted. By default, all diagnostics from a **runacct** are directed into the **/var/adm/acct/nite/fd2log** file.

When **runacct** starts up, its run is protected by lock files which cause any multiple invocations of the program to terminate. These lock files in the **nite** directory are named **lock** and **lock1**. The **lastdate** file contains the month and day **runacct** was last invoked and prevents more than one execution per day. If **runacct** detects an error, it writes a message to **/dev/console**, sends mail to **root**, removes the locks, saves the diagnostic files, and terminates execution.

So that **runacct** can be restartable, processing is broken down into separate reentrant states, using a case statement inside a *while* loop. Each state is one instance of the **case** statement.

When each state completes, **nite/statefile** is updated to reflect the next state. In the next loop through the **while**, **statefile** is read and the case falls through to the next state. At the CLEANUP state, **runacct** removes the locks and terminates. States are executed in the following order:

SETUP Move active accounting files into working files. The command **turnacct switch** is executed. The process accounting files, **/var/adm/pacct?**, are moved to **/var/adm/Spacct?.***MMDD*. The **/etc/wtmp** file is moved to **/var/adm/acct/nite/owtmp**.

wtmpfix Verify the integrity of **/etc/wtmp**, correcting date changes if necessary. The **wtmp** file in the **nite** directory is checked by the **wtmpfix** program. Some date changes cause the **acctcon1** shell procedure to fail, so **wtmpfix** tries to adjust the time stamps in the **wtmp** file if a date change record appears.

CONNECT1 Produce connect session records. Connect session records are written to **ctmp** in binary form. The **lineuse** file is created, and the **reboots** file is created showing all of the boot records the **wtmp** file.

CONNECT2 Convert session records into total accounting records. **Ctmp** is converted to **ctacct.***MMDD* which contains accounting records.

PROCESS The **acctprc1** and **acctprc2** programs convert the process accounting files, **/var/adm/Spacct?.***MMDD*, into total accounting records in **ptacct?.***MMDD*. The **Spacct** and **ptacct** files are correlated by number so that if **runacct**(1M) fails, **Spacct** files are not reprocessed.

NOTE: When restarting **runacct** in this state, remove the last **ptacct** file; the **ptacct** file will not be complete.

MERGE Merge the process accounting records with the connect accounting records to form the **daytacct** file.

FEES Merge any ASCII records from the file **fee** with **daytacct**.

DISK On the day after the **dodisk** procedure runs, merge **disktacct** with **daytacct**. This merge forms the daily total accounting records.

MERGETACCT Merge **daytacct** with **sum/tacct**, the cumulative total accounting file. Each day, **daytacct** is saved in **sum/tacct***MMDD*, so that **sum/tacct** can be recreated if it is corrupted or lost.

CMS Merge today's command summary with the cumulative command summary file **sum/cms**. Produce ASCII and internal format command summary files.

USEREXIT Include installation dependent (local) accounting programs here.

CLEANUP Clean up temporary files, run **prdaily** and save its output in **sum/rprt***MMDD*, remove the locks, then exit.

At the completion of the last state, an ASCII file exists in directory **sum**; it has a four-digit suffix on it where the first two digits represent the month, and the last two represent the day; for example, **rprt1120** is the report for November 20. To display the previous day's report, execute **/usr/lib/acct/prdaily**. For a monthly summary, you can print the reports assembled by **monacct**. These reports, which have a two-digit suffix representing the month, are in directory **fiscal**. The report for November would be **fiscrpt11**.

Runacct Accounting Reports

Runacct generates the following accounting reports:

- Daily line usage
- Daily usage by login name
- Daily and monthly total command summaries
- Last login

You cannot generate an individual accounting report, say one just for today's line usage. All five reports are generated at once. Sample reports and meanings of their data follow.

Daily Line Usage

The first part of the daily line usage report is the **from/to** banner. The banner displays the time the last accounting report was generated and the time the current accounting report is generated. It is followed by a log of system reboots, shutdowns, recoveries, and any other record dumped into **/etc/wtmp** by the **acctwtmp** program.

The second part of the report is a breakdown of line usage. `Total Duration` tells how long the system was accessible through the terminal lines.

Figure 15-2 is an example daily line usage report:

```
Nov 11 16:00 1989  DAILY REPORT For DG/UX page 1

from Tue Nov 10 08:27:00 1989
to   Wed Nov 11 04:00:42 1989

2    shutdown

1    runacct

Total Duration is 1174 Minutes

Line     Minutes    Percent    # Sess    # On    # Off
--------------------------------------------------------
tty01          0          0         1       1        0

tty02        506         43         1       1        1

tty03         48          4         5       5        5

console       41          3         1       1        1

TOTALS       693         --         9       9        8
--------------------------------------------------------
```

Figure 15-2 *Line Usage Report*

The columns in the report above are defined as follows:

`Line` Terminal line or access port

`Minutes` Total minutes of line use during the accounting period

`Percent` Total minutes of line use divided into the total duration

`# Sess` Number of times this port was accessed for a **login**(1) session

`# On` Number of times the port was used to log a user in to the system.

`# Off` Number of times a user logged off, and any interrupts on that line. Generally, interrupts occur on a port when **getty**(2) is first invoked when the system goes to multi-user mode. Interrupts occur at a rate of about two per event. Therefore, you often see more than twice the number of **Off** than **On** or **Sess**. If the number of of **Off** exceeds the number of **On** by a large factor, it usually indicates a faulty or failing multiplexer, modem, or cable connection. An unconnected cable dangling from the multiplexer can cause this.

During normal operation, you should monitor the size of **/etc/wtmp** because this is the file from which the connect accounting is geared. If the file grows rapidly, execute **acctcon1** to see which tty line is the noisiest. If interruption is occurring at a rapid rate, it can affect general system performance.

Daily Usage by Login Name

The daily usage by login name report gives a breakdown of system resource use for each user. Its data consists of:

`UID` User ID.

`LOGIN NAME` Login name of the user. (There can be more than one login name for a single user ID.)

`CPU (MINS)` CPU time used, divided into PRIME and NPRIME (nonprime). See "Updating Holidays," later in this chapter.

`KCORE-MINS` Total memory a process used, in kilobytes per minute. Divided into PRIME and NPRIME.

`CONNECT (MINS)`
How long a user was logged into the system, by PRIME and NPRIME time. If this time is long and the column #OF PROCS is low, this user rarely uses the terminal.

`DISK BLOCKS` When the disk accounting programs have been run, their output is merged into the total accounting record (**tacct.h**) and shows up in this column. **Acctdusg** does disk accounting.

`# OF PROCS` Number of processes invoked by the user. Large numbers may indicate that a shell procedure looped.

`# OF SESS` Number of times the user logged in to the system.

`# DISK SAMPLES`
Number of times the disk accounting was run to obtain the average number of disk blocks.

`FEE` Total accumulation of total charges against the user. The **chargefee** procedure charges for special services, such as file restores and tape manipulation by operators. This process is low-level and will track whatever units you define.

Figure 15-3 is a sample report for each user.

```
Nov 11 04:42 1989 Daily Usage Report

Uid Login    CPU (Min)   Kcore-Mins  Connect(Min)  Disk   # Of  # Of # Disk
    Name   Prime NPrime Prime NPrime Prime NPrime  Blocks Procs Sess Samples Fee
-------------------------------------------------------------------------------

0   TOTAL  37    235    1198  467190 56    130     0      6142  11   0       0

0   root   5     1      472   84     41    0       0      712   1    1       0

2   bin    0     0      0     0      0     0       397415 0     0    1       0

3   sys    0     0      0     0      0     0       0      0     1    0       0

4   adm    0     1      0     78     0     0       17630  291   0    1       0

5   uucp   2     2      182   249    0     0       0      740   2    1       0

8   mail   0     0      13    3      0     0       0      13    4    1       0

201 moe    5     0      718   0      0     0       0      151   1    1       0

202 larry  0     1      15    5      0     99      0      13    1    1       0

203 curly  25    5      3069  371    15    31      3345   404   2    1       0

204 poulet 0     0      0     0      0     0       32     0     0    1       0
```

Figure 15-3 *Daily Usage by Login Name*

Daily and Monthly Total Command Summaries

The daily and monthly total command summary reports are similar, but the Daily Command Summary reports only the current accounting period; the Monthly Total Command Summary reports from the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of **monacct**.

These reports tell which commands are used most. Since you know which system resources the commands use, you can tune the system accordingly. These reports are sorted by `TOTAL KCOREMIN` (see below), which is a good way to calculate drain on a system.

`COMMAND NAME` The name of the command. All shell procedures are reported under the name **sh**. Monitor the frequency of programs called **a.out**, or **core**, or any other program that does not conform to the DG/UX command structure. **Acctcom** is also a good way to determine who executed an incorrect command and if superuser privileges were used.

NUMBER CMDS	Number of times a command was invoked.
TOTAL KCOREMIN	Total Kbyte segments of memory used by a process, per minute of run time.
TOTAL CPU-MIN	A program's total processing time.
TOTAL REAL-MIN	Total real-time minutes this program has run.
MEAN SIZE-K	Mean of the TOTAL KCOREMIN divided by TOTAL CPU-MIN.
MEAN CPU-MIN	Mean of the NUMBER CMDS and TOTAL CPU-MIN.
HOG FACTOR	Ratio of system availability to system usage: hog factor = (total CPU time) / (elapsed time) This measures the total available CPU time the process used.
CHARS TRNSFD	Number of characters manipulated by the **read**(2) and **write**(2) system calls. The value in this column may be negative.
BLOCKS READ	Total physical block reads and writes that a process performed.

Figures 15-4 and 15-5 are examples of daily and monthly command summaries.

Nov 11 04:42 1989 Daily Command Summary

Command Name	Numb Cmds	Total Koremin	Total CPU-Min	Total Real-Min	Mean Size-K	Mean CPU-Min	Hog Factor	Chars Trnsfd	Blocks Read
TOTALS	2332	1624.74	16.05	15210.91	5.67	003	0.01	0	0
nroff	1	249.86	2.03	2.91	123.19	2.03	0.70	0	0
troff	3	305.12	2.60	3.10	411.11	2.89	0.99	0	0
sh	1028	434.53	7.24	7632.44	59.99	0.01	0.01	0	0
csh	1115	474.13	3.96	6534.53	41.111	0.01	0.40	0	0
sendmail	18	55.79	0.93	2.10	155.96	0.04	0.26	0	0
ls	111	62.69	0.57	4.35	98.99	0.01	0.21	0	0
more	35	25.43	0.19	207.16	84.93	0.06	0.00	0	0
ps	1	20.74	0.63	0.35	173.62	0.14	0.33	0	0
cp	20	317.311	2.94	3.41	339.97	0.09	0.21	0	0

Figure 15-4 *Daily Command Summary*

```
Nov 11 04:42 1989 Monthly Total Summary

Command  Numb   Total     Total   Total     Mean    Mean    Hog    Chars  Blocks
Name     Cmds   Koremin   CPU-Min Real-Min  Size-K  CPU-Min Factor Trnsfd Read
-------------------------------------------------------------------------------

TOTALS   27792  17118.47  227.94  77021.94  1122.74 3.71    0      0      0

nroff    30     4216.86   34.03   58.99     123.19  1.03    0.70   0      0

troff    23     3105.12   25.60   43.10     411.11  1.89    0.99   0      0

sh       13118  5551.53   93.16   50386.06  59.59   0.01    0.01   0      0

csh      10115  3474.13   33.96   26534.53  41.11   0.01    0.40   0      0

sendmail 238    1455.79   8.93    52.10     165.97  0.03    0.16   0      0

ls       1075   1062.21   9.57    54.85     113.99  0.01    0.17   0      0

more     185    825.43    6.19    107.16    124.93  0.06    0.00   0      0

ps       38     520.74    3.63    11.35     181.72  0.08    0.50   0      0

cp       2970   384.31    8.94    52.85     43.93   0.00    0.17   0      0

-------------------------------------------------------------------------------
```

Figure 15-5 *Monthly Command Summary*

Last Login

This report gives the date when a particular login was last used. The report can help you find likely candidates for the tape archives, such as **/usr** directories associated with unused login names.

Table 15-1 Last Login Report

```
Nov 11 04:42 1989 LAST LOGIN

00-00-00        bin     89-11-02        carson

00-00-00        croot2  89-10-09        moe

00-00-00        daemon  89-11-11        larry

00-00-00        svvs    89-11-10        curly

00-00-00        archive 89-11-01        tlp

87-11-12        poulet  89-11-11        uucp
```

A field of all zeros means that login has not been used since the last invocation of the **lastlogin** program.

Recovering from Failure

If the system crashes, **/usr** runs out of space, or a **wtmp** file is corrupted, **runacct** fails. If the **active***MMDD* file exists, check it first for error messages. If the **active** file and **lock** file exist, check **fd2log** for error messages.

Runacct produces the following error messages. We suggest ways to recover from them.

```
acctg already run for date: check /var/adm/acct/nite/lastdate
```

The date in **lastdate** and today's date are the same. Remove **lastdate**

```
connect acctg failed: check /var/adm/acct/nite/log
```

The **acctcon1** program encountered a bad wtmp file. Use **fwtmp** to fix the bad file. See "Fixing Corrupted Files" later in this chapter.

```
locks found, run aborted
```

The files **lock** and **lock1** were found in **/var/adm/acct/nite**. Remove these files before restarting **runacct**.

```
Spacct?.MMDD already exists
```

File setups have already run. Check status of files, then run setups manually. See the following section, "Restarting Runacct."

```
turnacct switch returned rc=?
```

Check the integrity of **turnacct** and **accton** by ensuring that the **accton** program is owned by **root** and has the setuid bit set. See **setuid**(2).

```
/var/adm/acct/nite/wtmp.MMDD already exists, run setup manually.
```

See "Fixing Corrupted Files" later in this chapter.

```
wtmpfix errors see /var/adm/acct/nite/wtmperror
```

Wtmpfix detected a corrupted **wtmp** file. Use **fwtmp** to fix the file.

Restarting Runacct

Runacct called without arguments assumes this is the first invocation of the day. You must use the argument *MMDD* if **runacct** is being restarted; *MMDD* specifies the month and day for which **runacct** will rerun the accounting. The entry point for processing is based on the contents of **statefile**. To override **statefile**, include the desired state on the command line. As we said earlier, **runacct** is normally started by **cron**. But should you need to start **runacct** from the command line, here are three ways you might do it:

To start **runacct**, type:

```
# nohup runacct 2> /var/adm/acct/nite/fd2log & ↵
```

To restart **runacct** specifying *MMDD* (0601), type:

```
# nohup runacct 0601 2> /var/adm/acct/nite/fd2log & ↵
```

To restart **runacct** at a specific state, such as wtmpfix, type:

```
# nohup runacct 0601 wtmpfix 2> /var/adm/acct/nite/fd2log & ↵
```

In the above examples, the **2** sends the standard error output to the file named **f2dlog**; check this file periodically for error messages. Specifying *MMDD* creates a new **active0601** file, dated June 1.

Fixing Corrupted Files

Sometimes, a file is corrupted or lost. Although you may restore some files from backup tapes, you must fix the **wtmp** and **tacct** files yourself.

Fixing wtmp Errors

Wtmp files record who logged in and when. When the date is changed and the DG/UX system is in multi-user mode, a set of date change records is written into **/etc/wtmp**. The **wtmpfix** program adjusts the time stamps in the **wtmp** records when a date change is encountered.

Some combinations of date changes and reboots, however, result in nonsense lines being added to **/etc/wtmp**; these lines cause **acctcon1** to fail. When this happens, **wtmp***MMDD* is created. The following procedure shows you how to fix this file. At the editing step below, you'll see binary lines mixed in with ASCII lines. Fixing this file consists of deleting the binary lines. Here's the procedure:

1. Go to directory **/var/adm/acct/nite**.

2. Enter the following at the prompt:

   # **fwtmp < wtmp***MMDD* **> xwtmp** ♪

   This command line executes the **fwtmp**(1M) program on the contents of **wtmp***MMDD*, and stores the output in file **xwtmp**. So what you are doing here is converting the binary contents of **wtmp***MMDD* into ASCII format.

3. Now, edit **xwtmp** and delete all binary lines.

4. When you're finished editing, convert from ASCII back to binary.

   # **fwtmp −ic < xwtmp > wtmp.***MMDD* ♪

If the **wtmp** file is beyond repair, create a null **wtmp** file. This will prevent any charging of connect time.

Fixing tacct Errors

If you are using the accounting system to charge users for system resources, the integrity of **/usr/adm/acct/sum/tacct** is quite important. Occasionally, corrupt **tacct** records appear with negative numbers, duplicate user IDs, or a user ID of 60,000.

First, check **/usr/adm/acct/sum/tacctprev** with **prtacct**. If **prtacct** does not report any errors, patch up **/usr/adm/acct/sum/tacct.***MMDD* and recreate **sum/tacct**. A simple patch-up procedure is:

1. Go to directory **/var/adm/acct/sum**.

2. Enter the following at the prompt:

   # **acctmerg −v < tacct.***MMDD* **> xtacct** ♪

   This command line executes the **acctmerg**(1M) program on the contents of **tacct.***MMDD*, and stores the output in file **xtacct**. So what you are doing is converting the binary contents of **tacct.***MMDD* into ASCII format.

3. Now, edit **xtacct** and delete all corrupted records.

4. When you've finished editing, convert from ASCII back to binary. Type the following at the prompt.

   # **acctmerg −i < xtacct > tacct.***MMDD* ♪

Remember that **monacct** removes all the **tacct.***MMDD* files; therefore, when you merge these files together you recreate **sum/tacct**.

Updating Holidays

The file **/usr/lib/acct/holidays** contains the prime/non-prime table for the accounting system. Edit the table to reflect your holiday schedule for the year. The table format has three types of entries:

1. *Comment Lines:* Comment lines can appear anywhere in the file as long as the first character in the line is an asterisk.

2 *Year Designation Line:* This line should be the first data line (noncomment line) in the file; it must appear only once. It has three fields: year, prime time, and non-prime time. For example, to specify the year 1989, prime time at 9:00 a.m., and non-prime time at 4:30 p.m., enter:

```
1989  0900  1630
```

The time 2400 is automatically converted to 0000.

3. *Holiday Lines:* These entries follow the year designation line, and have the format:

day-of-year Month Day Description of Holiday

The day-of-year field is a number between 1 and 366, indicating the day for the corresponding holiday; leading blanks, tabs, and spaces are ignored. The other three fields are commentary, and are not currently used by other programs.

The accounting system will not function properly unless a **holidays** file exists. You will receive a **/usr/lib/acct/holidays.proto** as part of the DG/UX system. When you boot the system for the first time, **holidays.proto** will automatically be copied to **holidays**. Add entries to **holidays** as suits your needs. A sample file follows:

```
*
* Copyright (C) Data General Corporation, 1984 - 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.

*       $What: <@(#) chap15,v  1.4> $


* Prime/Nonprime Table for UNIX Accounting System

* Curr  Prime   Non-Prime
* Year  Start   Start

  1990  0830    1700

* Day of        Calendar        Company
* Year          Date            Holiday
```

```
  1          Jan 1          New Year's Day
148          May 28         Memorial Day
185          Jul 4          Independence Day
246          Sep 3          Labor Day
326          Nov 22         Thanksgiving
327          Nov 23         Day After Thanksgiving
358          Dec 24         Day Before Christmas Day
359          Dec 25         Christmas Day
```

Accounting Directories and Files

This last sections of the chapter briefly describe all the data files within the DG/UX accounting structure.

Files in the /var/adm Directory

The following files reside in the **/var/adm** directory.

diskdiag — Diagnostic output during the execution of disk accounting programs.

dtmp — Output from the **acctdusg** program.

fee — Output from the **chargefee** program, ASCII **tacct** records.

pacct — Active process accounting file.

pacct? — Process accounting files switched via **turnacct**.

Spacct?.*MMDD* — Process accounting files for *MMDD* during execution of **runacct**.

Files in the /var/adm/acct/nite Directory

The following files reside in the **/var/adm/acct/nite** directory.

active — Used by **runacct** to record progress and print warning and error messages.

cms — ASCII total command summary used by **prdaily**.

ctact.*MMDD* — Connect accounting records in binary.

ctmp — Output of acctconl program; connect session records in binary.

daycms — ASCII daily command summary used by **prdaily**.

dayacct	Total accounting records for one day in binary.
disktacct	Disk accounting records in binary; they are created by **dodisk**.
fd2log	Diagnostic output during execution of **runacct** (see **cron** entry).
lastdate	Last date **runacct** executed in *MMDD* format.
lock lock1	Used to control serial use of **runacct**.
lineuse	TTY line usage report used by **prdaily**.
log	Diagnostic output from **acctconl**.
log*MMDD*	Same as log after **runacct** detects an error.
reboots	Contains beginning and ending dates from **wtmp** and a listing of reboots.
statefile	Contains current state of a **runacct** run.
tmpwtmp	**wtmp** file corrected by **wtmpfix**.
wtmperror	Place for **wtmpfix** error messages.
wtmperror*MMDD*	Same as **wtmperror** after **runacct** detects an error.
wtmp.*MMDD*	Previous day's **wtmp** file.

Files in the /var/adm/acct/sum Directory

The following files reside in the **/var/adm/acct/sum** directory.

cms	Total command summary file for current fiscal year in internal summary format.
cmsprev	Command summary file without latest update.
daycms	Command summary file for yesterday in internal summary format.
loginlog	Created by **lastlogin**.
pacct.*MMDD*	Concatenated version of all **pacct** files for *MMDD*, removed after reboot by **remove** procedure.
rprt.*MMDD*	Saved output of **prdaily** program.

tacct	Cumulative total accounting file for current fiscal year.
tacctprev	Same as **tacct** without latest update.
tacct.*MMDD*	Total accounting file for *MMDD*.
wtmp.*MMDD*	Saved copy of wtmp file for *MMDD*; removed after reboot by **remove** procedure.

Files in the /var/adm/acct/fiscal Directory

The following files reside in the **/var/adm/acct/fiscal** directory.

cms	Total command summary file for a fiscal period.
fiscrpt	Report similar to **prdaily** for a fiscal period.
tacct	Total accounting file for a fiscal period.

End of Chapter

Appendix A
Device Names and Codes

This appendix explains the different forms that nodes take for all devices that the DG/UX system can use, including terminal controllers and lines, disks, tape drives, line printers, and pseudo-devices. The **/usr/etc/master.d** directory contains files that list *all* devices, pseudo-devices, protocols, and configuration parameters that are supported by the DG/UX system and any other packages you have installed. When you run **sysadm newdgux** to build a kernel for a system, you edit the **/usr/src/uts/aviion/Build/system.aviion** file to reflect what devices that you have on your system.

Accessing Devices

A node describes a logical device. A node creates a link between a physical medium and a logical unit accessed by a program. When you boot your system, the kernel creates all the necessary nodes in **/dev**, based on the entries in your system file. Every time you reboot your system, the kernel deletes all nodes in **/dev** and creates new nodes that correspond to all the devices configured in your kernel (described in the system file) and present on your system. Because nodes are removed at boot time, we recommend that you do not create them manually.

When a node is created at boot time, the kernel associates a node name with information including a major number, a minor number and whether the file is to be accessed as a block or character device. A *major number* tells the kernel what type of device is represented by the device file. Major numbers are assigned by Data General; they are listed in the master file in **/usr/etc/master.d/dgux**. You can change them by editing the master file, but we don't recommend that you do so unless you're writing your own device driver.

Minor numbers are allocated by the kernel. As administrator, you never have to worry about specifying a minor number. We explain them here so that you will know how the system uses them. A minor number tells the kernel exactly which physical or logical device unit is specified. For example, with terminal controllers, the minor numbers represent the tty lines. For logical disks, minor numbers represent the order in which logical disks have been added to the system, beginning with 0. Tapes are an exception. The minor number for a tape specifies which unit *and* what flags to apply to that tape (rewind, no rewind, density).

The major number is an index to a table in the kernel. This table contains sets of I/O subroutines called *device drivers*. Device drivers are either character-oriented or character-and-block-oriented so that a program can access a device in either character or block mode. For each type of device, there is a driver that invokes **open**, **close**, **read**, and **write** commands. Programs which need to do I/O call the kernel; the kernel invokes device drivers which use minor numbers to access a particular unit or device.

For disks, **/dev** contains subdirectories corresponding to character (raw) and block driver access for each device. Tape devices have only character-special nodes which are in **/dev/rmt**. Physical disks are defined in **pdsk** for block special and in **rpdsk** for character special. Logical disks are defined in **rdsk** (raw disk); and **dsk** (block disk).

The following list summarizes node names, major numbers, minor numbers, and device codes:

node name
: The node name is the name through which the device will be accessed. Nodes for some devices have names that include the hardware device code and physical unit number. You may link other names to the node with the **ln**(1) command.

major number
: The major number identifies the type of device. Every node that refers to a device of a given type will be created with the same major number. For example, all disk units on all **cied** type disk controllers have the same major number. Separate nodes are made for block access and character access to the same device; the major number is the same for each type of access.

minor number
: The minor number points to a specific device or location to be addressed. Within a given type of device, minor numbers start at 0 and proceed sequentially across all devices in that type.

  Tape drives are an exception: their minor number is derived from information that includes the unit number, density, and rewind option required.

device code
: To access a device, the operating system must be able to uniquely identify the device. This is done with device codes. Device codes are assigned by the operating system to all devices that post interrupts. Default device codes are listed later in this appendix.

As Figure A-1 illustrates, devices are connected to the system through device controllers and by direct line.

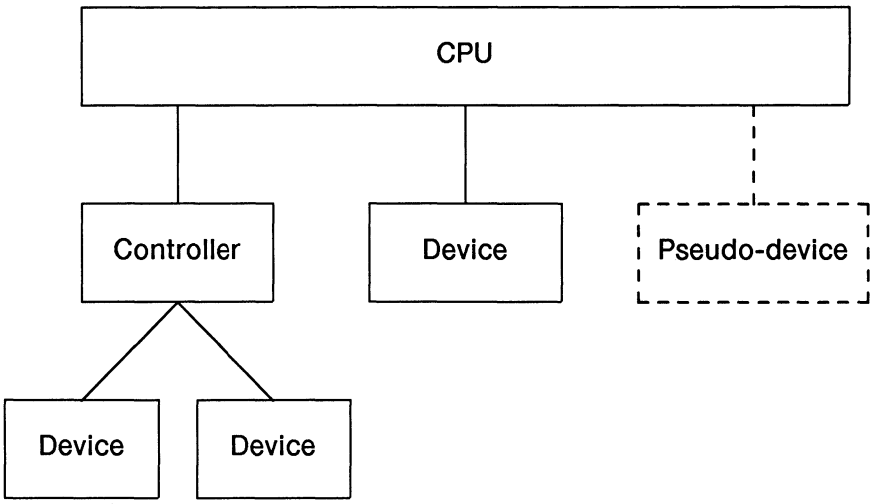


Figure A-1 *Device Connections*

Pseudo-devices are software constructs that the system uses as though they were devices. Some examples of pseudo-devices are:

/dev/mem	Physical Memory interface
/dev/kmem	Kernel memory interface
/dev/null	The null device
/dev/error	Error recording
/dev/prf	Operating system profiler
/dev/tty	Generic terminal interface; associates a tty line with a **login** process

System File Device Mnemonics

The following devices are supported by the DG/UX system:

con	The system console. This can be either a CRT or a hardcopy console.
cied	Ciprico ESDI controller: specified via the **cird** or **cied** driver in the system file.
cimd	Ciprico SMD controller: specified via the **cird** or **cimd** driver in the system file.
cisc	Ciprico SCSI adapter.

duart	Dual universal asychronous receiver transmitter; a serial communications device.
hken, inen	Intelligent LAN (local area network) controllers. These provide a communications interface, which is used by DG TCP/IP (DG/UX). **hken** is an Interphase Hawk controller.
insc	Integrated SCSI adapter.
ixe	IP to X.25 Encapsulation. A program that performs the translations required to use Internet Protocols (TCP/IP) over X.25 networks, as described in RFC 877.
meter	A pseudo-device that is used to monitor various processes in the kernel; not used by system administrators or general users.
prf	The operating system profiler, which provides access to system activity information. See **prf**(7).
pts, ptc	Pseudo terminals used by **telnet**(1), **shl**(1), and other applications. For each pseudo terminal connection, a master (**ptc**) and a slave (**pts**) are required.
sd	SCSI disk.
st	SCSI tape.
syac, sdcp	Systech terminal controller boards: **syac** is asynchronous, **sdcp** is synchronous.

Specifying Devices

A device specification names the physical devices by which your system communicates with peripherals, such as remote disks and tapes. A device specification contains a device driver name followed by a parenthesized list of optional parameters. The device name identifies the type of controller or device; the parameters provide additional information required to fully specify the peripheral.

The first optional parameter specifies a particular controller in configurations which may contain more than one controller of a given type. The second parameter almost always specifies a unit number for those controllers which support multiple units. Null or missing parameters, such as the first parameter in the specification **cied(,1)** or the second and subsequent parameters in **cied(0)** are interpreted as default values. The defaults are interpreted by the device driver itself; by convention, all numerical parameters default to zero. An asterisk as a parameter represents all possible values for the parameter.

The devices you use will depend on your hardware. For instance, AViiON workstations will use the integrated devices (**inen, insc**), but AViiON servers will not.

Device driver mnemonics are generally constructed from the first two letters of the manufacturer (Ciprico) or the manufacturer's name for the device (Hawk). The second two letters of the mnemonic usually stand for a device type specifier, such as **ed** for ESDI disk. Thus, **cied** represents the device driver program that allows a Ciprico ESDI disk to communicate with the operating system.

The parameter values are all zero-based. Therefore, **cied(1,2)** specifies the third drive of the second Ciprico ESDI controller in the system. The first drive on a system's first ESDI controller would be **cied()**.

AViiON System Specifications

The following tables show the specifications for devices on AViiON Systems. Tables A-1, A-2, and A-3 show the default entries for devices on the primary VME bus and the secondary SCSI bus. Note that the lines with entries such as cied@code stand for the nondefault situation. That is, when you have used the first and second defaults for cied, you will have to jumper your own device codes and select base addresses.

Tables A-4 and A-5 show the default base addresses for the entries in Tables A-1, A-2 and A-3. The lines with "set" entries mean that there are no default device codes or base addresses for these occurrences of the given device. You must set them yourself. Refer to your hardware documentation for details on setting your own device codes and base addresses.

Table A-1 Primary Bus (VME) AViiON System Devices

#	Device Name	Controller Number	Unit	3rd Param	Device Specification
1st	cied	0	0-3	-	cied(0,0)
2nd	cied	1	0-3	-	cied(1,0)
3rd	cied@code	address	0-3	-	cied@code(address,unit)
1st	cimd	0	0-3	-	cimd(0,0)
2nd	cimd	1	0-3	-	cimd(1,0)
3rd	cimd@code	address	0-3	-	cimd@code(address,unit)
1st	sdcp	0	0-1	-	sdcp(0,0)
1st	syac	0	0-15	-	syac(0,0)
2nd	syac	1	0-15	-	syac(1,0)
3rd	syac	2	0-15	-	syac(2,0)
4th	syac	3	0-15	-	syac(3,0)
5th	syac@code	address	0-15	-	syac@code(address,unit)
1st	hken	0	*ether_addr*	-	hken(0,*ether_addr*)
2nd	hken	1	*ether_addr*	-	hken(0,*ether_addr*)

Table A-2 Secondary Bus (SCSI) AViiON System Devices (SCSI Devices)

Device Name	Controller Number	ID	3rd Parameter	Device Specification
sd	See cisc below	0-3	-	sd(cisc (),0)
st	See cisc below	4-6	file	st(cisc (),0)

Table A-3 Secondary Bus (SCSI) AViiON System Devices (SCSI Adapters)

Device Name	Adapter	ID	3rd Parameter	Device Specification
1st cisc	0	-	-	cisc()
2nd cisc	1	-	-	cisc(1)

Table A-4 Default Base Addresses for AViiON System VME Devices

#	Device Name	Device Code	Base Address
1st	cied	18	FFFFEF00
2nd	cied	19	FFFFF100
3rd	cied@code	set	set
1st	cimd	18	FFFFEF00
2nd	cimd	19	FFFFF100
3rd	cimd@code	set	set
1st	sdcp	50	55B00000 and 55B01000
1st	syac	60	60000000
2nd	syac	61	60020000
3rd	syac	62	60040000
4th	syac	63	60060000
5th	syac@code	set	set
1st	hken	15	FFFF4000 and 55900000
2nd	hken	16	FFFF5000 and 55980000

Table A-5 Default Base Addreses for AViiON System SCSI Devices

#	Device Name	Device Code	Base Address
1st	cisc	28	FFFFF300
2nd	cisc	29	FFFFF500

AViiON Workstation Specifications

The tables in this section show the devices relevant to the DG/UX operating system running on AViiON Workstations. Table A-6 shows the primary device specifications for AViiON Workstations.

Table A-6 Primary (Integrated Bus) AViiON Workstation Devices

Device Name	Unit	3rd Parameter	Example Specification
inen	-	-	inen()
grfx	-	-	grfx()
lp	0-2	-	lp()
duart	-	-	duart()
kbd	-	-	kbd()

Table A-7 shows the secondary bus device specifications for AViiON Workstations.

Table A-7 Secondary Bus (SCSI) AViiON Workstation Devices

SCSI Type	Device Name	Controller	Adapter	SCSI ID	3rd Parameter	Example Specification
Device	sd	see insc below	-	0-3	-	sd(insc(),0)
Device	st	see insc below	-	4-6	file	st(insc(),4)
Adapter	1st insc	-	0	-	-	insc()

Nodes and Device Codes

The following sections give device code and node information for physical and logical disks, tapes, terminals, and all other devices supported by the DG/UX system.

Physical Disk Nodes

Each physical disk is accessible in either block or character mode. Block-access nodes are in **/dev/pdsk**. Character-access nodes are in **/dev/rpdsk**. The following are example entries created in **/dev**:

/dev/pdsk/cied@18(FFFFEE00,0)
/dev/rpdsk/cied@18(FFFFEE00,0)

/dev/pdsk/cied@18(FFFFEE00,1)
/dev/rpdsk/cied@18(FFFFEE00,1)

Logical Disks

Logical disks are accessible in either block or character mode. Therefore, the kernel creates both types of nodes. Block-access nodes are in **/dev/dsk** and character-access nodes are in **/dev/rdsk**. When you use **diskman** or **sysadm** to create a logical disk, a corresponding logical disk node is created. The minor number depends on the order a given disk was added. For instance, the first disk has minor number 0, the second has 1, and so on. If you create a logical disk named **comm**, the nodes in **/dev** will be:

/dev/dsk/comm
/dev/rdsk/comm

Tape Drive Nodes

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

/dev/rmt/*diskname***@***device_code*(*address*)**{lmh}**[**n**]

You select one of **l**, **m**, or **h** to indicate whether you are using a low, medium, or high density storage medium.

Use **n** to indicate that you do not want the device to rewind after the read or write.

The example device, **st(cisc)**, is a SCSI tape with a Ciprico SCSI adapter with default device code and address. The tape is at the default SCSI ID.

Terminal Nodes

Nodes are created in **/dev** from the information in your system file for each terminal line on your terminal controllers. The node names are in the form **tty***xxx*, where *xxx* corresponds to the minor number (line 5 is 05, line 100 is 100, and so on). All terminal nodes are character-access only.

AViiON Workstation tty Assignments

The kernel automatically configures the following ttys on AViiON workstations:

mouse tty00

duart async tty01

AViiON System tty Assignments

Depending on whether a cluster controller or a multiplexor (MUX) is present on the system, ttys are automatically configured as follows. If a cluster controller is present, the tty lines from 0 to 255 are all assigned no matter how many lines are on the cluster controller. **syac** lines are configured first, then **duart**.

If a MUX is present, tty lines are assigned 0-15, 16-31, 32-47, and 48-63, per MUX unit. **syac** lines are configured first, then **duart**.

End of Appendix

Appendix B
Directories and Files

This appendix lists the directories and files of the DG/UX system that are of interest to a system administrator. For a list of all directories and files shipped with the DG/UX system, look in the **/usr/release/dgux_*.fl** directory. For detailed information on any of the entries here, see the relevant manual reference page.

First, this appendix discusses the **root**, **var**, **usr**, and **srv** directories. Then it discusses some files of interest to a system administrator. Notice that some files and directories in the DG/UX system have *physical* locations and *logical* locations. A physical location is a file's real location. A logical location is a symbolic link. For instance, when you reference **/usr/spool/lp** (logical), you are actually referencing **/var/spool/lp** (physical). In this appendix, we denote a symbolic link by enclosing a directory in parentheses.

Contents of the Root (/) Directory

The **root** logical disk is mounted on a directory called **/**. We refer to those files on the **root** logical disk as "being in root." Remember that there are physical and logical directories on the **root** logical disk. They are:

(/bin)	Symbolic link to **/usr/bin**. Contains public commands.
(/lib)	Symbolic link to **/usr/lib**. Contains object libraries.
/admin	Home directory for the **sysadm** login.
/dev	Contains device nodes.
/etc	Contains configuration files and system data.
/usr/sbin/init.d	The **init.d** directory contains executable files used in upward and downward transitions to all system run levels. These files are linked to files beginning with **S** (start) or **K** (stop) in **/etc/rc*N*.d**, where *N* is the appropriate run level. Files are only executed from **/etc/rc*N*.d** directories.

/etc/rc*.d	There are eight of these directories in **/etc**: **rcS.d, rc0.d, rc1.d, rc2.d, rc3.d, rc4.d, rc5.d,** and **rc6.d**. Each directory contains links to all the shell scripts in **/etc/init.d**. See Chapter 3 for a complete list of all services that can be set for each run level.
/sbin	Contains minimum system commands to get the system up.
/tmp	Used for temporary system files.
/opt	Applications package parent directory.
/local	Contains site-specific files.
/tftpboot	Contains links to client's bootable executable files in **/usr/stand**.
/srv	A mount point for the **srv** logical disk. Contains client and release management files. See "Directories in /srv" later in this appendix.
/var	Used for system data files whose size varies as the system runs. See "Directories in /var" later in this appendix.

Contents of the /var Directory

The **/var** directory contains files that are release dependent, have read and write permissions set, and are dynamically sized.

/var/adm	This directory contains the data collection files for the accounting system. See Chapter 15 for complete information on accounting system files and directories.
/var/Build	Kernel builds by **sysadm newdgux** take place here.
/var/mail	Contains **mail** databases and dynamically-sized files.
/var/news	Contains **news** databases and dynamically-sized files.
/var/opt	Application package parent directory for dynamically-sized files.
/var/spool	Contains spooling files for **lp** and **uucp**.
/var/spool/lp	Contains all of the files and directories of the LP system. See Chapter 11 for complete information on these files and directories.

/var/spool/cron	Contains **cron** databases and dynamically-sized files.
/var/spool/uucp	Contains files specific to **uucp**.
/var/preserve	Text editor file save area for sudden program halt.
/var/tmp	User temporary file space.

Contents of the /usr Directory

The **usr** logical disk is mounted on the **/usr** directory. Files in this directory are release dependent and read-only. There are physical and logical directories on the **usr** logical disk. They are:

(/usr/spool)	Symbolic link to **/var/spool**.
(/usr/mail)	Symbolic link to **/var/mail**.
(/usr/adm)	Symbolic link to **/var/adm**.
(/usr/news)	Symbolic link to **/var/news**.
(/usr/ucb)	Symbolic link to **/var/ucb**.
(/usr/preserve)	Symbolic link to **/var/preserve**.
/usr/admin	Contains the files, directories, tables, menus, and defaults used by the **sysadm** system administration program.
/usr/bin	Contains user commands.
/usr/catman	Contains the on-line manual reference pages that users access with the **man**(1) command.
/usr/lib/gcc	Contains the DG/UX GNU C compiler. For information see the Release Notice that comes with the compiler package.
/usr/etc	Contains database and configuration files.
/usr/etc/master.d	Contains master files. These files list devices and kernel parameters for the DG/UX system. This directory may also contain master files for other packages that have kernel components, such as TCP/IP and NFS.
/usr/include	Contains include files for system software.
/usr/lib	Contains library routines.

/usr/lib/acct	Contains the C language programs and shell procedures that drive the accounting system. See Chapter 15.
/usr/local	Contains site-specific, read-only files.
/usr/release	Contains media notices, release notices, and system package names.
/usr/opt	Contains applications packages.
/usr/sbin	Commands used only by an administrator.
(/usr/share)	Symbolic link to **/srv/share**.
/usr/src	Parent directory for source code.
/usr/src/uts/aviion/lb	Contains the kernel libraries which are used to build the kernel image. See Chapter 4. Also see **config**(1M) and **make**(1).
/usr/stand	Contains stand-alone utilities and bootstrap programs.
(/usr/tmp)	Symbolic link to **/var/tmp**.

Contents of the /srv Directory

The **/srv** directory contains the directories and files needed for managing operating system releases and clients.

/srv/admin	Contains **sysadm** databases and information files.
/srv/admin/clients	Contains **sysadm** client data.
/srv/admin/defaults	Contains **sysadm** defaults for releases and clients.
/srv/admin/releases	Contains **sysadm** OS release data.
/srv/dump	Dump space on a one-per-client basis.
/tftpboot	Contains links to bootstraps for diskless clients.
/srv/release	Contains space for each release's **usr** and client roots.
/srv/release/PRIMARY	Contains symbolic links to the server's **usr** and **/** files.
/srv/share	Contains release independent shared software.
/srv/swap	Swap space on a one-per-client basis.

DG/UX Administrative Files

This section is not intended to be an exhaustive look at the DG/UX files; rather, it highlights the files that you'll be using more often than others. Subsections here are:

- **/etc** Database Files Maintained via **sysadm**
- **/etc** Database Files Maintained Manually
- Administrative Commands in the **/sbin** Directory
- Administrative Files in the **/usr** Directory
- Administrative Files in the **/var** Directory

/etc Database Files Maintained via sysadm

The **/etc** directory contains a number of files that are databases of information needed to support various subsystems. The following sections describe the files in **/etc** that **sysadm** maintains.

/etc/fstab

The **fstab** file specifies the file system(s) to be mounted by the **/etc/mount** command. The following is a sample entry in **fstab**. Note that it is in ONC/NFS format; we recommend this format even if you are not using ONC/NFS.

```
/dev/dsk/root  /  dg/ux  rw  d  1
```

The above entry indicates a local file system mount, and the following entry indicates an NFS remote file system mount.

```
titan:/usr/titan  nfs  rw,hard  x  0
```

The **fstab** format was changed to support ONC/NFS file systems as well as local file systems. The old style **fstab** entries are also supported. See **fstab**(4) for detailed information.

/etc/gettydefs

The **/etc/gettydefs** file contains information that **/etc/getty** uses to set the speed and terminal settings for a line. The **getty** command accesses the **gettydefs** file with a label (typically the baud rate). The general format of the **gettydefs** file is as follows:

```
label# initial-flags # final-flags #login-prompt #next-label
```

Each line entry in the **gettydefs** file is followed by a blank line. Refer to **gettydefs**(4) for more information. A typical **/etc/gettydefs** file follows:

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
console# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3 #Console Login:
```

/etc/group

The **/etc/group** file describes each group to the system. An entry is added for each new group. Each entry in the file is one line and consists of four fields, which are separated by a colon (:):

```
group name:password:group id:login names
```

See "Adding Groups" in Chapter 14 and **group**(4) for more information. If you have NFS, see **yppasswdd**(1M).

/etc/inittab

The **/etc/inittab** file contains instructions for the **/etc/init** command. The instructions define the processes that are to be created or terminated for each initialization state. Initialization states are called run levels. By convention, run level S is single-user mode; run level 1 is administrative mode; and run levels 2 and 3 are multi-user modes. Chapter 3 summarizes the various run levels and describes their uses. See **inittab**(4) for more information.

/etc/passwd

The **/etc/passwd** file identifies each user to the system. Add an entry for each new user. Each entry in the file is one line and consists of seven fields. The fields are separated by colons (:). The fields are:

```
login_name:password:uid:gid:comment:home_directory:program
```

Example:

```
poulet:Rm27oQak1:103:104:L.Q. Poulet:/usr/poulet:/bin/csh
```

See "The User's Environment" in Chapter 14 and **passwd**(4).

/etc/TIMEZONE

The **/etc/TIMEZONE** file sets the time zone shell variable TZ. The TZ variable in the **TIMEZONE** file is changed by the **sysadm datetime** command. The TZ variable can be redefined on a user (login) basis by setting the variable in the associated **.profile** or **.login**. See Chapter 4.

/etc/dgux.params

This file contains parameters that you can set to control the actions of **rc** scripts in **/etc/init.d**.

/etc/log

Contains log information on run level changes and daemon activity.

/etc/nfs.params

This file contains parameters for controlling ONC/NFS programs.

/etc/tcpip.params

This file contains the parameters for commands invoked by the **rc** scripts to initialize the network. Chapter 3 describes the **rc** scripts in detail.

/etc/dgux.rclinktab

This data table can be executed by **rc.links** to create or remove links from the **rc#.d** directories to the **/usr/sbin/init.d** directory.

/etc/dgux.prototab

A file listing all DG/UX prototype files in **/etc**.

/etc/dumptab

This file contains the dump table which lists the different media supported by **dump2**(1M). It describes the media characteristics for each medium made available to **dump2**. See **dumptab**(4).

/etc/inetd.conf

Contains the Internet server configuration database. This is a list of servers that **inetd** invokes when it receives an Internet request over a socket.

/etc Files Maintained Manually

The **/etc** directory contains a number of files that are databases of information needed to support various subsystems. The following sections describe the files in **/etc** that **sysadm** does not maintain.

/etc/login.csh

The default profile for **csh** users is the **/etc/login.csh** file. The default profile for **sh** users is the **/etc/profile** file. The standard (default) environment is established by the instructions in these global profile files. See "The User's Environment" in Chapter 14 for more details.

/etc/devlinktab

This files contains entries used to make short-named links to device nodes with otherwise unwieldy names. An example follows:

```
/dev/rmt        0        st(insc@99(0,0,0),4)
```

/etc/motd

The **/etc/motd** file contains the message of the day. The message of the day is output by instructions in the **/etc/profile** file after a successful login. This message should be kept short and to the point.

/etc/profile

The default profile for **sh** users is in the **/etc/profile** file. The default for **csh** users is the **/etc/login.csh** file. The standard (default) environment is established by the instructions in these global profile files. See Chapter 14 for examples.

/etc/utmp

The **/etc/utmp** file contains information on the run level of the system. This information is accessed with a **who −a** command.

/etc/wtmp

The **/etc/wtmp** file contains a history of system logins. The owner and group of this file must be **adm**, and the access permissions must be 664. Each time **login** is run this file is updated. As the system is accessed, this file increases in size. Periodically, this file should be cleared or truncated. The command line **>/etc/wtmp** when executed by **root** creates the file with nothing in it. The following command line limits the size of the **/etc/wtmp** file to the last 3600 characters in the file:

```
# tail -3600c /etc/wtmp > /tmp/wtmp; mv /tmp/wtmp /etc/wtmp ↵
```

Administrative Commands in the /sbin Directory

The following commands are available in **/sbin**. These are the minimum system administration commands necesary to get a system running.

/sbin/init

Command to change run levels S, 1, 2, 3.

/sbin/sh

The DG/UX **sh**(1) command.

/sbin/su

Switches user. For instance **su sysadm** switches you to the **sysadm** login.

/sbin/fsck

Runs the **fsck**(1M) file system check program.

/sbin/mount

Mounts a file system on the DG/UX directory tree.

/sbin/umount

Unmounts a file system.

/sbin/chk.fsck

An **rc** check script to run the **fsck** program.

/sbin/shutdown

Brings the operating system down to single-user mode (run level S). See Chapter 4.

/sbin/halt

Halts the operating system to the SCM prompt.

/sbin/setup.d/boot

Sets up scripts that must run on the host CPU.

/sbin/setup.d/root

Sets up scripts that modify a host's **root** space.

/sbin/rc.init

Executes the shell scripts in **/etc/init.d** via links in **/etc/init/rc*N*.d**. Execution is initiated from entries in **/etc/inittab**. For example, the following line specifies that all scripts associated with run level 3 be executed:

```
rc3:3:wait:/sbin/rc.init 3
```

Administrative Files in the /usr Directory

The following files are available in the **/usr** directory.

/usr/lib/cron/log

A history of all actions taken by **/etc/cron** is recorded in the **/usr/lib/cron/log** file. The **/usr/lib/cron/log** file should be periodically truncated to keep the size of the file within a reasonable limit. The following command line limits the size of the log file to the last 100 lines in the file:

```
# tail −100 /usr/lib/cron/log > /tmp/log; mv /tmp/log /usr/lib/cron/log
```

/usr/sbin/init.d

This directory contains the system's check scripts and **rc** scripts.

/usr/sbin/setup.d/usr

This directory contains set up scripts that modify a host's **usr** space. Setup scripts might include those for TCP/IP and NFS.

/usr/src/uts/aviion/cf/system.*.proto

This file contains your custom version of the devices and configuration parameters listed in **/usr/etc/master.d/dgux**. For configuration, the **config**(1M) program runs on the **system** file and produces program code in **conf.c**. Prototype system files are shipped with software packages with kernel content. Prototype files have names of the form **system.*.proto**.

Administrative Files in the /var Directory

The following files are located in the **/var** directory. These files are useful for monitoring superuser activity and performing recurring system administrative tasks.

/var/adm/sulog

The **/var/adm/sulog** file contains a history of superuser (**su**(1)) command usage. As a security measure, this file should not be readable by others. The **/var/adm/sulog** file should be periodically truncated to keep the size of the file within a reasonable limit. The following command lines limit the size of the log file to the last 100 lines in the file:

```
# tail −100 /usr/adm/sulog > /tmp/sulog ♪
# mv /tmp/sulog /usr/adm/sulog ♪
```

A typical **/usr/adm/sulog** file follows:

```
SU 08/18 16:16 + console smitht-root
SU 08/18 23:45 + tty00 jones-root
SU 08/19 11:53 + console smitht-sysadm
SU 08/19 15:25 + console root-sysadm
SU 08/19 23:45 + tty00 root-uucp
```

/var/spool/cron/crontabs

The **cron** function is useful for doing recurring and habitual system administration tasks. The **/var/spool/cron/crontabs** directory contains crontab files for **adm, root,** and **sys** logins. Providing their lognames are in the **/usr/lib/cron/cron.allow** file, users can establish their own **crontabs** file using the **crontab** command. If the **cron.allow** file does not exist, the **/usr/lib/cron/cron.deny** file is checked to determine if the user is denied the use of the **crontab** command.

As **root** or **sysadm**, you can either use the **crontab**(1) command or edit the appropriate file under **/var/spool/cron/crontabs** to make the desired entries. Revisions to the file take effect at the next reboot. Refer to **crontab**(1) for additional information.

End of Appendix

Appendix C
LP and UUCP Error and Status Messages

This appendix contains error and status messages for the LP and UUCP systems. For information on system call errors (*errno* values), see **intro**(2). PANIC error messages are shipped on-line and are located in **/usr/release/dgux_4.30.panic**. You should print out a paper copy of the PANIC error messages and and keep it handy in case of system failure.

LP Error and Status Messages

This section provides a description of the error and status messages that are associated with LP commands. The following variables are used in the LP error messages:

file(s)	Indicates the file or files that are to be printed.
dest	Indicates the name of the destination printer.
request id	Indicates the request identifier of the print job. The identifier is the printer name followed by a request identification number; for example, *newlp-10*.
printer name	Indicates the name of the printer.
program name	Indicates the program name that was executed.
user	Indicates the user who requested the printout.

Following each message is an explanation of the probable cause of the error and the corrective action to take. If you are not able to correct all the error conditions you encounter, call your service representative for assistance.

LP Error Message	Description/Action
acceptance status of destination *printer name* unknown	Use the **sysadm acceptlp** command to enable the printer so that it will accept requests.
dest is an illegal destination name	The *dest* you used is not a valid destination name. Use the **lpstat −p** command to list valid destination names.
file is a directory	File *file* is a directory and cannot be printed.
xx is not a request id or a printer	The argument you used with the **cancel** command is not a valid request identification number or a printer name. Use the **lpstat −t** command to give you all the printers and requests waiting to get printed.
xx is not a request id	The request identification number you used with the **lpmove** or **sysadm movelp** command is not a valid request identification number. To find out what requests are valid, use the **lpstat −u** command.
xx not a request id or a destination	You used an invalid request identification number or destination. To find out what is valid, use the **lpstat −t** command.
dest not accepting requests since *date*	The printer you are trying to use is in reject mode.
Can't access FIFO	The named pipe file **/var/spool/lp/FIFO** is incorrect. The mode should be 600.
can't access file *file*	The mode could be wrong on your directory or the file that you are trying to access.
can't create class *xx*–it is an existing printer name	The class name you are trying to use has already been given to a printer. You will have to use another name or remove the printer to use the class name.
can't create new acceptance status file	The mode may be wrong on the **/var/spool/lp** directory. It should be 755 with the owner **lp** and the group **bin**.

LP Error Message	Description/Action
can't create new class file	The mode may be wrong on the **/var/spool/lp** directory. It should be 755 with the owner **lp** and the group **bin**.
can't create new interface program	The mode may be wrong on the **/var/spool/lp/interface** directory. It should be 755 with the owner **lp** and the group **bin**.
can't create new member file	The mode may be wrong on the **/var/spool/lp/member** directory. It should be 755 with the owner **lp** and the group **bin**.
can't create new printer status file	The mode may be wrong on the **/var/spool/lp/pstatus**. It should be 644 with the owner **lp** and the group **bin**.
can't create new request directory	The mode may be wrong on the **/var/spool/lp/request** directory. It should be 755 with the owner **lp** and the group **bin**.
can't create printer *printer name* -- it is an existing class name	The printer name you are trying to use has already been used as a class name. You will have to assign another name for the printer. Use **sysadm modlp**.
can't move request *request id*	*Printer id* is the request identification number that cannot be moved. You will probably have to change the modes on the files and directories in **/var/spool/lp/request**. Also, you will have to manually move the request from the disabled printer directory to the new destination after you shut down the LP scheduler.
can't create new output queue	The mode on the file **/var/spool/lp/seqfile** is incorrect. It should be 644, and the mode on the directory should be 755. The owner should be **lp**, and the group should be **bin**.
can't create new sequence number file	The mode on the file **/var/spool/lp/seqfile** is incorrect. The mode of the file should be 644, and the mode of the directory should be 755. The owner should be **lp**, and the group should be **bin**.

LP Error Message	Description/Action
can't create request file *xx*	The mode on the file **/var/spool/lp/request/***printer_name/request_id* is incorrect. *Printer_name* is the name of the printer such as **dqp10**, and *request_id* is the request identification number. The mode of the file should be 444, and the mode of the directory should be 755. The owner should be **lp**, and the group should be **bin.**
can't fork	You either have several processes running and are not allowed to run any more, or the system has all the processes running that it can handle. You will have to rerun this command later.
can't lock acceptance status	This is a temporary file in **/var/spool/lp** that prevents more than one **lp** request from being taken at any given instant. You will have to rerun this command later.
can't lock output queue	The file **/var/spool/lp/QSTATLOCK** prevents more than one **lp** request from being printed on a printer at a time. You will have to rerun this command later.
can't lock printer status	The temporary file **/var/spool/lp/PSTATLOCK** prevents more than one **lp** request from being printed on a printer at a time. You will have to rerun this command later.
can't lock sequence number file	The file **/var/spool/lp/SEQLOCK** prevents more than one **lp** request from getting the next *request id* number at a time. Try this command again later.
can't open class file	The **lp** program is trying to access the list of classes for printers. One reason it may not be able to open the class file is that the system could have the maximum number of files open that are allowed at any time. Try the command again later.
can't open member file	The **lp** program is trying to access the list of members in the directory **/var/spool/lp/member**. The system could have the maximum number of files open that are allowed at any time. Try this command again later.

LP Error Message	Description/Action
can't open *file* file in class directory	One possibility why file *xx* cannot be opened is that the mode on the file or directory is incorrect. The file mode should be 644, and the directory mode should be 755. Another possibility is that the system has the maximum number of files open that are allowed at any time. The latter problem can be corrected by trying the command again later.
can't open *printer name*	You cannot print on printer *printer name* because the mode is incorrect on **/dev/tty**. The mode should be 622.
can't open FIFO	The mode on the named pipe file **/var/spool/lp/FIFO** may be incorrect. It should be 600. Or, the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by trying the command again later.
can't open default destination file	Check the mode on the file **/var/spool/lp/default**. The mode should be 644. If the mode is okay, it could be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open file *file*	The file *file* was incorrectly typed or you don't have the correct modes set. The mode should be at least 400 if you are the owner.
can't open output queue file	Check the mode on the file **/var/spool/lp/outputq**. It should be 644. This error message could also be generated if the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open printer status file	The mode on the file **/var/spool/lp/pstatus** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open request directory *directory name*	The mode on the directory **/var/spool/lp/request** is incorrect. The mode should be 655. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.

LP Error Message	Description/Action
can't open request file *request id*	The mode on the file **/var/spool/lp/member/request/***xx* is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command again later.
can't open system default destination file	The mode on the file **/var/spool/lp/default** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't open temporary output queue	The mode on the file **/var/spool/lp/outputq** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. Try this command again later.
can't proceed -- scheduler running	Many of the **lpadmin** command options cannot be executed while the scheduler is running. Stop the scheduler using the **lpshut** command and then try invoking the command again.
can't read current directory	The **lp** and **lpadmin** commands cannot read the directory containing the file to be printed. The directory name may be incorrect or you do not have read permission on that directory.
can't remove printer	The mode may be wrong on the **/var/spool/lp/member** directory. It should be 755, and the files in that directory should be 644. Both the directory and the files should be owned by **lp** and the group should be **bin.**
can't remove request directory	The mode may be wrong on the **/var/spool/lp/request** directory. It should be 755 and should be owned by **lp**, and the group should be **bin.** The directory may still have pending requests to be printed which will have to be removed before the directory can be removed.
can't set user ID to LP Administrator's user ID	The **lpsched** and **lpadmin** commands can only be used when you are logged in as root.

LP Error Message	Description/Action
can't write to *device*	The **lpadmin** command cannot write to device *device*. The mode is probably wrong on the **/dev/tty***xx* file. It should be 622 and owned by **lp**.
cannot create temp file *file*.	The system may be out of free space on the **/var** file system. Use the command **df /var** to determine the number of free blocks. Several hundred blocks are required to insure that the system will perform correctly.
class *xx* has disappeared!	Class *xx* was probably removed since the scheduler was started. The system may be out of free space on the **/var** file system. Use the command **df /var** to find out. Use the **lpshut** or **sysadm stoplp** command to stop the scheduler and restore the class from a backup.
class *xx* non-existent	The class *xx* may have been removed because the system is out of free space on the **/var** file system. Use the command **df /var** to find out how much free space is available. The class will probably have to be restored from a backup.
class directory has disappeared!	The **/var/spool/lp/class** directory has been removed. The system may be out of free space on **/var**; use the **df /var** command to find out. The class directory contains all the data for each printer class. To restore this directory, get these files and directory from a backup.
corrupted member file	The **/var/spool/lp/member** directory has a corrupted file in it. You should restore the directory from backup.
default destination *dest* non-existent	Either the default destination is not assigned or the printer *dest* has been removed. Use **lpadmin** or **sysadm defaultlp** to set up a default destination.
destination *dest* has disappeared!	A destination printer, *dest*, has been removed since **lpsched** was started. Use **sysadm dellp** to remove the printer.
destination *printer name* is no longer accepting requests	The printer is not accepting requests because the **reject** or **sysadm rejectlp** command has been invoked. Use **sysadm acceptlp** or **accept** to make the printer accept requests.

LP Error Message	Description/Action
destination *dest* non-existent	The destination printer you specified as an argument to the **accept** or **lpadmin** command is not a valid destination name, or it has been removed since the scheduler was started.
destination *printer name* was already accepting requests	The destination printer was previously enabled. Once a printer is accepting requests, issuing any more **accept** commands to it are ignored.
destination *printer name* was already not accepting requests	A **reject** command was already sent to the printer. Use the **accept** command to allow the printer to start accepting requests again.
destination *printer name* is not accepting requests move in progress ...	The printer has been disabled by the **reject** command, and requests are being moved from the disabled printer to another printer. The printer can be enabled again by the **accept** command.
destinations are identical	When using the **lpmove** command, you need to specify a printer to move the print requests from and a different printer to move the requests to.
disabled by scheduler: login terminal	The login terminal has been disabled by the LP scheduler. The printer can be reenabled by using the **enable** or **sysadm enablelp** command.
error in printer request *request id*	*Printer id* is the actual request identification number. The error was most likely due to an error in the printer. Check the printer, and reset it if needed.
illegal keyletter *xx*	An invalid option, *xx*, was used. For the correct options, See the manual page for the command you were using when you received this error.
keyletters **−***xx* and **−***yy* are contradictory	This combination of options to the **lpadmin** program cannot be used together.
keyletter *xx* requires a value	The option *xx* requires an argument. For example, in the command line **lpadmin −m** *model* the argument to the **−m** option is the name of a model interface program.

LP Error Message	Description/Action
keyletters **−e**, **−i**, and **−m** are mutually exclusive	These options to the **lpadmin** command cannot be used together. Refer to the manual page for more information.
LP Administrator not in password file	You must have an entry in the **/etc/passwd** file for **lp**, and you must belong to the group **bin**.
model *xx* non-existent	The name that you are using for a model interface program is not a valid one. A list of valid models is in the **/var/spool/lp/model** directory.
new printers require −v and either −e, −i, or −m	A printer must have an interface program, and this is specified by **−e**, **−i**, or **−m** options. The **−v** option specifies the device file for the printer. For more information on these options, refer to the **lpadmin** manual page.
no destinations specified	There are no destination printers specified. Use the **lpadmin** command to set one up.
no printers specified	There are no printers specified. Use the **lpadmin** command to set one up.
non-existent printer *xx* in PSTATUS	A printer with the name *xx* is in the **/var/spool/lp/pstatus** file but no longer exists. The printer should be removed using the **lpadmin** command.
non-existent printer in class *xx*	The printer that you are trying to address in class *xx* has been removed from that class.
out of memory	Implies the system is in trouble. The message implies that there is not enough memory to contain the text to be printed.
printer *printer name* already in class *xx*	The printer you are trying to move to class *xx* is already in that class. You cannot move a printer to a class that it is already in.

LP Error Message	Description/Action
printer *printer name* has disappeared!	The printer has been removed, and the **enable** command cannot find it. The printer was most likely removed since the machine was rebooted or since the scheduler was started.
printer *printer name* non-existent	*Printer name* is the name of a printer that has been removed since the scheduler has been started. You must use the **lpadmin −x***printer name* command.
printer status entry for *printer name* has disappeared	The **/var/spool/lp/pstatus** file must have been corrupted. You will have to resubmit the printer request.
printer *printer name* was not busy	The printer is not printing a request at this time. Either the request you wanted to cancel is finished printing or you have specified the wrong printer.
request *request id* non-existent	You are attempting to cancel a request that does not exist. You may have given the wrong printer name or wrong request id number or the request may have finished printing.
request not accepted	The request was not accepted by **lp**. The scheduler may not be running. Use the **lpstat −t** command to find out more information.
requests still queued for *printer name* use lpmove	*Printer name* is the printer that still has requests waiting to get printed. You need to use the **lpmove** command to get those requests moved to another printer.
scheduler is still running -- can't proceed	You cannot perform this command while the scheduler is running. You will have to use the **lpshut** command first.
spool directory non-existent	The directory **/var/spool** has been removed. You will have to use the **mkdir** command to restore the directory. This has probably removed some of the necessary LP files. You may have to reinstall the LP commands.
standard input is empty	You specified an invalid file name either by incorrectly typing a name or by specifying a nonexistent file. Nothing will be printed on the printers from this request.

LP Error Message	Description/Action
this command for use only by LP Administrators	This command is restricted to someone logged in as **root** or **lp**.
too many options for interface program	The **lp** command called the appropriate interface program with too many arguments. For more information on the options and arguments that can be used with the **lp** command, refer to the **lp** manual page.
unknown keyletter *xx*	An invalid option was supplied to the **lp** or **lpadmin** command.
unknown option *xx*	This message is displayed in response to an invalid option supplied to the **disable, lpstat,** or **reject** commands. Refer to Chapter 11 of this manual.
usage: disable [−c] [−r[reason]] printer ...	The syntax for the **disable** command is not correct. The valid options are: **−c** to cancel the currently printing request, and **−r** followed by the reason that you are disabling the printer.
usage: reject [−r[reason]] dest ...	The syntax for the **reject** command is not correct. The proper format is to specify the reason why the printer is not taking any more print requests and to identify the destination printer.
usage: accept dest	The syntax for the **accept** command is not correct. You did not specify what printer should accept requests.
usage: enable printer	The syntax for the **enable** program is to specify a destination printer.
usage: cancel id .... printer	The syntax for the **cancel** command is not correct. The proper format is to specify the request identification number or the printer name.

LP Error Message	Description/Action
usages: lpadmin −pprinter [−vdevice] [−cclass] [−rclass] [−eprinter | −iinterface | −mmodel] [−h | −l] -or- lpadmin −d[destination] -or- lpadmin −xdestination	The correct syntax for the **lpadmin** command is to specify at least one of the options mentioned above.
your printer request id was canceled by user	The printer request did not finish printing because another *user* canceled it. Typically, you will get this message in your mail. One reason a person may cancel a request other than their own is because the request is not printing correctly.

UUCP Error and Status Messages

This section lists two types of error and status messages associated with UUCP connections.

- ASSERT errors are recorded in the **/var/spool/uucp/.Admin/errors** file.
- STATUS errors are recorded in individual machine files found in the **/var/spool/uucp/.Status** directory.

ASSERT Error Messages

When a UUCP process fails, ASSERT error messages may be generated and recorded in **/var/spool/uucp/.Admin/errors**. These messages include the file name, sccsid, line number, and the text listed below. ASSERT messages reflect conditions which the system manager must correct. These errors are usually due to system problems.

Assert Error Message	Description/Action
BAD LINE	There is a bad line in the **Devices** file; there are not enough arguments on one or more lines.
BAD SPEED	A bad line speed appears in the **Devices/Systems** files (Class field).
BAD LOGIN_UID	The uid cannot be found in the **/etc/passwd** file. The file system is in trouble, or the **/etc/passwd** file is inconsistent.
BAD UID	Same as previous.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T CHDIR	A chdir() call failed.
CAN'T CHMOD	A chmod() call failed.
CAN'T CREATE	A create() call failed.
CAN'T CLOSE	A close() or fclose() call failed.
CAN'T FORK	An attempt to **fork** and **exec** failed. The current job should not be lost, but will be attempted later (**uuxqt**). No action need be taken.
CAN'T LINK	A link() call failed.
CAN'T LOCK	An attempt to make a LCK (lock) file failed.
CAN'T OPEN	An open() or fopen() failed.
CAN'T READ	A read(), fgets(), etc. failed.
CAN'T STAT	A stat() call failed.
CAN'T WRITE	A write(), fwrite(), fprint(), etc. failed.
CAN'T UNLINK	An unlink() call failed.
WRONG ROLE	This is an internal logic problem.

Assert Error Message	Description/Action
FILE EXISTS	The creation of a work file (**C.***file*) or data file (**D.***file*) file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.
ULIMIT TOO SMALL	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
SYSLST OVERFLOW	An internal table in **gename.c** overflowed. A big/strange request was attempted.
TOO MANY FILES	Same as previous.
RETURN FROM fixline ioctl	An ioctl, which should never fail, failed. There is a system driver problem.
PERMISSIONS file: BAD OPTION	There is a bad line or option in the **Permissions** file. Fix it immediately!
PKCGET READ	The remote machine probably hung up. No action need be taken.
PKXSTART	The remote machine aborted in a non-recoverable way. This can generally be ignored.
SYSTAT OPEN FAIL	There is a problem with the modes of **/etc/uucp/.Status**, or there is a file with bad modes in the directory.
TOO MANY LOCKS	There is an internal problem! Contact your Data General Representative.
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were suppose to be checked before this process was attempted.

STATUS Messages

Status messages are stored in the **/var/spool/uucp/.Status** directory. This directory contains a separate file for each remote machine that your system attempts to communicate with. These files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common messages that may appear in these files.

Status Message	Description/Action
ASSERT ERROR	An ASSERT error occurred. Check the **/var/spool/uucp/.Admin/errors** file for the error message.
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login/machine name that does not agree with the **Permissions** file.
CALLBACK REQUIRED	The called machine requires that it calls your DG/UX system.
CALLER SCRIPT FAILED	This is usually the same as "DIAL FAILED." However, if it occurs often, suspect the caller script in the **dialers** file. Use **Uutry** to check.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the **Systems** and **Devices** files.
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
DEVICE FAILED	The open of the device failed.
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the Dialer-Token-Pairs script.

Status Message	Description/Action
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the **Devices** file for the particular system. Check the **Systems** file for the device to be used to call the system.
OK	Things are OK.
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your computer in its **Systems** file.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your computer. They could be trying to call your machine. If they have an older version of UUCP, the process that was talking to your machine may have failed leaving the LCK file. Check to see if the process that has a LCK file is hung.
REMOTE REJECT AFTER LOGIN	The login used by your computer to login does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your computer for an unknown reason. The remote machine may not be running a standard version of UUCP.
STARTUP FAILED	Login succeeded, but initial handshake failed.
SYSTEM NOT IN Systems	The system is not in the **Systems** file.
TALKING	Local UUCP and remote UUCP are communicating successfully.
WRONG TIME TO CALL	A call was placed to the system at a time other than what is specified in the **Systems** file.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.

End of Appendix

Appendix D
File System Checking: fsck

The **fsck**(1) program is a multipass file system check and repair program. Each file system pass invokes a different phase of the **fsck** program. You must use this program when when you are bringing up your system after an abnormal shutdown such as a power outage or system crash.

The **fsck** program checks, in order, blocks and file sizes, directory contents, connectivity, link counts and resource allocation, and disk allocation region (DAR) information, including the free-block bitmap, the free-inode list, and summary counts. The program reports any inconsistencies. It is your option to fix or ignore them.

This appendix:

- Discusses the normal updating of the file system.
- Discusses the possible causes of file system corruption.
- Presents the corrective actions taken by **fsck**. It describes both the program and the interaction between the program and the system administrator.
- Contains the **fsck** error conditions, giving their meanings, possible responses to them, and related error conditions.

File System Update

Every time a file is modified, the DG/UX operating system performs a series of file system updates. When written to disk, these updates yield a consistent file system.

There are five types of file system updates. The updates involve the following areas:

- the superblock
- inodes
- index (indirect) blocks
- data blocks (directories and other files)
- disk allocation region information, which includes the free-block bitmap and the inode table

Corrupted File Systems

Many things can corrupt a file system. Improper shutdown procedures and hardware failures are the most common causes.

Some examples of improper shutdown procedures are:

- Forgetting to use the **shutdown**(1M) command (which unmounts all file systems, including the root) before halting the CPU.
- Physically write protecting a mounted file system.
- Taking a mounted file system off line.

Each DG/UX file system contains a flag in the superblock which indicates whether or not the file system is mountable. You can only mount a file system if it is marked as mountable; if a file system is unmountable, you must first run **fsck** in order to repair inconsistencies. The **fsck** program will mark a file system mountable only when it is internally consistent.

A file system is marked mountable when it is created. It is marked unmountable whenever it is mounted, and is not marked mountable again until it is either unmounted or has passed the **fsck** program's internal consistency checks. Therefore, file systems which were still mounted at the time of an abnormal system shutdown cannot be remounted until **fsck** has been run over them, whereas those file systems which were cleanly unmounted before shutdown can immediately be remounted.

Fixing Corrupted Files

This section discusses ways to discover and fix inconsistencies for different kinds of update requests.

The **fsck** program lets you check a file system for structural integrity by performing consistency checks on redundant data. Redundant data is either read from the file system or computed from other known values. When **fsck** reports an inconsistency, it asks whether the inconsistency is to be corrected by repairing or deleting the corrupted item. You can accept or reject this request. In the following example, we invoke **fsck**. The program finds an incorrect link count in Phase 4 and asks if it should fix the problem.

```
# fsck /dev/dsk/mydisk ↵

** /dev/dsk/mydisk:
** Phase 1 - Check Blocks and File Sizes
** Phase 2 - Check Directory Contents
** Phase 3 - Check Connectivity
** Phase 4 - Check Link Counts and Resource Accounting

Inode 67 (owner: 2 [bin]; group: 2 [bin]; size: 52736 bytes;
    type: Ordinary; mode: 755; mtime: Fri Nov 20 17:54:36 1987)
    has incorrect link count (2 should be 1) -- fix?  y ↵

** Phase 5 - Check Disk Allocation Region Information
File system is now mountable.

13936 of 50000 blocks used (36064 free); 288 of 5822 inodes
    used (5534 free).

#
```

After we respond **y**, the **fsck** program corrects the incorrect link count that it found for inode 67 and moves on to Phase 5. When it finishes, it reports that the file system is mountable.

Superblock and Disk Allocation Region Information

The superblock and disk allocation region information are some of the most commonly corrupted items. Every change to the file system's blocks or inodes modifies the superblock and the disk allocation region information. The superblock and disk allocation region are most often corrupted when the system was not properly shut down with the **shutdown**(1M) command.

Superblock and disk allocation region inconsistencies can involve file system size, the number of available inodes and blocks, the free-block bitmaps and the free inode lists. The following sections give brief discussions of these sections.

Free-Block Bitmap

Each disk allocation region (DAR) contains a bitmap representing all the blocks in the DAR. The **fsck** program compares that information with its own map of allocated blocks, looking for inconsistencies that suggest DAR corruption.

Free-Inode List

Each DAR contains a link list of free inodes in that DAR. The **fsck** program traverses that list to ensure that all free inodes from that DAR are in the list, and that no allocated inodes are in it.

Summary Counts

The superblock and each DAR contain several counts: the number of used inodes, the number of used blocks, the number of directories. The **fsck** program compares these counts to the information it has compiled.

Inodes

An individual inode is less likely than the superblock to be corrupted. However, because of the great number of active inodes, the free inode lists are as susceptible as the superblock to corruption. The **fsck** program checks for inconsistencies involving format and type, link count, duplicate blocks, and inode size.

Format and Type

Each inode contains mode information. This information describes the type of the inode. Inodes may be one of eight types: regular, directory, control point directory, special block, symbolic link, special character, FIFO, or socket. Any other type is illegal.

Link Count

Each inode contains a count of the directory entries linked to the inode. The **fsck** program verifies the inode count by checking down the total directory structure, starting from the root directory, and calculating an actual link count for each inode. For more information about inodes and file systems, see Chapter 9.

When the link count (which is stored on the disk) is nonzero and the actual link count (kept by **fsck**) is zero, no directory entry appears for the inode. If no entry appears, **fsck** may link the disconnected file to the **/lost+found** directory.

If the stored and actual link counts are nonzero and unequal, **fsck** may replace the link count on the disk by the actual link count. When this situation arises, a directory entry may have been added or removed without the inode being updated.

Duplicate Blocks

Each inode contains a list and sometimes pointers to lists (index blocks) of all the blocks claimed by the inode. The **fsck** program checks these lists for duplicate blocks. Duplicate blocks can occur when a file system uses blocks claimed by both the free-block bitmap and other parts of the system or when two or more inodes claim the same block.

Any block claimed more than once is flagged by **fsck** as a duplicate block. If there are any duplicate blocks, **fsck** makes a partial second pass of the inode list to find the inode of the duplicated block. If the files associated with these inodes are not examined for correct content, **fsck** will not have enough information to decide which inode is corrupted and should be cleared. Usually, the inode with the earliest modification time is incorrect and should be cleared.

Size Checks

Each inode contains a size field. This field's size indicates the number of bytes in the file associated with the inode. The **fsck** program can check the size for inconsistencies, such as directory sizes that are not a multiple of 512 bytes, or a mismatch between the number of blocks actually used and the number indicated by the inode size. The **fsck** program also checks for directory corruption, where conflicting information is found within the directory entries.

The **fsck** program can also perform a check of the size field of an inode. It uses the size field to compute the number of blocks that should be associated with the inode, and then compares that number to the actual number of blocks claimed by the inode.

Control Point Directories

The root inode of a file system is a special type of directory known as a *control point directory*. A control point directory is like an ordinary directory except that it has resource limits associated with it for inodes and for data blocks. The total resources consumed by the control point directory and all its descendants (to which it is the *space parent*) may not exceed the size limits.

Index Blocks

Index blocks (also known as indirect blocks) are owned by an inode. Therefore, inconsistencies in an index block directly affect the inode that owns the block. The **fsck** program can check inconsistencies involving blocks already claimed by another inode and block numbers outside the range of the file system.

Data Blocks

There are two types of data blocks: plain data blocks and directory data blocks. Plain data blocks contain the information stored in a file. Directory data blocks contain directory entries. The **fsck** program does not try to check a plain data block.

The **fsck** program can check each directory data block for:

- bad self-identification information
- directory entries for unallocated inodes
- directory entries for inodes which do not exist in the file system
- directories that are disconnected from the file system

If a directory entry inode number points to an unallocated inode, **fsck** may remove that directory entry. Directory entry inode numbers can point to unallocated inodes when the data blocks containing the directory entries are modified and written out, but the inode is not written out.

If a directory entry inode number is pointing to a non-existent inode, **fsck** may remove that directory entry. This condition occurs if bad data is written into a directory data block.

The **fsck** program checks that all directories are linked into the file system; i.e., they have a parent directory pointing to them (except for the root). If **fsck** finds unlinked directories, it links the directory back into the file system in the **/lost+found** directory. When inodes are being written to the file system without the corresponding directory data blocks being written to the file system, the directories are not linked into the file system.

Invoking the fsck Program

You can invoke **fsck** one of four ways:

Startup script The startup script is most common way of invoking **fsck**. When you are in multi-user mode bringing up the system with the **init** command, you can automatically execute **fsck** from within your startup script.

Command line From the command line, type:
fsck *[options] [file_system_names]*
where *options* are single character flags that modify the behavior of the command and *file_system_names* refer to the file systems that you want to check. The *options* are covered in detail a little later in this appendix.

Initialization The initialization version of **fsck** is built into the operating system kernel and is automatically run over the root file system when you boot your system.

Stand-alone The stand-alone method is the least common way to invoke **fsck**. You will use the stand-alone version of **fsck** via the stand-alone **diskman** menu. See Chapter 7 of this manual.

Options to fsck

All options are represented by single-character flags; options must begin with a hyphen. All options except for **−t** are Boolean flags, and may thus be combined. For example, you can combine the options **-p**, **-x**, and **-D** as follows: **fsck −pxD**.

The following options are interpreted by **fsck**:

−p Detect all possible inconsistencies, but correct only those inconsistencies that may be expected to occur from an abnormal system halt. For each corrected inconsistency, one or more lines will be printed identifying the file system and the nature of the correction. Any other inconsistencies will cause the check of that file system to fail. The following 15 inconsistencies (and only those listed) will be corrected for the specified file systems:

1. An inode has an incorrect count of the blocks it uses. The count is corrected.

2. An inode is partially truncated. Partial truncation can occur if the system is abnormally halted while a file is being truncated, leaving the file claiming more data blocks than its size in bytes would require. The extra blocks are freed.

3. A directory has an incorrect child count. The count is corrected.

4. A directory entry exists for an inode which is unallocated. The directory entry is removed.

5. A directory entry's file name length is incorrect. The length is corrected.

6. An inode is unreferenced (has no directory entries anywhere in the file system). The inode is reconnected in the **/lost+found** directory.

7. No **/lost+found** directory exists, but an inode needs to be reconnected there. The **/lost+found** directory is created.

8. The root directory needs to be expanded in order to make room for a **/lost+found** directory entry. The directory is expanded.

9. The **/lost+found** directory needs to be expanded in order to make room for a directory entry for an inode being reconnected there. The directory is expanded.

10. An inode's link count is incorrect. The count is corrected.

11. The root control point directory's resource accounting (blocks, inodes) is incorrect. The counts are corrected.

12. A disk allocation region (DAR) has an incorrect free-block bitmap. The bitmap is corrected.

13. A DAR has an incorrect free-inode list. The list is corrected.

14. A DAR has incorrect summary counts of used blocks, inodes or directories. The counts are corrected.

15. The summary counts in the superblock are incorrect. The counts are corrected.

−q Repair the inconsistencies listed under the **−p** option automatically, without asking for user approval. Unlike **−p** however, more serious inconsistencies will not cause **fsck** to fail; the user must still answer the resulting queries.

−y Audit and interactively repair all file system inconsistencies assuming a "yes" response to all questions asked by **fsck**.

CAUTION: Use this option with great care, since it could lead to irreversible changes to the file system.

−n Audit and interactively repair all file system inconsistencies, assuming a "no" response to all questions asked by **fsck**. This option also means that all file systems will be opened with read-only intent.

−x Look at a file system's superblock to see if it is marked mountable. If so, do not check the file system for inconsistencies. If the file system is marked unmountable, check it.

−s Ignore the actual free-block bitmap and unconditionally reconstruct a new one.

−S Conditionally reconstruct the free-block bitmap. A free-block bitmap is reconstructed if and only if the file system is consistent. This option also forces a "no" response to all **fsck** questions.

−t Use the specified scratch file for temporary storage if **fsck** cannot obtain enough memory. The scratch file's name must be the next argument after **−t**.

−D Directories are checked for bad blocks.

−f Fast check: blocks and sizes are checked; the free block bitmap is reconstructed if necessary.

The following options are mutually exclusive, and use of more than one per invocation is not allowed: **−y**, **−n**, **−p**, **−q**, and **−S**.

Arguments to fsck Options

The file system(s) to be checked may be specified either implicitly or explicitly. If no arguments are given, the file systems to be checked may be found in one of two special files: **/etc/checklist** or **/etc/fstab**. If **/etc/checklist** exists, then every entry in it is checked in order. If **/etc/checklist** does not exist, but **/etc/fstab** does, then all the file systems listed with a non-zero pass number and a "rw" or "ro" mounting status are checked. If the **−p** option was specified, the checking occurs in order of pass number, with those file systems of equal pass number being checked in parallel with each other. Otherwise, checking occurs in order of appearance in **fstab**.

If arguments are specified (the rest of the command line after the option flags), those file systems, and only those file systems, are checked sequentially in the order given.

File systems may be specified as arguments to **fsck** in one of two ways: by the special device file (in **/dev/dsk** or **/dev/rdsk**) containing the file system; or by the directory that **/etc/fstab** indicates will serve as the mount point for the file system.

Checking File Systems

File system checking proceeds without any input from the operator if no errors are discovered. When a fatal inconsistency is discovered, no further checking is done on that file system; the **fsck** program either exits or proceeds to the next specified file system. When an inconsistency is discovered with the **−p** option, and that error is one of those listed under **−p**, the inconsistency is fixed without operator approval. Any other discoveries of inconsistencies require the operator to make a decision. The **fsck** program prompts with its recommended action. If you answer **yes**, then **fsck** takes the recommended action. In no case will any damaging action be taken without approval. Note, however, that advance approval or disapproval may be given by invoking **fsck** with the **−y** and **−n** options, respectively.

The **fsck** program will refuse to check any file system for which any of the following conditions hold true:

- The file system is mounted
- The special file associated with the file system cannot be opened.
- The specified path name (or its device node associate in **/etc/fstab**) is not a block-special, character-special, or regular file whose size can be determined.

The **fsck** program checks for the following inconsistencies (the term "Bad format" refers to system blocks which do not have the required self-identification information).

- Unreadable or inconsistent superblocks.
- Bad format in superblocks.
- Invalid contents in superblock's reserved area.
- Bad value for superblock's file system size.
- Bad value for superblock's DAR size.
- Bad value for superblock's inode/DAR density.
- Bad value for superblock's default data element size.
- Bad value for superblock's default index element size.
- Bad value for superblock's default directory data element size.
- Bad value for superblock's default directory index element size.
- Bad value for superblock's default first allocation threshold.
- Bad value for superblock's default second allocation threshold.
- Bad format in inode table block.
- Invalid contents in inode's reserved area.
- Files of unknown type.
- Files with bad fragment size.
- Files which are partially truncated.
- Files claiming impossible blocks.
- Files claiming system-area blocks.

- Bad Index-block format.
- Files with incorrect block counts.
- Files claiming already-claimed blocks.
- Unallocated root inode.
- Bad file type for root.
- Incorrect resource limit information in root.
- Incorrect parent directory in root.
- Directories with "holes" (unallocated blocks before end-of-file).
- Bad format in directory blocks.
- Directories with invalid information in reserved areas.
- Directories with empty blocks at end.
- Directories with incorrect child counts.
- Extra directory entries named "." or ".."
- Directory entries with invalid characters in file names: "/" or non-ASCII characters.
- Directory entries which would have too-long path names.
- Directory entries which are out of order.
- Directory entries with incorrect entry lengths.
- Directory entries with incorrect file name lengths.
- Extraneous hard links to directories (including cycles in file system name space).
- Extraneous hard links to Symbolic Link files.
- Directory entries to invalid inodes.
- Directory entries to unallocated inodes.
- Files with incorrect space parent.
- Unconnected files or directories.
- Bad or missing **lost+found** directories.
- Bad **lost+found** directory entries.

- Root or **lost+found** directories needing expansion.
- Files with incorrect link counts.
- Incorrect resource allocation counts in control point directories.
- Bad format in DAR blocks.
- Invalid contents in reserved area of DAR blocks.
- Incorrect free-block bitmaps in DARs.
- Incorrect or incomplete free-inode lists in DARs.
- Incorrect DAR summary counts: blocks used, inodes used, directories used.
- Incorrect superblock summary counts.

fsck Output

If the **–p** option is used, **fsck** prints out one or more lines for each inconsistency it corrects, indicating the file system fixed and the error corrected. After successfully checking or correcting a file system, **fsck** prints out the name of the file system, the number of files on it, and the number of free and used blocks.

If the **–p** option is not used, **fsck** is more verbose. It will first print out the name of the file system. Then **fsck** prints a message as it enters each phase of checking a file system. A message is printed for each inconsistency encountered, and the operator is prompted for approval before each correction is attempted. (If the **–y** or **–n** flags are used, **fsck** automatically answers such prompts itself.) When checking is complete for the file system, a message is printed if any corrections were made. Finally, the numbers (used, free and total) of files and blocks are printed.

The **fsck** program attempts to give as much information as possible about any files for which you must make decisions (such as whether to remove it, etc.). At least the following information will always be displayed:

- I-number, which is a number specifying a particular inode on a file system
- Owner's user ID
- Owner's group ID
- File type
- Mode
- Size
- Time of last modification

When possible, the following additional information will be displayed:

- Path name
- Owner's user name
- Owner's group name

fsck Error Conditions

When **fsck** detects an inconsistency, it reports the error condition to the operator. If a response is required, **fsck** prints a prompt message and waits for a response. This appendix explains the meaning of each error condition, possible responses, and related error conditions.

The error conditions are organized by the phase of the **fsck** program in which they can occur. The error conditions that may occur in more than one phase are discussed under "General Error Messages."

The following error messages are presented in their basic form. Fatal errors (such as during **fsck −p**) cause the error message to be prefaced by the string `Fatal Error:`. Running with **−p** also causes messages to be preceded by the name of the file system to which the message applies. The following abbreviations appear in the description of error messages:

B	A (decimal) disk block number.
N	A decimal number.
O	An octal number.
C	A character.
D, *F*	A directory name, file name or path name string.
I	An inode description string. At the very least, this will consist of the inode number. If possible, the inode's size, file type, file mode, UID, GID, time of last modification, owner name, group name and path name will also be present.

General Error Messages

The messages described in this section may appear at any time during an **fsck** session.

Cannot allocate memory for internal tables (N bytes requested)

The **fsck** program cannot allocate enough memory; this can only occur during stand-alone **fsck**. The **fsck** program will abort. Bring up your system and use the **fsck** command instead.

Cannot read block B

A disk read of block number *B* has failed. The **fsck** program treats the block it could not read as if it were filled with all zeroes, and continues execution, but the file system is not marked as mountable upon conclusion of checking. Use **diskman** to remap the bad block *B* and run **fsck** again.

Cannot write block B

A disk write of block number B has failed. The **fsck** program continues execution, but the file system being checked is not marked as mountable upon conclusion of checking. Use **diskman** to remap the bad block *B*, and run **fsck** again.

Fork failed

The **fsck** program has failed in an attempt to spawn a child process. This will only occur when running **fsck** with the **−p** option. The only file system affected will be the one for which the child **fsck** process was being created; no check will occur.

Internal Software Error: Cannot seek to block B -- aborting

A disk seek to block number *B* has failed; this should never happen. Contact your Data General support representative if this message is displayed. The **fsck** program terminates.

Invalid response; please answer yes or no

An invalid answer has been entered in response to one of **fsck**'s questions. The **fsck** program will not continue until a valid response has been entered. The following strings are valid responses: **y, Y, yes, YES, n, N, no** and **NO**.

Errors During fsck Invocation

Before starting to check a file system, **fsck** must parse its command line and determine which files to check. The following messages result from command line errors or information in the file **/etc/fstab**.

The directory D is the mount point for F

The **fsck** program has been given a directory *D* to check and has determined that *D* is the mount point for the file system *F*. This message is purely advisory.

The flags −y, −n, −p, −q and −S are all mutually exclusive

More than one of the above flags has been specified on the **fsck** command line. At most one of them is allowed. The **fsck** program will abort.

Unknown option: −C

An unknown option flag, *C*, has been specified on the **fsck** command line. Valid flags are as follows: **−y**, **−n**, **−p**, **−q**, **−t**, **−D**, **−f**, **−s**, **−S**, and **−x**. When you give it an invalid option, **fsck** will abort.

Errors During fsck Initialization

Before a file system check can be performed, **fsck** must set up certain tables and open certain files. The following messages can result from errors during this phase.

Block B is invalid Inode Table Block -- rewrite as empty block?

The inode table block *B* does not contain the proper self-identification information. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `rewrite as empty block?` prompt are:

YES Fix this error condition by rewriting this block as an empty file node table block. Any inodes that formerly occupied slots in this block will be cleared.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Cannot determine disk size of F

The **fsck** program has been given a file system *F* to check, but the size of *F* cannot be determined. This should never happen. Contact your Data General support representative if this message is displayed. The **fsck** program will abort checking this file system.

Cannot find a readable copy of the superblock

Neither of the two copies of the superblock can be read. The **fsck** program will abort checking this file system.

Cannot find a valid copy of the superblock

Neither of the two copies of the superblock contain the required self-identification information. The **fsck** program will abort checking this file system.

Cannot open F for reading

The **fsck** program has been given a file system *F* to check, but *F* cannot be opened for reading. Check the mode of *F*. The **fsck** program will abort checking this file system.

Cannot open F for writing

The **fsck** program has been given a file system *F* to check, but *F* cannot be opened for writing. Check the mode of *F* and make sure that no disks containing the file system are physically write-disabled. The **fsck** program will abort checking this file system.

Cannot read superblock copy N

One of the two superblock copies cannot be read. The **fsck** program will attempt to use the other copy and continue.

F is not a regular file, block-special file, character-special file or valid mount point

The **fsck** program has been given a file system *F* to check, but *F* is not of the correct type. *F* must be a file of type ordinary, block-special or character-special, or else it must be listed in the file **/etc/fstab** as a valid mount point directory. The **fsck** program will abort checking this file system.

File system is too large to check

Stand-alone **fsck** cannot allocate enough memory for its internal tables to begin checking the file system. The **fsck** program will abort checking this file system. Bring up your system and use the **fsck** command instead.

File system size stored in superblock is incorrect (N1 blocks should be N2) -- fix?

The superblocks contain an incorrect file system size figure. If run with the **−p** or **−q** options, **fsck** will automatically correct this. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the file system size to N2, the actual size of the disk containing the file system.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Invalid default Data Element Size exponent: N -- fix?

The default data element size for files (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 4 (meaning an element size of 16 blocks).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default data element size's exponent to 4.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Directory Data Element Size exponent: N -- fix?

The default data element size for directories (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 4 (meaning an element size of 16 blocks).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default directory data element size's exponent to 4.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Directory Index Element Size exponent: N -- fix?

The default index element size for directories (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 0 (meaning an element size of 1 block).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default directory index element size's exponent to 0.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid default Index Element Size exponent: N -- fix?

The default index element size for files (stored in the superblocks as a base-2 logarithm) is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to set the size's exponent to the default of 0 (meaning an element size of 1 block).

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the default index element size's exponent to 0.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid Disk Allocation Region size: N blocks

The DAR size stored in the superblocks is invalid. The **fsck** program will abort checking this file system.

Invalid first allocation threshold file size: N -- fix?

The superblocks contain an invalid first allocation threshold file size (the number of blocks a file can allocate in its initial DAR before all subsequent allocations are made from a different DAR). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the first allocation threshold file size to the default limit for DARs of the size specified in the superblock.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Invalid number of inodes per Disk Allocation Region

The number of inodes per DAR stored in the superblocks is invalid. The **fsck** program will abort checking this file system.

Invalid second allocation threshold file size: N -- fix?

The superblocks contain an invalid second allocation threshold file size (the number of blocks a file can allocate in a non-initial DAR before all subsequent allocations are made from a different DAR). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to correct the size.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the second allocation threshold file size to the default limit for DARs of the size specified in the superblock.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

No check necessary for F

The file system F is already marked mountable and **fsck** was invoked with the **-x** flag. The **fsck** program will not check this file system.

Superblock copies differ; using newer copy

Both copies of the superblock are readable and both contain the required self-identification information, but they differ. The **fsck** program will use the first copy (which is guaranteed to be more recent) and continue.

Superblock copy N is invalid

One of the two superblock copies does not contain the required self-identification information. The **fsck** program will attempt to use the other copy and continue.

Superblock has invalid contents in reserved area -- fix?

A copy of the superblock has non-zero contents in a reserved area. If running with the **-p** flag, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES The superblock's reserved area is initialized so that it contains all 0s.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 1 - Check Blocks and File Sizes

This phase of **fsck** operation is concerned with inodes. The following messages result from errors in inode types, inode format, file sizes and the data element pointers and index element pointers that make up a file's structure.

Incorrect block count in Inode I (N1 should be N2) -- fix?

The inode I's count of the blocks it uses is incorrect. If run with the **-p** option, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct the count.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting inode I's block count to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I claims an invalid block (B) -- clear bad pointer?

The inode I claims block B, which does not exist. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES Fix this error condition by clearing the pointer in inode I that claims the non-existent block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I claims a system block (B) -- clear bad pointer?

The inode I claims block B, which is a system block (a bitmap block, file node table block, DAR entry table block or superblock). If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the system block.

Possible responses to the `clear bad pointer?` prompt are:

YES Fix this error condition by clearing the pointer in inode I that claims the system block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has an Index Block (B) with invalid format -- clear bad pointer?

The inode I claims block B as an index block, but block B does not contain the proper self-identification information. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the element pointer claiming the invalid block.

Possible responses to the `clear bad pointer?` prompt are:

YES Fix this error condition by clearing the pointer in inode I that claims the index block. This may result in a "hole" in the file if the cleared pointer was before the last block of the file.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has invalid contents in its reserved area -- fix?

The inode I does not contain the proper self-identification information. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the reserved area.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by initializing inode *I*'s reserved area.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has invalid fragment size exponent (N) -- clear?

The inode I has a disallowed exponent representing the size of the file's fragment. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES Fix this error condition by clearing inode I.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Inode I is of unknown file type (O) -- clear?

The inode I is of type *O*, which is an unrecognized octal number. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES Fix this error condition by clearing inode I.

NO Ignore this error condition. The **fsck** program will abort checking this file system.

Inode I is partially truncated -- fix?

The inode I's size is shorter than the number of blocks allocated to it. If run with the **−p** option, **fsck** will automatically complete the truncation. Otherwise, **fsck** will ask to complete truncating inode I.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by completing the truncation of inode I down to the size stored in the inode.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 1b - Resolve Duplicate Claims

When **fsck** finds a block claimed by two or more files, it rescans the file system to find the original claimant of that block. This section lists the error messages that result from settling the claim to the disputed block.

Inode I claims another file's blocks -- clear?

The inode I claims some blocks that belong to another file. **fsck** will ask to clear the file.

Possible responses to the `clear?` prompt are:

YES Fix this error condition by clearing inode I.

NO Ignore this error condition. This will result in the same question being asked about the next claimant of the disputed block. As long as enough files are eventually cleared to resolve the duplicate claims on block B, **fsck** will continue normally. However, if at the end of Phase 1b any duplicate claims still exist, **fsck** will not mark this file system as mountable upon completing the check.

Errors During Phase 2 - Check Directory Contents

This phase is concerned with the contents of directories. The messages in this section result from improperly formatted directory blocks, an improperly formatted root directory, and bad directory entries. During this phase, all bad entries and inodes are removed from the file system tree.

Directory inode I has a hole -- fix?

The directory inode *I* has at least one "hole" in its file structure (gaps before the end of file). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rearrange the directory blocks to fill in the hole.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by rearranging the blocks in the directory to eliminate the hole.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I has incorrect child count (N1 should be N2) -- fix?

The directory inode *I*'s count of children, *N1*, is incorrect. If run with the **−p** or **−q** options, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct the child count.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting inode *I*'s child count to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I has an invalid block (B) -- rewrite as empty block?

The directory inode *I* has a block (address *B*) which does not contain the proper self-identification information. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `rewrite as empty block?` prompt are:

YES Fix this error condition by rewriting block *B* as an empty directory block. Any directory entries that formerly occupied this block will be destroyed.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 of invalid size -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry is too long, too short, or is not a multiple of 4 bytes in size. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 which is out of order -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2* which has a bad sequence number, meaning that the entry is invalid. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with filename of invalid size -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's file name is too long or too short. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2 with an illegal filename: F -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name F is "." or ".." These two names are reserved for the directory's links to itself and to its parent, respectively. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has a filename with an illegal character, octal value O -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name includes the illegal character *O*. A character is disallowed if it is non-ASCII or it is the slash character. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has an illegally long pathname -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode *I2*, but the path name for that entry relative to the root of the file system would exceed **MAXPATHLEN** (1024) bytes. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode I2, which has invalid contents in its reserved area -- fix?

The directory inode *I1* has a directory entry for inode *I2*, which has non-zero information in its reserved area. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the contents of the reserved area of inode *I2*.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by initializing the reserved area of inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode number I2, which is invalid -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is not a valid inode number. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry for inode number I2, which is unallocated -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is not an allocated inode. If run with the **-p** option, **fsck** will automatically remove the directory entry for inode *I2*. Otherwise, **fsck** will ask to remove the directory entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry which is an extraneous link to directory inode I2 -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is a directory which does not list *I1* as its parent directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode I2.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has entry which is an extraneous link to symbolic link inode I2 -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I2* is a symbolic link which already has another hard link. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I1 has an entry (for inode I2) which crosses a control point directory boundary -- remove bad directory entry?

The directory inode *I1* has a directory entry for inode number *I2*, but *I1* and *I2* have different space parent control point directories. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the directory entry for inode *I2*.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the directory entry for inode *I2*. If inode *I2* is an allocated inode with no remaining links, there will be an opportunity to reattach it in the **/lost+found** directory during Phase 3.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Incorrect filename length in directory inode I1 for directory inode I2 (N1 should be N2) -- fix?

The directory inode *I1* has a directory entry for inode *I2*, but the entry's name length, *N1*, is incorrect. If run with the **−p** option, **fsck** will automatically correct the directory entry's name length to *N2*. Otherwise, **fsck** will ask to correct the name length.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the directory entry's length to *N2* bytes.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode is of wrong file type -- fix?

The root inode (inode 2) is not a control point directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to fix the incorrect file type.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the file type of inode 2 to type control point directory.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode is not allocated -- fix?

The root inode (inode 2) is not allocated. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to allocate inode 2.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by allocating inode 2 as the root.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's parent directory is not the root -- fix?

The root inode's parent directory is not the root (itself). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to list the root inode as its own parent.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the root inode's parent directory to itself.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's space parent control point directory is not the root -- fix?

The root inode's space parent control point directory is not the root (itself). If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to list the root inode as its own space parent.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the root inode's space parent control point directory to itself.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Root inode's space usage limit is too large (N1 should be N2) -- fix?

The root inode's space usage limit, *N1*, is bigger than the size of the file system, *N2*. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset the limit to *N2* blocks.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by setting the root inode's space usage limit to *N2* blocks.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 3 - Check Connectivity

Phase 3 of **fsck** deals with the reconnection of unreferenced files and directories onto the file system tree. The messages in this section result from attempts to connect unreferenced files into the **lost+found** directory. Note also that any of the Phase 2 messages may be seen in this phase, as the contents of any reconnected directories must be checked.

Cannot find enough contiguous free blocks to expand directory inode I

The **fsck** program could not find enough contiguous free blocks to expand the directory inode I. Some unreferenced files may not reconnected as a result of this failure; they can be reconnected during a later **fsck** session after enough space has been freed in the file system.

Control point directory inode I has an entry named 'lost+found' which is not a directory -- remove bad directory entry?

The control point directory inode I already has an entry named **/lost+found**, but which is not of type directory. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the bad entry from inode I. The bad entry's inode will itself be reattached in the new **/lost+found** directory which will be created in directory I1.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Could not reconnect inode I

The **fsck** program was unable to reconnect the unreferenced inode I because it could not allocate enough blocks to expand the **/lost+found** directory, or because it could not allocate a free inode to use as the **/lost+found** directory.

Directory inode I is already as large as it can become

The **fsck** program has discovered that a directory it was attempting to expand is already the maximum size a directory can become.

Inode I1 lists as its space parent inode number I2, which is not a valid control point directory -- reset space parent to root?

The inode I1 has the non-control point directory inode I2 listed as its space parent. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to reset I1's space parent to inode 2, the root of the file system.

Possible responses to the `reset?` prompt are:

YES Fix this error condition by resetting I1's space parent to inode 2, the root of the file system.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Directory inode I needs to be expanded -- fix?

The directory inode I needs to be expanded so that another directory entry can be added to it; I is either the root directory or the **/lost+found** directory. If run with the **−p** or **−q** options, **fsck** will automatically attempt to expand the directory. Otherwise, **fsck** will ask to expand it.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by attempting to expand inode I.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I is unreferenced -- clear?

The inode I has no links in the file system and an earlier reconnection failed or was refused.

Possible responses to the `clear?` prompt are:

YES Fix this error condition by clearing inode I. The contents of the file will be destroyed.

NO Ignore this error condition. Inode I will remain unattached and can be reattached during a later **fsck** session provided that enough blocks and/or inodes are free.

Inode I is unreferenced -- reconnect?

The inode I has no links in the file system. If run with the **-p** option, **fsck** will automatically attempt to reconnect the file. Otherwise, **fsck** will ask to reconnect it.

Possible responses to the `reconnect?` prompt are:

YES Fix this error condition by reconnecting inode *I* in the **/lost+found** directory, with the name "*#N*", where *N* is the inode number of I.

NO Ignore this error condition.

The lost+found directory inode I already has an entry named 'F' -- remove bad directory entry?

The **/lost+found** directory inode I has discovered that it already has an entry of the name F when it was trying to reconnect an unreferenced file which would have had the same name. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to remove the spurious entry.

Possible responses to the `remove bad directory entry?` prompt are:

YES Fix this error condition by removing the entry for F; the inode referred to by that entry will be reattached with a name constructed from its inode number.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 4 - Check Link Counts and Resource Accounting

This phase checks the link counts of individual inodes and the resource counts (blocks and inodes used) of control point directories. The messages result from errors in these counts.

Control point directory inode I has incorrect inode allocation count (N1 should be N2) -- fix?

The control point directory inode I has a bad count of the inodes used by it and all its space descendants. If run with the **−p** or **−q** options, **fsck** will automatically adjust the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by adjusting the inode count for inode *I* to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Control point directory inode I has incorrect space allocation count (N1 should be N2) -- fix?

The control point directory inode I has a bad count of the blocks used by it and all its space descendants. If run with the **−p** or **−q** options, **fsck** will automatically adjust the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by adjusting the space count for inode I to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Inode I has incorrect link count (N1 should be N2) -- fix?

The inode I has a bad link count. If run with the **−p** or **−q** options, **fsck** will automatically adjust the count to *N2*. Otherwise, **fsck** will ask to fix the count.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by adjusting the link count for inode I to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Errors During Phase 5 - Check Disk Allocation Region Information

This phase deals with the disk allocation regions. Messages in this section result from errors in the components of the DARs: the bitmap, the free inode list, and various resource counts.

Block B of the Disk Allocation Region Information Area is invalid -- fix?

The disk allocation region information area block B does not contain the proper self-identification information. If run with the **-p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to rewrite the block.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by rewriting this block as an empty disk allocation region information area block. The DAR information in the block will be corrected later in this Phase.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect Bitmap -- fix?

The bitmap for DAR N is incorrect. If run with the **-p** option, **fsck** will automatically correct the bitmap. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by rewriting the bitmap correctly.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect count of blocks used (N1 should be N2) -- fix?

The block count for DAR *N* is incorrect. If run with the **-p** or **-q** options, **fsck** will automatically correct the count to *N2*. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by changing DAR *N*'s block count from *N1* to *N2*.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect counts of directories and inodes used -- fix?

The counts of used files and directories for DAR N are incorrect. If run with the **−p** or **−q** options, **fsck** will automatically correct the counts. Otherwise, **fsck** will ask to correct them.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by rewriting the counts of used inodes and directories correctly.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has incorrect free inode list -- fix?

The linked list of free inodes in DAR number N is incorrect: it contains allocated inodes, duplicates, or it does not contain some inodes which are actually unallocated. If run with the **−p** or **−q** options, **fsck** will automatically correct the free list. Otherwise, **fsck** will ask to correct it.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by rewriting the free list for DAR number N.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Disk Allocation Region N has invalid contents in its reserved area -- fix?

Disk allocation region number N has non-zero contents in its reserved area. If run with the **−p** option, **fsck** will abort checking this file system. Otherwise, **fsck** will ask to zero out the reserved area.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by initializing the contents of the reserved area.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Incorrect summary counts in superblocks -- fix?

The counts of used blocks and files in the two copies of the superblock are incorrect. If run with the **−p** or **−q** options, **fsck** will automatically correct the counts. Otherwise, **fsck** will ask to correct them.

Possible responses to the `fix?` prompt are:

YES Fix this error condition by rewriting the counts of used blocks and files correctly.

NO Ignore this error condition. The **fsck** program will not mark this file system as mountable upon completing the check.

Advisory Messages During File System Cleanup

Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system.

File System is now mountable

The **fsck** program has successfully completed checking the file system and it has been marked as mountable.

File System is still inconsistent and not mountable

The **fsck** program has completed checking the file system, but inconsistencies remain and the file system is still marked as unmountable. Re-run **fsck** in order to fix the remaining inconsistencies.

N1 of N2 blocks used (N3 free); N4 of N5 inodes used (N6 free)

The indicated number of blocks and inodes have been used, leaving the indicated number unallocated.

Unconnected files still remain. Mount the file system and remove files to free data blocks and inodes

The **fsck** program has successfully completed checking the file system and it has been marked as mountable. However, there are still unreferenced files in the file system. These unreferenced files can be recovered by running **fsck** again after enough blocks and inodes have been freed to allow them room to be reconnected.

End of Appendix

Appendix E
Using SCSI Devices

This appendix contains information about the installation and use of the following SCSI devices:

- CDROM device.
- Mulitple-read/write magneto-optical disk device.
- Diskette device.

For information about devices in general, see the section in Chapter 2, "Device Specifications: General Background," and Appendix A, "Device Names and Codes."

General Information

The information in this section applies to all SCSI devices.

Make sure you have the proper terminator on the last device in a chain of SCSI devices; otherwise, your system will panic.

If the total length of SCSI cable for a particular adapter is greater than 19 feet, you could get errors about problems in a device's physical file table. Excessive cable length could also result in problems when trying to access the last device on the chain.

If you have multiple CDROM devices, each must have a unique SCSI ID. Multiple magneto-optical or diskette devices, on the other hand, may be clustered so that as many as four share the same SCSI ID. If you put multiple units on the same SCSI ID, you will need to use device specifications that include third arguments (for unit number). For example, the following specifications represent three 5.25-inch diskette devices at SCSI ID 3:

```
sd(insc(),3,0)
sd(insc(),3,1)
sd(insc(),3,2)
```

For every device on your system, there is a short name created at boot time that you can use to refer to the device. You can find a table showing the long name/short name pairs in the file **/etc/devlinktab**. For example, a workstation's SCSI tape device has this specification in DG/UX common format:

```
st(insc(),4)
```

With this name in mind, you can locate the device's entry (which has a very similar long name) in the **devlinktab** file. The example below shows the device's entry (as well as the comment lines from the file that have the table headers).

```
# directory     short     long
#
/dev/rmt        0         st(insc@7(FFF8A000),4,0)
```

From this entry, you know that the short name for **st(insc(),4)** is **/dev/rmt/0**. The device is a tape device, so it has a no-rewind version too, which is **/dev/rmt/0n**.

You use a device's short name when you need to write to it (like a tape) or mount it as a filesystem).

The precise adapter specification, SCSI id, and unit number depends on the kind of hardware you have and how the jumpers are set.

Using the CDROM Device

After inserting a CD into a CDROM device, there are two things you must do before you can use it:

1. You must register the device. As superuser, use the **diskman** utility to register the device. You may invoke **diskman** in either of the following ways:

   # sysadm diskmgmt ↵

   or

   # /usr/sbin/diskman ↵

2. After registering the device, mount it as file system type **cdrom**. If the short name for your CDROM device were **/dev/pdsk/4**, you mount it onto **/cd_dir** like this:

   # mount −t cdrom /dev/pdsk/4 /cd_dir ↵

   An **/etc/fstab** entry for the device could look like this:

   ```
   /dev/pdsk/4      /cd_dir     cdrom  ro  x  0
   ```

To take a CD out of the device, unmount the file system first, then deregister the device. You cannot remove a CD without deregistering and unmounting it first.

Using the Magneto-Optical Device

After inserting a magneto-optical disk into the device, there are some things you must do before you can use it as a file system:

1. You must initialize the device. As superuser, invoke the **diskman** one of these two ways:

   # sysadm diskmgmt ♪

   or

   # /usr/sbin/diskman ♪

2. In the Diskman Main Menu, select option 1, Physical Disk Management Menu, then select option 5, Format a Physical Disk. Select option 1, Install a Disk Label on a Physical Disk, selecting OPTICAL SCSI when prompted.

3. After installing the label, select option 2, Perform Hardware Formatting on a Physical Disk. When formatting is finished, select option 3, Create DG/UX System Areas on Physical Disk.

4. After creating system areas on the disk, return to the Diskman Main Menu and select option 2, Create a Logical Disk. Create a file system on the disk.

5. After creating the file system, mount it as file system type **dg/ux**. The following command mounts the file system on the device **/dev/pdsk/5** on directory **/op_disk**.

   # mount −t dg/ux /dev/pdsk/5 /wmrm_fs

   An **/etc/fstab** entry for the device should follow this form:

   ```
   /dev/pdsk/5      /wmrm_fs     dg/ux  rw  x  0
   ```

To take a magneto-optical disk out of the device, first unmount the file system, then deregister the device. You cannot remove a disk without deregistering and unmounting it first.

You may use a magneto-optical device as a tape as well as a filesystem. Like a tape, you can write to the disk using **tar**(1) or **cpio**(1). Before you can use the device this way, you must register the disk.

You cannot use **sysadm fsdump** to dump to a magneto-optical device. You may dump to a magneto-optical disk from the shell using the **dump2**(1M) command, but you are limited to one disk.

Using the Diskette Device

This section contains information specific to diskette devices.

Disk Format

The 3.5-inch diskette device supports 720KB and 1.44MB formats. The 5.25-inch diskette device supports 360KB, 720KB, and 1.2MB formats. Obtain pre-formatted diskettes from your DG representative, or use DG RBOS diagnostics to format the diskettes.

Assigning Unit Numbers

If you intend to install several diskette devices on the same SCSI ID, you need to observe these rules:

- You may have four devices total, units **0** through **3**.
- If the devices on the SCSI ID are all 5.25-inch diskette devices, they may have any of the four available unit numbers (make sure the hardware is jumpered correctly).
- If any of the devices on the SCSI ID is a 3.5-inch diskette device, then only units 0 and 1 may be used for 3.5-inch devices, and units 3 and 4 are reserved for 5.25-inch devices. If you have only 3.5-inch devices, they are still restricted to units 0 and 1.

For example:

- If you have three 5.25-inch diskette devices, they can be units 0 through 2.
- If you decide to add a 3.5-inch device to the same adapter with the three 5.25-inch devices, however, the 3.5-inch device may to be unit 0 or 1, and two of the 5.25-inch devices may be units 2 and 3. The third 5.25-inch device must use a different SCSI ID.
- If you have three 3.5-inch devices, you may put two on the same SCSI ID, but the third must go at a different SCSI ID.

You need to make sure the SCSI ID, unit number, and adapter card straps are set correctly. The adapter card has the SCSI ID settings and the strap settings, and the device itself has the unit setting.

If your peripheral housing unit contains any 3.5-inch diskette devices, set the straps on the TEAC adapter card to H, F, LEV, STL, and PAR. If the housing unit contains only 5.25-inch diskette devices, set the straps on the TEAC adapter card to G, F, LEV, STL, and PAR. The H is for 1.44MB format, G is for 1.2MB format, and F is for 720KB format.

If you have two 5.25-inch diskette devices in the same peripheral housing unit, set the IS strap so the drive does not reset the ready state every time the drive changes density between the two 5.25-inch diskette devices.

If you have two 3.5-inch diskette devices in the same peripheral housing unit, set the HHI strap so that high density input is active high when a high density diskette is accessed and active low when a low density diskette is accessed.

Changing Diskettes

To take a diskette out of the device, first unmount the file system (if mounted) and deregister the device (if registered). Removing a diskette that is mounted and/or registered results in an error like this:

```
From System:
LDU logical_disk sealed,
                          Status nnnnnnn
        Run fsck to restore
```

or an error like this:

```
LDU is no longer fault tolerant
```

where *logical_disk* is the name of the logical disk, and *nnnnnn* is a status number.

If either kind of error occurs, you must re-insert the diskette and run **fsck**(1M) on it to recover it.

Using a Diskette as a File System

Before you can use a diskette as a file system, you must create a file system on it and mount the file system. You may do this with **diskman**, or you may do it yourself from the shell command line.

The advantage of building a file system yourself is in the diskette space you can safe. The disadvantage is that you gain this space by eliminating the bad block remapping table on the diskette. See Chapter 7 for more information on bad block remapping.

When **diskman** builds a file system on a logical disk, it allocates space for system areas, the bad block remapping table, and other things. The bad block table is big, and it can take up a considerable percentage of the space on a medium as small as a diskette. If you build the file system yourself, on the other hand, you can leave out the system areas, bad block remapping table, and so on, thereby saving space.

Using diskman to Create the File System

To use **diskman** to create the file system, follow these steps:

1. As superuser, invoke **diskman** like this:

   # sysadm diskmgmt ↵

   or like this:

   # /usr/sbin/diskman ↵

2. In the Diskman Main Menu, select option 1, Physical Disk Management Menu, then select option 5, Format a Physical Disk. Select option 1, Install a Disk Label on a Physical Disk, selecting any SCSI label when prompted.

3. After installing the label, select option 2, Perform Hardware Formatting on a Physical Disk. When formatting is finished, select option 3, Create DG/UX System Areas on Physical Disk.

4. After creating system areas on the disk, return to the Diskman Main Menu and select option 2, Create a Logical Disk. Create a file system on the disk.

5. After initializing the device, mount it as file system type **dg/ux**:

   # mount −t dg/ux /dev/pdsk/6 /floppy ↵

   An **/etc/fstab** entry for the device should follow this form:

   ```
   /dev/pdsk/6      /floppy     dg/ux  hard,rw  x  0
   ```

The file system is now ready to use.

Creating the File System from the Shell

To build the file system yourself, follow these steps in the shell (you must be superuser):

1. Invoke **mkfs**(1M) from the shell command line, specifying the diskette device's character-special (raw) physical device name (the one in the **/dev/rpdsk** directory). For example, if the diskette's character-special physical device name is **/dev/rpdsk/5**, create the file system like this:

   # mkfs /dev/rpdsk/5 ↵

2. Then use the **mount**(1M) to mount the file system, this time speciying the block-special physical device name (the one in the **/dev/pdsk** directory). For example, if the diskette device's block-special physical device name is **/dev/pdsk**, mount the file system like this:

```
# mount -t dg/ux /dev/pdsk/5 /floppy ♪
```

An **/etc/fstab** entry for the device should follow this form:

```
/dev/pdsk/5      /floppy     dg/ux  hard,rw  x  0
```

Using the Diskette Device as a Tape

Instead of using the diskette device as a filesystem, you may use it as a tape. Like a tape, you can write to the diskette using **tar**, **cpio**, **dump**, **dump2**, or **dd**. Refer to the device as **/dev/rpdsk** or **/dev/pdsk**. Before you can use the device this way, you must unmount it (if mounted) and deregister it (if registered).

Because diskettes do not have tape marks (as found on magnetic tape), you cannot use multiple diskettes with **tar**, **dump**, **dump2**, or **dd**. The **cpio** command, on the other hand, allows you to use multiple diskettes when archiving files.

You cannot use **sysadm fsdump** to dump to a diskette device. You may dump to a diskette from the shell using the **dump2**(1M) command, but you are limited to one diskette.

If you boot the system while a diskette with a **tar** or **cpio** format file is in the diskette device, you may see an error having to do with the physical file table. Ignore this error. If the diskette device is registered as a filesystem when you try to write to it as a file (with **tar** or **cpio**), you will receive the error, "conflict on open." Deregister the device.

Transferring Data Between DG/UX systems and 386/ix Systems

You can exchange **cpio**(1) and **tar**(1) files between 386/ix 2.0 and DG/UX 4.30 systems using 1.2MB and 360KB formats for 5.25-inch diskettes and 1.44MB for 3.5-inch diskettes. Hardware limitations prevent 386/ix systems from reading 720KB format diskettes.

When you use **cpio** to create files that you intend to move from a DG/UX system to a 386/ix system, use the **-c** option. For files that you intend to move from a 386/ix system to a DG/UX system, the **-c** option is not necessary. The default block sizes for **tar** on 386/ix and DG/UX 4.30 systems are compatible.

End of Appendix

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

This appendix tells how to install the 4.30 release of the DG/UX system on an installed 4.20 release of the DG/UX system. If you are installing the DG/UX system on a new (uninstalled) base system, see Chapter 2.

Table F-1 shows the default sizes for the logical disks required for release 4.30 of the DG/UX system.

Table F-1 Default Sizes for DG/UX 4.30 System Logical Disks

Logical Disk Unit	Size in 512-Byte Blocks
root	40,000
swap	50,000*
usr	160,000
usr_opt_X11	105,000**

* 32,768 for 179MB disks.

** Includes the Looking Glass (X11.lg) package.

Booting diskman from Tape

When you boot **diskman** from tape, you see a message from **init** about not being able to run **fsck** to check the **/usr** file system. In this one case, you can ignore the message. (Stand-alone **diskman** is actually a memory based kernel, so there is no **usr** file system.) If the message occurs under any other circumstances (as when booting any kernel from disk), you need to run **fsck** on the **usr** logical disk and mount **/usr** before changing run levels.

Loading the 4.30 Release on a 4.20 System

When loading the 4.30 system on an installed 4.20 system, you should check the amount of free space in the **/usr** file system before loading 4.30. The **/usr** file system in 4.30 is 7,500 blocks (3.75MB) larger than the 4.20 **/usr** file system. We assume that you have followed the guidelines in Chapter 2 and maintained the **usr** file system as a read-only file system containing only DG/UX system software.

If you have the 4.21 release of the DG/UX system loaded on your system, you should first remove (or move to another file system) the file **/usr/stand/dgux.diskless.4.21**.

To see how much free space remains in the **/usr** file system, issue this command:

```
# df -g /usr ↵
```

There should be close to 16,000 free blocks (as opposed to available blocks) in **/usr**. If you do not have at least 12,000 free blocks, do NOT attempt to load the 4.30 **/usr** file system on the existing **/usr** file system. Instead, recreate the **usr** logical disk, making it large enough to hold both the 4.30 **/usr** release plus whatever other software (besides the DG/UX system software) you have moved into **/usr**.

Loading the X11 Package

You should not attempt to use the 4.20 X11 package on a 4.30 system. Conversely, you should not attempt to use the 4.30 X11 package on a 4.20 system. If you intend to keep the 4.20 X11 package for use with a 4.20 DG/UX system secondary release, there are several things you need to do:

1. Create a new logical disk for the 4.20 X11 package. It should be the exact same size as your current **usr_opt_X11** logical disk (105,000 blocks was the default and recommended size for 4.20 X11). You should give the new logical disk a different name, like **X11_420**.

2. Copy the 4.20 **usr_opt_X11** logical disk to the new logical disk by invoking **diskman**, selecting the Logical Disk Management Menu, and then option 4, Copy a Logical Disk.

3. Use **sysadm addfsys** to mount the 4.20 X11 file system in an appropriate place under your 4.20 DG/UX system secondary release (like **/srv/release/88k_dgux_420/usr/opt/X11**, for example).

You may now load and set up the 4.30 X11 package on the **usr_opt_X11** logical disk.

Preserving Old System Files

If you are loading the 4.30 release of the DG/UX system on a 4.20 system or an "early" version of the 4.30 system, your **/var/Build/*system** files will be renamed the first time you change run levels. The old system files are renamed so that they have **_4.20** appended to the name.

Updating Client Root Areas

After loading the DG/UX release, OS servers must mount the **/srv/release/PRIMARY/root/*** file systems before changing run levels for the first time. This is necessary in order to trigger the update of the client root areas.

Configuring TCP/IP

If your system is on a subnetted TCP/IP network, add **routed** to the list of daemons at the end of **/etc/tcpip.params** *or* set the **ROUTE_*** parameters in **/etc/tcpip.params** to the appropriate values.

Building Kernels

When you use **sysadm newdgux** to build your 4.30 kernel, a new **system.aviion** file will be created from all the prototype system files in **/usr/src/uts/aviion/cf**. Do not build a 4.30 kernel until all 4.30 system software packages (like the DG/UX system, ONC/NFS, and TCP/IP) have been loaded.

Do not build a 4.30 kernel using 4.20 system files.

End of Appendix

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

You can install the DG/UX system on a tapeless stand-alone workstation. A tapeless stand-alone workstation has one or more disks but no tape device. You can install such a system only if the tapeless workstation is on a network that includes an OS server running DG/UX 4.30 that has enough disk space to set the workstation up as a client temporarily. The server should also have a tape device.

Installation Procedure

The installation procedure requires that you perform some steps on the server with the tape device and other steps on the workstation.

Before starting the procedure on the server, power up the workstation to get its Ethernet address. You also need to know the Internet address of the workstation (see your network administrator) and the host name you want the workstation to have.

Setting Up the Workstation's OS Client Space

Begin the installation by first following these steps on the server:

1. Use **sysadm newdgux** to build a kernel for the tapeless workstation (call the kernel something like **dgux.tapeless**). This kernel should be the same as the usual diskless client's kernel except that it should include the specification(s) for the tapeless workstation's disk(s).

   Move this kernel to the **_Kernels** directory of a release area; for example:

   ```
   # mv /dgux.tapeless /srv/release/PRIMARY/root/_Kernels ↵
   ```

2. Use **sysadm addether** to add an entry for the workstation to the server's **/etc/ethers** file.

3. Use **sysadm addhost** to add an entry for the workstation to the server's **/etc/hosts** file.

4. Add an entry for the tapeless system to the **/etc/hosts.equiv** file. An entry consists of a single line containing the tapeless system's host name.

5. Use **sysadm addclient** to attach the tapeless workstation to the release area where you put the tapeless kernel. Answer the questions as you would for any diskless client, except that for swap size, specify **4m**, and for the kernel, specify the path of the tapeless kernel. For example:

```
...
Swap size? 4m ♪
...
Kernel? /srv/release/PRIMARY/root/_Kernels/dgux.tapeless ♪
...
```

   See Chapter 6 for information on adding a client to a release.

Setting Up the Workstation

After completing the steps above, perform the following steps on the tapeless workstation:

1. Boot the workstation over the net:

```
SCM> b inen() ♪
```

2. Copy some necessary prototype files by issuing these commands:

```
# cp /etc/services.proto /etc/services ♪
# cp /etc/pmttapetab.proto /etc/pmttapetab ♪
# cp /etc/pmterrtab.proto /etc/pmterrtab ♪
```

3. Start the daemon for the magnetic tape pseudo-device:

```
# pmtd ♪
```

4. Make sure your term variable is set to **vt100**. Use these Bourne shell command lines:

```
TERM=vt100
export TERM
```

5. Execute **/usr/stand/diskman** and install the DG/UX system. Proceed according to the instructions starting at Step 6 except for these two points:

   - At the `Device Name?` prompt, enter **kbd**() and your LAN controller device. For a workstation, this is typically **inen**(). If you are attached to the net via a Hawk LAN controller, on the other hand, you should enter **hken**() instead.

   - When **diskman** queries you for the tape device name, specify the remote tape device like this:

host:*device*

where *host* is the host name of the remote system where the tape device is located, and *device* is the path name of the tape device on the remote system. For example:

```
Enter the tape drive specification in DG/UX common
format: sales:/dev/rmt/0 ↵
```

When you specify a remote tape as above, **diskman** checks to see if the device already has an entry in **/etc/pmttapetab**. If it does not, **diskman** adds an entry, querying you in the process about the type of operating system you are running. For the DG/UX system, enter **dg**.

Continue to load and install the system. Create the **swap**, **root**, and **usr** logical disks and load the **usr** and **root** file systems. If you intend to install the X11 package, create the **usr_opt_X11** logical disk and load the X11 package.

6. When you have completed the installation of the DG/UX system, halt the system:

   ```
   # /etc/halt -q ↵
   ```

7. Boot the workstation from disk:

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

   When prompted for starter devices, specify your keyboard (**kbd**()) and disk device (**sd(insc**(),**0)**).

8. Change to run level 1:

   ```
   # init 1 ↵
   ```

   Going to run level 1 the first time starts the DG/UX system setup scripts. When setup is complete, you receive a login prompt.

9. Log in as **sysadm** (you do not need a password):

   ```
   login: sysadm ↵
   ```

10. Create the **/srv** directory tree:

    ```
    # sysadm makesrv ↵
    ```

11. If you loaded X11 before, create an **/etc/fstab** entry for the **/usr/opt/X11** file system.

12. Halt the system:

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

13. Boot over the net:

    SCM> **b inen()** ↵

14. Having booted over the net, your current **root** and **usr** environment reside on the server, in the OS client release area. Next you need to create a minimal **root** environment on your local disk so you can use the **chroot**(1M) command to change your effective **root directory from the one on the server's disk to one on your local disk. The purpose of this step is to cause sysadm** to continue system installation and setup in your local **root** space instead of in the **root** space on the OS server.

    Execute the following commands (you mount **/usr/opt/X11** only if you have the X11 package):

    ```
    # mkdir /local_root ↵
    # mount /dev/dsk/root /local_root ↵
    # mount /dev/dsk/usr /local_root/usr ↵
    # mount /dev/dsk/usr_opt_X11 /usr/opt/X11 ↵
    # cp /etc/pmttapetab /local_root/etc ↵
    # cp /etc/pmterrtab /local_root/etc ↵
    # cp /etc/services /local_root/etc ↵
    # cp /etc/hosts /local_root/etc ↵
    # cp /usr/bin/pmtd /local_root/usr/bin ↵
    # chroot /local_root /bin/sh ↵
    # mkdir /dev/pmt ↵
    # mknod /dev/pmt/0 c 32 0 ↵
    # mknod /dev/pmt/0n c 32 1 ↵
    # mknod /dev/syscon c 0 0 ↵
    # pmtd ↵
    ```

15. Create an entry in **/etc/pmttapetab** for the no-rewind tape device. There is already an entry for the rewind tape device (**diskman** created it). If the tape driver you are using on the server is **/dev/rmt/0**, the entry looks like this:

    ```
    0  server  dg  /dev/rmt/0  /etc  512  N
    ```

    where the first field is the unit number in **/dev/pmt**, and *server* is the OS server host name.

    To create the no-rewind entry, duplicate this line and append **n** after each of the two unit numbers. Your **/etc/pmttapetab** file should now contain lines similar to these (the unit number may be different):

    ```
    0   server  dg  /dev/rmt/0   /etc  512  N
    0n  server  dg  /dev/rmt/0n  /etc  512  N
    ```

16. Load your application packages:

    # **sysadm loadpackage** ↵

Specify **PRIMARY** for the release area. You will then see this message:

```
Warning: The default tape device (/dev/rmt/0) has disappeared.
```

Ignore this message.

17. When **sysadm** prompts for a tape drive, respond as follows:

    `Tape Drive? [0]` **pmt/*unit*** ↵

    where *unit* is the unit number for the rewind device as it appears in your **/etc/pmttapetab** file.

18. Continue with the installation process starting with Step 16, "Setting up Software Packages with sysadm."

Removing the Workstation's OS Client Space

The last step you perform is on the OS server. Delete the workstation from the OS release:

```
# sysadm delclient ↵
```

Using a Remote Tape Device

Whenever you need to use the tape device on the remote system, refer to the tape device as **/dev/pmt/*n***, where *n* is the tape device's unit number.

End of Appendix

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

J

K

L

M

N

O

P

S

T

U

V

W

X

Y

Documentation Set

This section lists documents relevant to the AViiON product line. The titles of Data General manuals are followed by nine-digit numbers used for ordering; you can order any of these manuals via mail or telephone (see the TIPS Order Form in the back of this manual).

Following the list of Data General manuals are relevant documents published by other organizations (no ordering information is provided).

Documents specifically referred to in the text of this manual are also listed in the "Related Documents" section of the Preface.

Data General Software Manuals

88open Binary Compatibility Standard (069-701043)

Specifies a Binary Compatibility Standard (BCS).

88open Object Compatibility Standard (069-701044)

Specifies an Object Compatibility Standard (OCS) for operating systems based on Motorola MC88100 as well as future related microprocessors. Provides for portability of application-level software at the linkable level by specifying interfaces between the object file and the operating system libraries.

C: A Reference Manual (069-100226)

Describes lexical structure, the preprocessor, declarations, types, expressions, statements, functions, programs, and the run-time libraries.

Documenter's Tool Kit Technical Summary for the DG/UX™ System (069-701041)

Provides technical details about the tools supplied with the Documenter's Tool Kit; specifically, the **mm** macroinstruction package, the **tbl** table preprocessor, and the **nroff/troff** formatter.

Green Hills Software User's Manual C-88000 (069-100230)

Describes the differences in the C programming language when run on an 88000 system.

Green Hills Software User's Manual Fortran-88000 (069-100232)

Describes differences in the FORTRAN programming language when run on an 88000 system.

Green Hills Software User's Manual Pascal-88000 (069-100231)

Describes differences in the Pascal programming language when run on an 88000 system.

IEEE Standard Portable Operating System Interface for Computer Environments (POSIX.1) (069-701045)

Specifies a POSIX standard.

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

Shows how to install and manage the DG/UX operating system on AViiON hosts that will run as stand-alone, server, or client systems. Aimed at system administrators who are familiar with the UNIX operating system.

Learning the UNIX® Operating System (069-701042)

Helps beginners learn UNIX fundamentals through a step-by-step tutorial. (UNIX is a U.S. registered trademark of American Telephone and Telegraph Company.)

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.)

OSF/Motif™ Application Environment Specification (069-100326)

Specifies the interfaces that support the development of portable programs for OSF/Motif platforms.

OSF/Motif™ Programmer's Guide (069-100324)

A guide to programming using the various components of the OSF/Motif environment: the toolkit, window manager, and user interface language.

OSF/Motif™ Style Guide (069-100323)

Provides a framework for behavior specifications to guide application developers, widget developers, and window manager developers in the design of new products consistent with Presentation Manager and the OSF/Motif user interface.

Porting Applications to the DG/UX™ System (069-701059)

Describes how to port UNIX application programs to the DG/UX system.

POSIX.1 Conformance Document (069-701078)

Gives definitions and general requirements for conforming to the IEEE POSIX.1 standard.

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.

Programmer's Reference for the X.25 Provider Interface on the DG/UX™ System (093-701082)

Describes how to use the data structures and messages of the X.25 Provider Interface in application programs.

Programming in the DG/UX™ System Application Environment (093-701076)

Discusses libraries, interprocess communications, programming interface, common object file format, and other programming-related topics.

Programming with TCP/IP on the DG/UX™ System (093-701024)

Describes how to use the socket system calls to access TCP, UDP, and IP protocol software.

Setting Up and Managing PAD on the DG/UX™ System (093-701073)

Tells you how to set up and manage the Packet Assembler/Disassembler (PAD) for AViiON Systems package. Also contains manual pages for the PAD package.

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.

Setting Up and Managing X.25 on the DG/UX™ System (093-701071)

This manual is for X.25 wide area network system administrators. It describes how to set up and manage the X.25 for AViiON Systems software. It also contains manual pages for the X.25 package.

STREAMS Primer for the DG/UX™ System (069-701033)

Defines STREAMS, a set of tools for developing DG/UX system communications services; explains how to build a stream; and discusses user-level and kernel-level functions.

STREAMS Programmer's Guide for the DG/UX™ System (069-701034)

Describes the development methods and design philosophy of STREAMS.

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 API LU0,1,2,3 for AViiON™ Systems (093-000679)

Explains how to use the application program interface (API) for Logical Unit (LU) types 0, 1, 2, and 3 of IBM's System Network Architecture (SNA).

Using API LU6.2 for AViiON™ Systems (093-000680)

Explains how to use the application program interface (API) for Logical Unit (LU) type 6.2 of IBM's System Network Architecture (SNA).

Using PAD on the DG/UX™ System (069-701079)

Describes the user interface to the X.25 Packet Assembler/Disassembler (PAD) for AViiON Systems package.

Using SNA 3270 for AViiON™ Systems (093-000677)

Explains how to use the 3278 display and 3287 printer emulation capabilities within a multi-user environment.

Using SNA for AViiON™ Systems (093-000676)

Explains how to activate the SNA link to the host, establish node processes, and create configurations.

Using SNA/RJE for AViiON™ Systems (093-000678)

Explains how to use the 3776 emulation capabilities to submit jobs to and receive output from the host.

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™ Software Development Tools (093-701078)

Discusses programming support tools (**awk, nawk, lex, yacc, ld, lint,** and **as**), archiving, the C language, and SCCS.

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,**

Using the Documenter's Tool Kit on the DG/UX™ System (069-701039)

Provides a series of tutorials about the tools included in the Documenter's Tool Kit package. Describes the **mm** and **mv** macro instruction packages; the **tbl, eqn, pic,** and **grap** preprocessors; the tools **checkmm, diffmk, hyphen, ndx,** and **subj,** and the nroff/troff formatter.

Writing a Device Driver for the DG/UX™ System (093-701053)

Describes how to write a device driver for a DG/UX system running on an AViiON computer. Describes the drivers written to address specific devices or adapters that manage secondary bus access to specific devices.

xlib Programming Manual (069-100227)

Explains programming concepts and techniques for the X library, which is the lowest level programming interface to the X Window System.

xlib Reference Manual (069-100228)

Provides a programmer's reference to the X library, including information about functions, event types, macroinstructions, and structures.

X Window System User's Guide (069-100229)

Explains the X Window System and common client applications, and describes how to customize the X environment.

Other Organizations' Documents

To obtain any of the following documents, contact the indicated organization directly.

AIC-6250 High-Performance Protocol Chip data sheet (Adaptec)

Brooktree® Product Databook (Brooktree Corporation)

Local Area Controller Am7990 (LANCE) Technical Manual (Advance Micro Devices)

Memory Products Databook (SGS-Thompson Microelectronics)

Microprocessor Data Manual (Signetics)

System V Interface Definition (AT&T)

The VMEbus Specification (Motorola)

uPD72120 Advanced Graphics Display Controller User's Manual (NEC, Inc.)

Z8536 Z-CIP/Z8536 CIO Counter/Timer and Parallel I/O Unit (Zilog, Inc.)

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a) MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

      Send your order form with payment to:
      Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581-9973

   b) TELEPHONE – Call TIPS at (508) 870-1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a) Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.
   b) Check or Money Order – Make payable to Data General Corporation.
   c) Credit Card – A minimum order of $20 is required for Mastercard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1-4 Units	$5.00
5-10 Units	$8.00
11-40 Units	$10.00
41-200 Units	$30.00
Over 200 Units	$100.00

   If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$1-$149.99	0%
$150-$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870-1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME _____	COMPANY NAME _____
ATTN: _____	ATTN: _____
ADDRESS _____	ADDRESS (NO PO BOXES) _____
CITY _____	CITY _____
STATE _____ ZIP _____	STATE _____ ZIP _____

Priority Code _____ (See label on back of catalog)

_____ _____ _____ _____ _____
Authorized Signature of Buyer (Agrees to terms & conditions on reverse side) | Title | Date | Phone (Area Code) | Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS	ADD
1-4 Items	$ 5.00
5-10 Items	$ 8.00
11-40 Items	$ 10.00
41-200 Items	$ 30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0 - $149.99	0%
$150 - $499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	-
SUB TOTAL	
Your local* sales tax	+
Shipping and handling - See A	+
TOTAL - See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is _____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number _____ Expiration Date _____

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508-870-1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE
TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION

Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES

Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS

Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY

DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY

EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY

A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL

A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)

Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

134-755-02

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name _____ Your Title _____

Company _____ Phone _____

Street _____

City _____ State _____ Zip _____

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title _____ Manual No. _____

Who are you? ☐EDP/MIS Manager ☐Analyst/Programmer ☐Other _____
☐Senior Systems Analyst ☐Operator _____
☐Engineer ☐End User

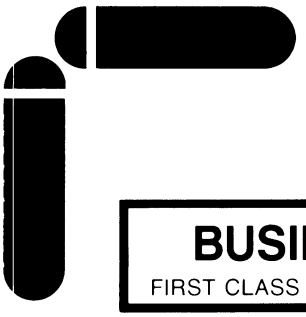
How do you use this manual? *(List in order: 1 = Primary Use)*

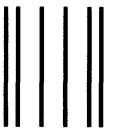
___ Introduction to the product ___ Tutorial Text ___ Other
___ Reference ___ Operating Guide _____

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only) or contact your sales representative or dealer.

Comments:





NO POST
NECESS
IF MAIL
IN TH
UNITED ST

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 WESTBORO, MA 01581

POSTAGE WILL BE PAID BY ADDRESSEE

DataGeneral

Customer Documentation
MS E-111
4400 Computer Drive
P.O. Box 4400
Westboro, MA 01581-9890

Cut here and insert in binder spine pocket

Data General

Data General Corporation, Westboro, Massachusetts 01580

093-701052-02